Advances in Electron Microscopy with Deep Learning

by

Jeffrey Mark Ede

Thesis

To be submitted to the University of Warwick

for the degree of

Doctor of Philosophy in Physics

Department of Physics

January 2021

Contents

Contents i

List of Abbreviations iii

List of Figures viii

List of Tables xvii

Acknowledgments xix

Declarations xx

Research Training xxv

Abstract xxvi

Preface xxvii
- I Initial Motivation . . . xxvii
- II Thesis Structure . . . xxvii
- III Connections . . . xxix

Chapter 1 Review: Deep Learning in Electron Microscopy 1
- 1.1 Scientific Paper . . . 1
- 1.2 Reflection . . . 100

Chapter 2 Warwick Electron Microscopy Datasets 101
- 2.1 Scientific Paper . . . 101
- 2.2 Amendments and Corrections . . . 133
- 2.3 Reflection . . . 133

Chapter 3 Adaptive Learning Rate Clipping Stabilizes Learning 136
- 3.1 Scientific Paper . . . 136
- 3.2 Amendments and Corrections . . . 147
- 3.3 Reflection . . . 147

Chapter 4 Partial Scanning Transmission Electron Microscopy with Deep Learning 149
- 4.1 Scientific Paper . . . 149
- 4.2 Amendments and Corrections . . . 176
- 4.3 Reflection . . . 176

Chapter 5 Adaptive Partial Scanning Transmission Electron Microscopy with Reinforcement Learning **178**
5.1 Scientific Paper 178
5.2 Reflection 202

Chapter 6 Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder **203**
6.1 Scientific Paper 203
6.2 Amendments and Corrections 212
6.3 Reflection 212

Chapter 7 Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning **214**
7.1 Scientific Paper 214
7.2 Reflection 239

Chapter 8 Conclusions **240**

References **243**

Vita **263**

List of Abbreviations

AE	Autoencoder
AFM	Atomic Force Microscopy
ALRC	Adaptive Learning Rate Clipping
ANN	Artificial Neural Network
ASPP	Atrous Spatial Pyramid Pooling
A-tSNE	Approximate t-Distributed Stochastic Neighbour Embedding
AutoML	Automatic Machine Learning
Bagged	Bootstrap Aggregated
bfloat16	16 Bit Brain Floating Point
BM3D	Block-Matching and 3D Filtering
BPTT	Backpropagation Through Time
CAE	Contractive Autoencoder
CBED	Convergent Beam Electron Diffraction
CBOW	Continuous Bag-of-Words
CCD	Charge-Coupled Device
cf.	*Confer*
Ch.	Chapter
CIF	Crystallography Information File
CLRC	Constant Learning Rate Clipping
CNN	Convolutional Neural Network
COD	Crystallography Open Database
COVID-19	Coronavirus Disease 2019
CPU	Central Processing Unit
CReLU	Concatenated Rectified Linear Unit
CTEM	Conventional Transmission Electron Microscopy
CTF	Contrast Transfer Function
CTRNN	Continuous Time Recurrent Neural Network
CUDA	Compute Unified Device Architecture
cuDNN	Compute Unified Device Architecture Deep Neural Network
DAE	Denoising Autoencoder
DALRC	Doubly Adaptive Learning Rate Clipping
DDPG	Deep Deterministic Policy Gradients

D-LACBED	Digital Large Angle Convergent Beam Electron Diffraction
DLF	Deep Learning Framework
DLSS	Deep Learning Supersampling
DNN	Deep Neural Network
DQE	Detective Quantum Efficiency
DSM	Doctoral Skills Module
EBSD	Electron Backscatter Diffraction
EDX	Energy Dispersive X-Ray
EE	Early Exaggeration
EELS	Electron Energy Loss Spectroscopy
e.g.	*Exempli Gratia*
ELM	Extreme Learning Machine
ELU	Exponential Linear Unit
EM	Electron Microscopy
EMDataBank	Electron Microscopy Data Bank
EMPIAR	Electron Microscopy Public Image Archive
EPSRC	Engineering and Physical Sciences Research Council
Eqn.	Equation
ESN	Echo-State Network
ETDB-Caltech	Caltech Electron Tomography Database
EWR	Exit Wavefunction Reconstruction
FIB-SEM	Focused Ion Beam Scanning Electron Microscopy
Fig.	Figure
FFT	Fast Fourier Transform
FNN	Feedforward Neural Network
FPGA	Field Programmable Gate Array
FT	Fourier Transform
FT^{-1}	Inverse Fourier Transform
FTIR	Fourier Transformed Infrared
FTSR	Focal and Tilt Series Reconstruction
GAN	Generative Adversarial Network
GMS	Gatan Microscopy Suite
GPU	Graphical Processing Unit
GRU	Gated Recurrent Unit
GUI	Graphical User Interface

HPC	High Performance Computing
ICSD	Inorganic Crystal Structure Database
i.e.	*Id Est*
i.i.d.	Independent and Identically Distributed
IndRNN	Independently Recurrent Neural Network
JSON	Javascript Object Notation
KDE	Kernel Density Estimated
KL	Kullback-Leibler
LR	Learning Rate
LSTM	Long Short-Term Memory
LSUV	Layer-Sequential Unit-Variance
MAE	Mean Absolute Error
MDP	Markov Decision Process
MGU	Minimal Gated Unit
MLP	Multilayer Perceptron
MPAGS	Midlands Physics Alliance Graduate School
MTRNN	Multiple Timescale Recurrent Neural Network
MSE	Mean Squared Error
N.B.	*Nota Bene*
NiN	Network-in-Network
NIST	National Institute of Standards and Technology
NMR	Nuclear Magnetic Resonance
MSE	Neural Network Exchange Format
NTM	Neural Turing Machine
ONNX	Open Neural Network Exchange
OpenCL	Open Computing Language
OU	Ornstein-Uhlenbeck
PCA	Principal Component Analysis
PDF	Probability Density Function **or** Portable Document Format

PhD	Doctor of Philosophy
POMDP	Partially Observed Markov Decision Process
PReLU	Parametric Rectified Linear Unit
PSO	Particle Swarm Optimization
RADAM	Rectified ADAM
RAM	Random Access Memory
RBF	Radial Basis Function
RDPG	Recurrent Deterministic Policy Gradients
ReLU	Rectified Linear Unit
REM	Reflection Electron Microscopy
RHEED	Reflection high-energy electron diffraction
RHEELS	Reflection High Electron Energy Loss Spectroscopy
RL	Reinforcement Learning
RMLP	Recurrent Multilayer Perceptron
RMS	Root Mean Squared
RNN	Recurrent Neural Network
RReLU	Randomized Leaky Rectified Linear Unit
RTP	Research Technology Platform
RWI	Random Walk Initialization
SAE	Sparse Autoencoder
SELU	Scaled Exponential Linear Unit
SEM	Scanning Electron Microscopy
SGD	Stochastic Gradient Descent
SNE	Stochastic Neighbour Embedding
SNN	Self-Normalizing Neural Network
SPLEEM	Spin-Polarized Low-Energy Electron Microscopy
SSIM	Structural Similarity Index Measure
STM	Scanning Tunnelling Microscopy
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TEM	Transmission Electron Microscopy
TIFF	Tag Image File Format
TPU	Tensor Processing Unit
tSNE	t-Distributed Stochastic Neighbour Embedding
TV	Total Variation

URL	Uniform Resource Locator
US-tSNE	Uniformly Separated t-Distributed Stochastic Neighbour Embedding
VAE	Variational Autoencoder
VAE-GAN	Variational Autoencoder Generative Adversarial Network
VBN	Virtual Batch Normalization
VGG	Visual Geometry Group
WDS	Wavelength Dispersive Spectroscopy
WEKA	Waikato Environment for Knowledge Analysis
WEMD	Warwick Electron Microscopy Datasets
WLEMD	Warwick Large Electron Microscopy Datasets
w.r.t.	With Respect To
XAI	Explainable Artificial Intelligence
XPS	X-Ray Photoelectron Spectroscopy
XRD	X-Ray Diffraction
XRF	X-Ray Fluorescence

List of Figures

Preface

1. Connections between publications covered by chapters of this thesis. An arrow from chapter x to chapter y indicates that results covered by chapter y depend on results covered by chapter x. Labels indicate types of research outputs associated with each chapter, and total connections to and from chapters.

Chapter 1 Review: Deep Learning in Electron Microscopy

1. Example applications of a noise-removal DNN to instances of Poisson noise applied to 512×512 crops from TEM images. Enlarged 64×64 regions from the top left of each crop are shown to ease comparison. This figure is adapted from our earlier work under a Creative Commons Attribution 4.0 license.
2. Example applications of DNNs to restore 512×512 STEM images from sparse signals. Training as part of a generative adversarial network yields more realistic outputs than training a single DNN with mean squared errors. Enlarged 64×64 regions from the top left of each crop are shown to ease comparison. a) Input is a Gaussian blurred 1/20 coverage spiral. b) Input is a 1/25 coverage grid. This figure is adapted from our earlier works under Creative Commons Attribution 4.0 licenses.
3. Example applications of a semantic segmentation DNN to STEM images of steel to classify dislocation locations. Yellow arrows mark uncommon dislocation lines with weak contrast, and red arrows indicate that fixed widths used for dislocation lines are sometimes too narrow to cover defects. This figure is adapted with permission under a Creative Commons Attribution 4.0 license.
4. Example applications of a DNN to reconstruct phases of exit wavefunction from intensities of single TEM images. Phases in $[-\pi, \pi)$ rad are depicted on a linear greyscale from black to white, and Miller indices label projection directions. This figure is adapted from our earlier work under a Creative Commons Attribution 4.0 license.
5. Reciprocity of TEM and STEM electron optics.
6. Numbers of results per year returned by Dimensions.ai abstract searches for SEM, TEM, STEM, STM and REM qualitate their popularities. The number of results for 2020 is extrapolated using the mean rate before 14th July 2020.
7. Visual comparison of various normalization methods highlighting regions that they normalize. Regions can be normalized across batch, feature and other dimensions, such as height and width.
8. Visualization of convolutional layers. a) Traditional convolutional layer where output channels are sums of biases and convolutions of weights with input channels. b) Depthwise separable convolutional layer where depthwise convolutions compute one convolution with weights for each input channel. Output channels are sums of biases and pointwise convolutions weights with depthwise channels.
9. Two 96×96 electron micrographs a) unchanged, and filtered by b) a 5×5 symmetric Gaussian kernel with a 2.5 px standard deviation, c) a 3×3 horizontal Sobel kernel, and d) a 3×3 vertical Sobel kernel. Intensities in a) and b) are in [0, 1], whereas intensities in c) and d) are in [-1, 1].
10. Residual blocks where a) one, b) two, and c) three convolutional layers are skipped. Typically, convolutional layers are followed by batch normalization then activation.

11. Actor-critic architecture. An actor outputs actions based on input states. A critic then evaluates action-state pairs to predict losses.

12. Generative adversarial network architecture. A generator learns to produce outputs that look realistic to a discriminator, which learns to predict whether examples are real or generated.

13. Architectures of recurrent neural networks with a) long short-term memory (LSTM) cells, and b) gated recurrent units (GRUs).

14. Architectures of autoencoders where an encoder maps an input to a latent space and a decoder learns to reconstruct the input from the latent space. a) An autoencoder encodes an input in a deterministic latent space, whereas a b) traditional variational autoencoder encodes an input as means, μ, and standard deviations, σ, of Gaussian multivariates, $\mu + \sigma \cdot \epsilon$, where ϵ is a standard normal multivariate.

15. Gradient descent. a) Arrows depict steps across one dimension of a loss landscape as a model is optimized by gradient descent. In this example, the optimizer traverses a small local minimum; however, it then gets trapped in a larger sub-optimal local minimum, rather than reaching the global minimum. b) Experimental DNN loss surface for two random directions in parameter space showing many local minima. The image in part b) is reproduced with permission under an MIT license.

16. Inputs that maximally activate channels in GoogLeNet after training on ImageNet. Neurons in layers near the start have small receptive fields and discern local features. Middle layers discern semantics recognisable by humans, such as dogs and wheels. Finally, layers at the end of the DNN, near its logits, discern combinations of semantics that are useful for labelling. This figure is adapted with permission under a Creative Commons Attribution 4.0 license.

Chapter 2 Warwick Electron Microscopy Datasets

1. Simplified VAE architecture. a) An encoder outputs means, $\boldsymbol{\mu}$, and standard deviations, $\boldsymbol{\sigma}$, to parameterize multivariate normal distributions, $\mathbf{z} \sim \mathrm{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$. b) A generator predicts input images from $\mathbf{z}$.

2. Images at 500 randomly selected points in two-dimensional tSNE visualizations of 19769 96×96 crops from STEM images for various embedding methods. Clustering is best in a) and gets worse in order a)→b)→c)→d).

3. Two-dimensional tSNE visualization of 64-dimensional VAE latent spaces for 19769 STEM images that have been downsampled to 96×96. The same grid is used to show a) map points and b) images at 500 randomly selected points.

4. Two-dimensional tSNE visualization of 64-dimensional VAE latent spaces for 17266 TEM images that have been downsampled to 96×96. The same grid is used to show a) map points and b) images at 500 randomly selected points.

Chapter 2 Supplementary Information: Warwick Electron Microscopy Datasets

S1. Two-dimensional tSNE visualization of the first 50 principal components of 19769 STEM images that have been downsampled to 96×96. The same grid is used to show a) map points and b) images at 500 randomly selected points.

S2. Two-dimensional tSNE visualization of the first 50 principal components of 19769 96×96 crops from STEM images. The same grid is used to show a) map points and b) images at 500 randomly selected points.

S3. Two-dimensional tSNE visualization of the first 50 principal components of 17266 TEM images that have been downsampled to 96×96. The same grid is used to show a) map points and b) images at 500 randomly selected points.

S4. Two-dimensional tSNE visualization of the first 50 principal components of 36324 exit wavefunctions that have been downsampled to 96×96. Wavefunctions were simulated for thousands of materials and a large range of physical hyperparameters. The same grid is used to show a) map points and b) wavefunctions at 500 randomly selected points. Red and blue colour channels show real and imaginary components, respectively.

S5. Two-dimensional tSNE visualization of the first 50 principal components of 11870 exit wavefunctions that have been downsampled to 96×96. Wavefunctions were simulated for thousands of materials and a small range of physical hyperparameters. The same grid is used to show a) map points and b) wavefunctions at 500 randomly selected points. Red and blue colour channels show real and imaginary components, respectively.

S6. Two-dimensional tSNE visualization of the first 50 principal components of 4825 exit wavefunctions that have been downsampled to 96×96. Wavefunctions were simulated for thousands of materials and a small range of physical hyperparameters. The same grid is used to show a) map points and b) wavefunctions at 500 randomly selected points. Red and blue colour channels show real and imaginary components, respectively.

S7. Two-dimensional tSNE visualization of means parameterized by 64-dimensional VAE latent spaces for 19769 STEM images that have been downsampled to 96×96. The same grid is used to show a) map points and b) images at 500 randomly selected points.

S8. Two-dimensional tSNE visualization of means parameterized by 64-dimensional VAE latent spaces for 19769 96×96 crops from STEM images. The same grid is used to show a) map points and b) images at 500 randomly selected points.

S9. Two-dimensional tSNE visualization of means parameterized by 64-dimensional VAE latent spaces for 19769 TEM images that have been downsampled to 96×96. The same grid is used to show a) map points and b) images at 500 randomly selected points.

S10. Two-dimensional tSNE visualization of means and standard deviations parameterized by 64-dimensional VAE latent spaces for 19769 96×96 crops from STEM images. The same grid is used to show a) map points and b) images at 500 randomly selected points.

S11. Two-dimensional uniformly separated tSNE visualization of 64-dimensional VAE latent spaces for 19769 96×96 crops from STEM images.

S12. Two-dimensional uniformly separated tSNE visualization of 64-dimensional VAE latent spaces for 19769 STEM images that have been downsampled to 96×96.

S13. Two-dimensional uniformly separated tSNE visualization of 64-dimensional VAE latent spaces for 17266 TEM images that have been downsampled to 96×96.

S14. Examples of top-5 search results for 96×96 TEM images. Euclidean distances between $\boldsymbol{\mu}$ encoded for search inputs and results are smaller for more similar images.

S15. Examples of top-5 search results for 96×96 STEM images. Euclidean distances between $\boldsymbol{\mu}$ encoded for search inputs and results are smaller for more similar images.

Chapter 3 Adaptive Learning Rate Clipping Stabilizes Learning

1. Unclipped learning curves for 2× CIFAR-10 supersampling with batch sizes 1, 4, 16 and 64 with and without adaptive learning rate clipping of losses to 3 standard deviations above their running means. Training is more stable for squared errors than quartic errors. Learning curves are 500 iteration boxcar averaged.

2. Unclipped learning curves for 2× CIFAR-10 supersampling with ADAM and SGD optimizers at stable and unstably high learning rates, η. Adaptive learning rate clipping prevents loss spikes and decreases errors at unstably high learning rates. Learning curves are 500 iteration boxcar averaged.

3. Neural network completions of 512×512 scanning transmission electron microscopy images from 1/20 coverage blurred spiral scans.

4. Outer generator losses show that ALRC and Huberization stabilize learning. ALRC lowers final mean squared error (MSE) and Huberized MSE losses and accelerates convergence. Learning curves are 2500 iteration boxcar averaged.

5. Convolutional image 2× supersampling network with three skip-2 residual blocks.

6. Two-stage generator that completes 512×512 micrographs from partial scans. A dashed line indicates that the same image is input to the inner and outer generator. Large scale features developed by the inner generator are locally enhanced by the outer generator and turned into images. An auxiliary inner generator trainer restores images from inner generator features to provide direct feedback.

Chapter 4 Partial Scanning Transmission Electron Microscopy with Deep Learning

1. Examples of Archimedes spiral (top) and jittered gridlike (bottom) 512×512 partial scan paths for 1/10, 1/20, 1/40, and 1/100 px coverage.

2. Simplified multiscale generative adversarial network. An inner generator produces large-scale features from inputs. These are mapped to half-size completions by a trainer network and recombined with the input to generate full-size completions by an outer generator. Multiple discriminators assess multiscale crops from input images and full-size completions. This figure was created with Inkscape.

3. Adversarial and non-adversarial completions for 512×512 test set 1/20 px coverage blurred spiral scan inputs. Adversarial completions have realistic noise characteristics and structure whereas non-adversarial completions are blurry. The bottom row shows a failure case where detail is too fine for the generator to resolve. Enlarged 64×64 regions from the top left of each image are inset to ease comparison, and the bottom two rows show non-adversarial generators outputting more detailed features nearer scan paths.

4. Non-adversarial generator outputs for 512×512 1/20 px coverage blurred spiral and gridlike scan inputs. Images with predictable patterns or structure are accurately completed. Circles accentuate that generators cannot reliably complete unpredictable images where there is no information. This figure was created with Inkscape.

5. Generator mean squared errors (MSEs) at each output pixel for 20000 512×512 1/20 px coverage test set images. Systematic errors are lower near spiral paths for variants of MSE training, and are less structured for adversarial training. Means, μ, and standard deviations, σ, of all pixels in each image are much higher for adversarial outputs. Enlarged 64×64 regions from the top left of each image are inset to ease comparison, and to show that systematic errors for MSE training are higher near output edges.

6. Test set root mean squared (RMS) intensity errors for spiral scans in $[0, 1]$ selected with binary masks. a) RMS errors decrease with increasing electron probe coverage, and are higher than deep learning supersampling (DLSS) errors. b) Frequency distributions of 20000 test set RMS errors for 100 bins in $[0, 0.224]$ and scan coverages in the legend.

S1. Discriminators examine random $w \times w$ crops to predict whether complete scans are real or generated. Generators are trained by multiple discriminators with different w. This figure was created with Inkscape.

S2. Two-stage generator that completes 512×512 micrographs from partial scans. A dashed line indicates that the same image is input to the inner and outer generator. Large scale features developed by the inner generator are locally enhanced by the outer generator and turned into images. An auxiliary trainer network restores images from inner generator features to provide direct feedback. This figure was created with Inkscape.

S3. Learning curves. a) Training with an auxiliary inner generator trainer stabilizes training, and converges to lower than two-stage training with fine tuning. b) Concatenating beam path information to inputs decreases losses. Adding symmetric residual connections between strided inner generator convolutions and transpositional convolutions increases losses. c) Increasing sizes of the first inner and outer generator convolutional kernels does not decrease losses. d) Losses are lower after more interations, and a learning rate (LR) of 0.0004; rather than 0.0002. Labels indicate inner generator iterations - outer generator iterations - fine tuning iterations, and k denotes multiplication by 1000 e) Adaptive learning rate clipped quartic validation losses have not diverged from training losses after 10^6 iterations. f) Losses are lower for outputs in [0, 1] than for outputs in [-1, 1] if leaky ReLU activation is applied to generator outputs.

S4. Learning curves. a) Making all convolutional kernels 3×3, and not applying leaky ReLU activation to generator outputs does not increase losses. b) Nearest neighbour infilling decreases losses. Noise was not added to low duration path segments for this experiment. c) Losses are similar whether or not extra noise is added to low-duration path segments. d) Learning is more stable and converges to lower errors at lower learning rates (LRs). Losses are lower for spirals than grid-like paths, and lowest when no noise is added to low-intensity path segments. e) Adaptive momentum-based optimizers, ADAM and RMSProp, outperform non-adaptive momentum optimizers, including Nesterov-accelerated momentum. ADAM outperforms RMSProp; however, training hyperparameters and learning protocols were tuned for ADAM. Momentum values were 0.9. f) Increasing partial scan pixel coverages listed in the legend decreases losses.

S5. Adaptive learning rate clipping stabilizes learning, accelerates convergence and results in lower errors than Huberisation. Weighting pixel errors with their running or final mean errors is ineffective.

S6. Non-adversarial 512×512 outputs and blurred true images for 1/17.9 px coverage spiral scans selected with binary masks.

S7. Non-adversarial 512×512 outputs and blurred true images for 1/27.3 px coverage spiral scans selected with binary masks.

S8. Non-adversarial 512×512 outputs and blurred true images for 1/38.2 px coverage spiral scans selected with binary masks.

S9. Non-adversarial 512×512 outputs and blurred true images for 1/50.0 px coverage spiral scans selected with binary masks.

S10. Non-adversarial 512×512 outputs and blurred true images for 1/60.5 px coverage spiral scans selected with binary masks.

S11. Non-adversarial 512×512 outputs and blurred true images for 1/73.7 px coverage spiral scans selected with binary masks.

S12. Non-adversarial 512×512 outputs and blurred true images for 1/87.0 px coverage spiral scans selected with binary masks.

Chapter 5 Adaptive Partial Scanning Transmission Electron Microscopy with Reinforcement Learning

1. Example 8×8 partial scan with $T = 5$ straight path segments. Each segment in this example has 3 probing positions separated by $d = 2^{1/2}$ px and their starts are labelled by step numbers, t. Partial scans are selected from STEM images by sampling pixels nearest probing positions, even if the probing position is nominally outside an imaging region.

2. Test set 1/23.04 px coverage partial scans, target outputs and generated partial scan completions for 96×96 crops from STEM images. The top four rows show adaptive scans, and the bottom row shows spiral scans. Input partial scans are noisy, whereas target outputs are blurred.

3. Learning curves for a-b) adaptive scan paths chosen by an LSTM or GRU, and fixed spiral and other fixed paths, c) adaptive paths chosen by an LSTM or DNC, d) a range of replay buffer sizes, e) a range of penalties for trying to sample at probing positions over image edges, and f) with and without normalizing or clipping generator losses used for critic training. All learning curves are 2500 iteration boxcar averaged and results in different plots are not directly comparable due to varying experiment settings. Means and standard deviations of test set errors, "Test: Mean, Std Dev", are at the ends of labels in graph legends.

Chapter 5 Supplementary Information: Adaptive Partial Scanning Transmission Electron Microscopy with Reinforcement Learning

S1. Actor, critic and generator architecture. a) An actor outputs action vectors whereas a critic predicts losses. Dashed lines are for extra components in a DNC. b) A convolutional generator completes partial scans.

S2. Learning curves for a) exponentially decayed and exponentially decayed cyclic learning rate schedules, b) actor training with differentiation w.r.t. live or replayed actions, c) images downsampled or cropped from full images to 96×96 with and without additional Sobel losses, d) mean squared error and maximum regional mean squared error loss functions, e) supervision throughout training, supervision only at the start, and no supervision, and f) projection from 128 to 64 hidden units or no projection. All learning curves are 2500 iteration boxcar averaged, and results in different plots are not directly comparable due to varying experiment settings. Means and standard deviations of test set errors, "Test: Mean, Std Dev", are at the ends of graph labels.

S3. Learning rate optimization. a) Learning rates are increased from $10^{-6.5}$ to $10^{0.5}$ for ADAM and SGD optimization. At the start, convergence is fast for both optimizers. Learning with SGD becomes unstable at learning rates around 2.2×10^{-5}, and numerically unstable near 5.8×10^{-4}, whereas ADAM becomes unstable around 2.5×10^{-2}. b) Training with ADAM optimization for learning rates listed in the legend. Learning is visibly unstable at learning rates of $2.5\times10^{-2.5}$ and 2.5×10^{-2}, and the lowest inset validation loss is for a learning rate of $2.5\times10^{-3.5}$. Learning curves in (b) are 1000 iteration boxcar averaged. Means and standard deviations of test set errors, "Test: Mean, Std Dev", are at the ends of graph labels.

S4. Test set 1/23.04 px coverage adaptive partial scans, target outputs, and generated partial scan completions for 96×96 crops from STEM images.

S5. Test set 1/23.04 px coverage adaptive partial scans, target outputs, and generated partial scan completions for 96×96 crops from STEM images.

S6. Test set 1/23.04 px coverage spiral partial scans, target outputs, and generated partial scan completions for 96×96 crops from STEM images.

Chapter 6 Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder

1. Simplified network showing how features produced by an Xception backbone are processed. Complex high-level features flow into an atrous spatial pyramid pooling module that produces rich semantic information. This is combined with simple low-level features in a multi-stage decoder to resolve denoised micrographs.
2. Mean squared error (MSE) losses of our neural network during training on low dose ($\ll 300$ counts ppx) and fine-tuning for high doses (200-2500 counts ppx). Learning rates (LRs) and the freezing of batch normalization are annotated. Validation losses were calculated using one validation example after every five training batches.
3. Gaussian kernel density estimated (KDE) MSE and SSIM probability density functions (PDFs) for the denoising methods in table 1. Only the starts of MSE PDFs are shown. MSE and SSIM performances were divided into 200 equispaced bins in $[0.0, 1.2] \times 10^{-3}$ and [0.0, 1.0], respectively, for both low and high doses. KDE bandwidths were found using Scott's Rule.
4. Mean absolute errors of our low and high dose networks' 512×512 outputs for 20000 instances of Poisson noise. Contrast limited adaptive histogram equalization has been used to massively increase contrast, revealing grid-like error variation. Subplots show the top-left 16×16 pixels' mean absolute errors unadjusted. Variations are small and errors are close to the minimum everywhere, except at the edges where they are higher. Low dose errors are in [0.0169, 0.0320]; high dose errors are in [0.0098, 0.0272].
5. Example applications of the noise-removal network to instances of Poisson noise applied to 512×512 crops from high-quality micrographs. Enlarged 64×64 regions from the top left of each crop are shown to ease comparison.
6. Architecture of our deep convolutional encoder-decoder for electron micrograph denoising. The entry and middle flows develop high-level features that are sampled at multiple scales by the atrous spatial pyramid pooling module. This produces rich semantic information that is concatenated with low-level entry flow features and resolved into denoised micrographs by the decoder.

Chapter 7 Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning

1. Wavefunction propagation. a) An incident wavefunction is perturbed by a projected potential of a material. b) Fourier transforms (FTs) can describe a wavefunction being focused by an objective lens through an objective aperture to a focal plane.
2. Crystal structure of $\mathrm{In}_{1.7}\mathrm{K}_2\mathrm{Se}_8\mathrm{Sn}_{2.28}$ projected along Miller zone axis [001]. A square outlines a unit cell.
3. A convolutional neural network generates $w \times w \times 2$ channelwise concatenations of wavefunction components from their amplitudes. Training MSEs are calculated for phase components, before multiplication by input amplitudes.
4. A discriminator predicts whether wavefunction components were generated by a neural network.
5. Frequency distributions show 19992 validation set mean absolute errors for neural networks trained to reconstruct wavefunctions simulated for multiple materials, multiple materials with restricted simulation hyperparameters, and $\mathrm{In}_{1.7}\mathrm{K}_2\mathrm{Se}_8\mathrm{Sn}_{2.28}$. Networks for $\mathrm{In}_{1.7}\mathrm{K}_2\mathrm{Se}_8\mathrm{Sn}_{2.28}$ were trained to predict phase components directly; minimising squared errors, and as part of generative adversarial networks. To demonstrate robustness to simulation physics, some validation set errors are shown for $n = 1$ and $n = 3$ simulation physics. We used up to three validation sets, which cumulatively quantify the ability of a network to generalize to unseen transforms consisting of flips, rotations and translations; simulation hyperparameters, such as thickness and voltage; and materials. A vertical dashed line indicates an expected error of 0.75 for random phases, and frequencies are distributed across 100 bins.

6. Training mean absolute errors are similar with and without adaptive learning rate clipping (ALRC). Learning curves are 2500 iteration boxcar averaged.

7. Exit wavefunction reconstruction for unseen NaCl, B_3BeLaO_7, $PbZr_{0.45}Ti_{0.55}0_3$, CdTe, and Si input amplitudes, and corresponding crystal structures. Phases in $[-\pi, \pi)$ rad are depicted on a linear greyscale from black to white, and show that output phases are close to true phases. Wavefunctions are cyclically periodic functions of phase so distances between black and white pixels are small. Si is a failure case where phase information is not accurately recovered. Miller indices label projection directions.

Chapter 7 Supplementary Information: Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning

S1. Input amplitudes, target phases and output phases of 224×224 multiple material training set wavefunctions for unseen flips, rotations and translations, and $n = 1$ simulation physics.

S2. Input amplitudes, target phases and output phases of 224×224 multiple material validation set wavefunctions for seen materials, unseen simulation hyperparameters, and $n = 1$ simulation physics.

S3. Input amplitudes, target phases and output phases of 224×224 multiple material validation set wavefunctions for unseen materials, unseen simulation hyperparameters, and $n = 1$ simulation physics.

S4. Input amplitudes, target phases and output phases of 224×224 multiple material training set wavefunctions for unseen flips, rotations and translations, and $n = 3$ simulation physics.

S5. Input amplitudes, target phases and output phases of 224×224 multiple material validation set wavefunctions for seen materials, unseen simulation hyperparameters, and $n = 3$ simulation physics.

S6. Input amplitudes, target phases and output phases of 224×224 multiple material validation set wavefunctions for unseen materials, unseen simulation hyperparameters are unseen, and $n = 3$ simulation physics.

S7. Input amplitudes, target phases and output phases of 224×224 validation set wavefunctions for restricted simulation hyperparameters, and $n = 3$ simulation physics.

S8. Input amplitudes, target phases and output phases of 224×224 validation set wavefunctions for restricted simulation hyperparameters, and $n = 3$ simulation physics.

S9. Input amplitudes, target phases and output phases of 224×224 $In_{1.7}K_2Se_8Sn_{2.28}$ training set wavefunctions for unseen flips, rotations and translations, and $n = 1$ simulation physics.

S10. Input amplitudes, target phases and output phases of 224×224 $In_{1.7}K_2Se_8Sn_{2.28}$ validation set wavefunctions for unseen simulation hyperparameters, and $n = 1$ simulation physics.

S11. Input amplitudes, target phases and output phases of 224×224 validation set wavefunctions for unseen simulation hyperparameters and materials, and $n = 1$ simulation physics. The generator was trained with $In_{1.7}K_2Se_8Sn_{2.28}$ wavefunctions.

S12. Input amplitudes, target phases and output phases of 224×224 $In_{1.7}K_2Se_8Sn_{2.28}$ training set wavefunctions for unseen flips, rotations and translations, and $n = 1$ simulation physics.

S13. Input amplitudes, target phases and output phases of 224×224 $In_{1.7}K_2Se_8Sn_{2.28}$ validation set wavefunctions for unseen simulation hyperparameters, and $n = 3$ simulation physics.

S14. Input amplitudes, target phases and output phases of 224×224 validation set wavefunctions for unseen simulation hyperparameters and materials, and $n = 3$ simulation physics. The generator was trained with $\mathrm{In}_{1.7}\mathrm{K}_2\mathrm{Se}_8\mathrm{Sn}_{2.28}$ wavefunctions.

S15. GAN input amplitudes, target phases and output phases of 144×144 $\mathrm{In}_{1.7}\mathrm{K}_2\mathrm{Se}_8\mathrm{Sn}_{2.28}$ validation set wavefunctions for unseen flips, rotations and translations, and $n = 1$ simulation physics.

S16. GAN input amplitudes, target phases and output phases of 144×144 $\mathrm{In}_{1.7}\mathrm{K}_2\mathrm{Se}_8\mathrm{Sn}_{2.28}$ validation set wavefunctions for unseen simulation hyperparameters, and $n = 1$ simulation physics.

S17. GAN input amplitudes, target phases and output phases of 144×144 $\mathrm{In}_{1.7}\mathrm{K}_2\mathrm{Se}_8\mathrm{Sn}_{2.28}$ validation set wavefunctions for unseen flips, rotations and translations, and $n = 3$ simulation physics.

S18. GAN input amplitudes, target phases and output phases of 144×144 $\mathrm{In}_{1.7}\mathrm{K}_2\mathrm{Se}_8\mathrm{Sn}_{2.28}$ validation set wavefunctions for unseen simulation hyperparameters, and $n = 3$ simulation physics.

List of Tables

Preface

1. Word counts for papers included in thesis chapters, the remainder of the thesis, and the complete thesis.

Chapter 1 Review: Deep Learning in Electron Microscopy

1. Deep learning frameworks with programming interfaces. Most frameworks have open source code and many support multiple programming languages.
2. Microjob service platforms. The size of typical tasks varies for different platforms and some platforms specialize in preparing machine learning datasets.

Chapter 2 Warwick Electron Microscopy Datasets

1. Examples and descriptions of STEM images in our datasets. References put some images into context to make them more tangible to unfamiliar readers.
2. Examples and descriptions of TEM images in our datasets. References put some images into context to make them more tangible to unfamiliar readers.

Chapter 2 Supplementary Information: Warwick Electron Microscopy Datasets

S1. To ease comparison, we have tabulated figure numbers for tSNE visualizations. Visualizations are for principal components, VAE latent space means, and VAE latent space means weighted by standard deviations.

Chapter 3 Adaptive Learning Rate Clipping Stabilizes Learning

1. Adaptive learning rate clipping (ALRC) for losses 2, 3, 4 and ∞ running standard deviations above their running means for batch sizes 1, 4, 16 and 64. ARLC was not applied for clipping at ∞. Each squared and quartic error mean and standard deviation is for the means of the final 5000 training errors of 10 experiments. ALRC lowers errors for unstable quartic error training at low batch sizes and otherwise has little effect. Means and standard deviations are multiplied by 100.
2. Means and standard deviations of 20000 unclipped test set MSEs for STEM supersampling networks trained with various learning rate clipping algorithms and clipping hyperparameters, $n^{\uparrow}$ and $n^{\downarrow}$, above and below, respectively.

Chapter 4 Partial Scanning Transmission Electron Microscopy with Deep Learning

1. Means and standard deviations of pixels in images created by takings means of 20000 512×512 test set squared difference images with intensities in [-1, 1] for methods to decrease systematic spatial error variation. Variances of Laplacians were calculated after linearly transforming mean images to unit variance.

Chapter 6 Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder

1. Mean MSE and SSIM for several denoising methods applied to 20000 instances of Poisson noise and their standard errors. All methods were implemented with default parameters. Gaussian: 3×3 kernel with a 0.8 px standard deviation. Bilateral: 9×9 kernel with radiometric and spatial scales of 75 (scales below 10 have little effect while scales above 150 cartoonize images). Median: 3×3 kernel. Wiener: no parameters. Wavelet: BayesShrink adaptive wavelet soft-thresholding with wavelet detail coefficient thresholds estimated using . Chambolle and Bregman TV: iterative total-variation (TV) based denoising, both with denoising weights of 0.1 and applied until the fractional change in their cost function fell below 2.0×10^{-4} or they reached 200 iterations. Times are for 1000 examples on a 3.4 GHz i7-6700 processor and 1 GTX 1080 Ti GPU, except for our neural network time, which is for 20000 examples.

Chapter 7 Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning

1. New datasets containing 98340 wavefunctions simulated with clTEM are split into training, unseen, validation, and test sets. Unseen wavefunctions are simulated for training set materials with different simulation hyperparameters. Kirkland potential summations were calculated with $n = 3$ or truncated to $n = 1$ terms, and dashes (-) indicate subsets that have not been simulated. Datasets have been made publicly available at .

2. Means and standard deviations of 19992 validation set errors for unseen transforms (trans.), simulations hyperparameters (param.) and materials (mater.). All networks outperform a baseline uniform random phase generator for both $n = 1$ and $n = 3$ simulation physics. Dashes (-) indicate that validation set wavefunctions have not been simulated.

Acknowledgments

Most modern research builds on a high variety of intellectual contributions, many of which are often overlooked as there are too many to list. Examples include search engines, programming languages, machine learning frameworks, programming libraries, software development tools, computational hardware, operating systems, computing forums, research archives, and scholarly papers. To help developers with limited familiarity, useful resources for deep learning in electron microscopy are discussed in a review paper covered by ch. 1 of my thesis. For brevity, these acknowledgments will focus on personal contributions to my development as a researcher.

- Thanks go to Jeremy Sloan and Richard Beanland for supervision, internal peer review, and co-authorship.
- Thanks go to my Feedback Supervisors, Emma MacPherson and Jon Duffy, for comments needed to partially fulfil requirements of Doctoral Skills Modules (DSMs).
- I am grateful to Marin Alexe and Dong Jik Kim for supervising me during a summer project where I programmed various components of atomic force microscopes. It was when I first realized that I want to be a programmer. Before then, I only thought of programming as something that I did in my spare time.
- I am grateful to James Lloyd-Hughes for supervising me during a summer project where I automated Fourier analysis of ultrafast optical spectroscopy signals.
- I am grateful to my family for their love and support.

As a special note, I first taught myself machine learning by working through Mathematica documentation, implementing every machine learning example that I could find. The practice made use of spare time during a two-week course at the start of my Doctor of Philosophy (PhD) studentship, which was needed to partially fulfil requirements of the Midlands Physics Alliance Graduate School (MPAGS).

My Head of Department is David Leadley. My Director of Graduate Studies was Matthew Turner, then James Lloyd-Hughes after Matthew Turner retired.

I acknowledge funding from Engineering and Physical Sciences Research Council (EPSRC) grant EP/N035437/1 and EPSRC Studentship 1917382.

Declarations

This thesis is submitted to the University of Warwick in support of my application for the degree of Doctor of Philosophy. It has been composed by myself and has not been submitted in any previous application for any degree.

Parts of this thesis have been published by the author:
The following publications [1–8] are part of my thesis.

J. M. Ede. Review: Deep Learning in Electron Microscopy. *arXiv preprint arXiv:2009.08328 (accepted by Machine Learning: Science and Technology –* `https://doi.org/10.1088/2632-2153/abd614`*)*, 2020

J. M. Ede. Warwick Electron Microscopy Datasets. *Machine Learning: Science and Technology*, 1(4): 045003, 2020

J. M. Ede and R. Beanland. Adaptive Learning Rate Clipping Stabilizes Learning. *Machine Learning: Science and Technology*, 1:015011, 2020

J. M. Ede and R. Beanland. Partial Scanning transmission Electron Microscopy with Deep Learning. *Scientific Reports*, 10(1):1–10, 2020

J. M. Ede. Adaptive Partial Scanning Transmission Electron Microscopy with Reinforcement Learning. *arXiv preprint arXiv:2004.02786 (under review by Machine Learning: Science and Technology)*, 2020

J. M. Ede and R. Beanland. Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder. *Ultramicroscopy*, 202:18–25, 2019

J. M. Ede, J. J. P. Peters, J. Sloan, and R. Beanland. Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning. *arXiv preprint arXiv:2001.10938 (under review by Ultramicroscopy)*, 2020

J. M. Ede. Resume of Jeffrey Mark Ede. Zenodo, Online: `https://doi.org/10.5281/zenodo.4429077`, 2021

The following publications [9–12] are part of my thesis. However, they are appendices.

J. M. Ede. Supplementary Information: Warwick Electron Microscopy Datasets. Zenodo, Online: `https://doi.org/10.5281/zenodo.3899740`, 2020

J. M. Ede. Supplementary Information: Partial Scanning Transmission Electron Microscopy with Deep Learning. Online: `https://static-content.springer.com/esm/art%3A10.1038%2Fs41598-020-65261-0/MediaObjects/41598_2020_65261_MOESM1_ESM.pdf`, 2020

J. M. Ede. Supplementary Information: Adaptive Partial Scanning Transmission Electron Microscopy with Reinforcement Learning. Zenodo, Online: `https://doi.org/10.5281/zenodo.4384708`, 2020

J. M. Ede, J. J. P. Peters, J. Sloan, and R. Beanland. Supplementary Information: Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning. Zenodo, Online: `https://doi.org/10.5281/zenodo.4277357`, 2020

The following publications [13–25] are not part of my thesis. However, they are auxiliary to publications that are part of my thesis.

J. M. Ede. Warwick Electron Microscopy Datasets. *arXiv preprint arXiv:2003.01113*, 2020

J. M. Ede. Source Code for Warwick Electron Microscopy Datasets. Online: `https://github.com/Jeffrey-Ede/datasets`, 2020

J. M. Ede. Warwick Electron Microscopy Datasets Archive. Online: `https://github.com/Jeffrey-Ede/datasets/wiki`, 2020

J. M. Ede and R. Beanland. Adaptive Learning Rate Clipping Stabilizes Learning. *arXiv preprint arXiv:1906.09060*, 2019

J. M. Ede. Source Code for Adaptive Learning Rate Clipping Stabilizes Learning. Online: `https://github.com/Jeffrey-Ede/ALRC`, 2020

J. M. Ede and R. Beanland. Partial Scanning Transmission Electron Microscopy with Deep Learning. *arXiv preprint arXiv:1910.10467*, 2020

J. M. Ede. Deep Learning Supersampled Scanning Transmission Electron Microscopy. *arXiv preprint arXiv:1910.10467*, 2019

J. M. Ede. Source Code for Partial Scanning Transmission Electron Microscopy. Online: `https://github.com/Jeffrey-Ede/partial-STEM`, 2019

J. M. Ede. Source Code for Deep Learning Supersampled Scanning Transmission Electron Microscopy. Online: `https://github.com/Jeffrey-Ede/DLSS-STEM`, 2019

J. M. Ede. Source Code for Adaptive Partial Scanning Transmission Electron Microscopy with Reinforcement Learning. Online: `https://github.com/Jeffrey-Ede/adaptive-scans`, 2020

J. M. Ede. Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder. *arXiv preprint arXiv:1807.11234*, 2018

J. M. Ede. Source Code for Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder. Online: `https://github.com/Jeffrey-Ede/Electron-Micrograph-Denoiser`, 2019

J. M. Ede. Source Code for Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning. Online: `https://github.com/Jeffrey-Ede/one-shot`, 2019

The following publications[26–32] are not part of my thesis. However, they are referenced by my thesis, or are referenced by or associated with publications that are part of my thesis.

J. M. Ede. Progress Reports of Jeffrey Mark Ede: 0.5 Year Progress Report. Zenodo, Online: `https://doi.org/10.5281/zenodo.4094750`, 2020

J. M. Ede. Source Code for Beanland Atlas. Online: `https://github.com/Jeffrey-Ede/Beanland-Atlas`, 2018

J. M. Ede. Thesis Word Counting. Zenodo, Online: `https://doi.org/10.5281/zenodo.4321429`, 2020

J. M. Ede. Posters and Presentations. Zenodo, Online: `https://doi.org/10.5281/zenodo.4041574`, 2020

J. M. Ede. Autoencoders, Kernels, and Multilayer Perceptrons for Electron Micrograph Restoration and Compression. *arXiv preprint arXiv:1808.09916*, 2018

J. M. Ede. Source Code for Autoencoders, Kernels, and Multilayer Perceptrons for Electron Micrograph Restoration and Compression. Online: https://github.com/Jeffrey-Ede/Denoising-Kernels-MLPs-Autoencoders, 2018

J. M. Ede. Source Code for Simple Webserver. Online: https://github.com/Jeffrey-Ede/simple-webserver, 2019

All publications were produced during my period of study for the degree of Doctor of Philosophy in Physics at the University of Warwick.

The work presented (including data generated and data analysis) was carried out by the author except in the cases outlined below:

Chapter 1 Review: Deep Learning in Electron Microscopy

Jeremy Sloan and Martin Lotz internally reviewed my paper after I published it in the arXiv.

Chapter 2 Warwick Electron Microscopy Datasets

Richard Beanland internally reviewed my paper before it was published in the arXiv. Further, Jonathan Peters discussed categories used to showcase typical electron micrographs for readers with limited familiarity. At first, our datasets were openly accessible from my Google Cloud Storage. However, Richard Beanland contacted University of Warwick Information Technology Services to arrange for our datasets to also be openly accessible from University of Warwick data servers. Chris Parkin allocated server resources, advised me on data transfer, and handled administrative issues. In addition, datasets are openly accessible from Zenodo and my Google Drive.

Simulated datasets were created with clTEM multislice simulation software developed by a previous EM group PhD student, Mark Dyson, and maintained by a previous EM group postdoctoral researcher, Jonathan Peters. Jonathan Peters advised me on processing data that I had curated from the Crystallography Open Database (COD) so that it could be input into clTEM simulations. Further, Jonathan Peters and I jointly prepared a script to automate multislice simulations. Finally, Jonathan Peters computed a third of our simulations on his graphical processing units (GPUs).

Experimental datasets were curated from University of Warwick Electron Microscopy (EM) Research Technology Platform (RTP) dataservers, and contain images collected by dozens of scientists working on hundreds of projects. Data was curated and published with permission of the Director of the EM RTP, Richard Beanland. In addition, data curation and publication were reviewed and approved by Research Data Officers, Yvonne Budden and Heather Lawler. I was introduced to the EM dataservers by Richard Beanland and Jonathan Peters, and my read and write access to the EM dataservers was set up by an EM RTP technician, Steve York.

Chapter 3 Adaptive Learning Rate Clipping Stabilizes Learning

Richard Beanland internally reviewed my paper after it was published in the arXiv. Martin Lotz later recommend the journal that I published it in. In addition, a Scholarly Communications Manager, Julie Robinson, advised me on finding publication venues and open access funding. I also discussed publication venues with editors of Machine Learning, Melissa Fearon and Peter Flach, and my Centre for Scientific Computing Director, David Quigley.

Chapter 4 Partial Scanning Transmission Electron Microscopy with Deep Learning

Richard Beanland internally reviewed an initial draft of my paper on partial scanning transmission electron microscopy (STEM). After I published our paper in the arXiv, Richard Beanland contributed most of the content in the first two paragraphs in the introduction of the journal paper. In addition, Richard Beanland and I both copyedited our paper.

Richard Beanland internally reviewed a paper on uniformly spaced scans after I published it in the arXiv. The uniformly spaced scans paper includes some experiments that we later combined into our partial STEM paper. Further, my experiments followed a preliminary investigation into compressed sensing with fixed randomly spaced masks, which Richard Beanland internally reviewed.

Chapter 5 Adaptive Partial Scanning Transmission Electron Microscopy with Reinforcement Learning

Jasmine Clayton, Abdul Mohammed, and Jeremy Sloan internally reviewed my paper after I published it in the arXiv.

Chapter 6 Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder

After I published my paper in the arXiv, Richard Beanland internally reviewed it and advised that we publish it in a journal. In addition, Richard Beanland and I both copyedited our paper.

Chapter 7 Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning

Jeremy Sloan internally reviewed an initial draft of our paper. Afterwards, Jeremy Sloan contributed all crystal structure diagrams in our paper. The University of Warwick X-Ray Facility Manager, David Walker, suggested materials to showcase with their crystal structures, and a University of Warwick Research Fellow, Jessica Marshall, internally reviewed a figure showing exit wavefunction reconstructions (EWRs) with the crystal structures.

Richard Beanland contacted a professor at Humboldt University of Berlin, Christoph Koch, to ask for a DigitalMicrograph plugin, which I used to collect experimental focal series. Further, Richard Beanland helped me get started with focal series measurements, and internally reviewed some of my first focal series. In addition, Richard Beanland internally reviewed our paper.

Jonathan Peters drafted initial text about clTEM multislice simulations for a section of our paper on "Exit Wavefunction Datasets". In addition, Jonathan Peters internally reviewed our paper.

This thesis conforms to regulations governing the examination of higher degrees by research:
The following regulations[33,34] were used during preparation of this thesis.

Guide to Examinations for Higher Degrees by Research. University of Warwick Doctoral College, Online: `https://warwick.ac.uk/services/dc/pgrassessments/gtehdr`, 2020

Regulation 38: Research Degrees. University of Warwick Calendar, Online: `https://warwick.ac.uk/services/gov/calendar/section2/regulations/reg38pgr`, 2020

The following guidance[35] was helpful during preparation of this thesis.

Thesis Writing and Submission. University of Warwick Department of Physics, Online: `https://warwick.ac.uk/fac/sci/physics/current/postgraduate/regs/thesis`, 2020

The following thesis template[36] was helpful during preparation of this thesis.

A Warwick Thesis Template. University of Warwick Department of Physics, Online: https://warwick.ac.uk/fac/sci/physics/staff/academic/mhadley/wthesis, 2020

Thesis structure and content was discussed with my previous Director of Graduate Studies, Matthew Turner, and my current Director of Graduate Studies, James Lloyd-Hughes, after Matthew Turner retired. My thesis was also discussed with my both my previous PhD supervisor, Richard Beanland, and my current PhD supervisor, Jeremy Sloan. My formal thesis plan was then reviewed and approved by both Jeremy Sloan and my feedback supervisor, Emma MacPherson. Finally, my complete thesis was internally reviewed by both Jeremy Sloan and Jasmine Clayton.

Permission is granted by the Chair of the Board of Graduate Studies, Colin Sparrow, for my thesis appendices to exceed length requirements usually set by the University of Warwick. This is in the understanding that my thesis appendices are not usually crucial to the understanding or examination of my thesis.

Research Training

This thesis presents a substantial original investigation of deep learning in electron microscopy. The only researcher in my research group or building with machine learning expertise was myself. This meant that I led the design, implementation, evaluation, and publication of experiments covered by my thesis. Where experiments were collaborative, I both proposed and led the collaboration.

Abstract

Following decades of exponential increases in computational capability and widespread data availability, deep learning is readily enabling new science and technology. This thesis starts with a review of deep learning in electron microscopy, which offers a practical perspective aimed at developers with limited familiarity. To help electron microscopists get started with started with deep learning, large new electron microscopy datasets are introduced for machine learning. Further, new approaches to variational autoencoding are introduced to embed datasets in low-dimensional latent spaces, which are used as the basis of electron microscopy search engines. Encodings are also used to investigate electron microscopy data visualization by t-distributed stochastic neighbour embedding. Neural networks that process large electron microscopy images may need to be trained with small batch sizes to fit them into computer memory. Consequently, adaptive learning rate clipping is introduced to prevent learning being destabilized by loss spikes associated with small batch sizes.

This thesis presents three applications of deep learning to electron microscopy. Firstly, electron beam exposure can damage some specimens, so generative adversarial networks were developed to complete realistic images from sparse spiral, gridlike, and uniformly spaced scans. Further, recurrent neural networks were trained by reinforcement learning to dynamically adapt sparse scans to specimens. Sparse scans can decrease electron beam exposure and scan time by 10-100× with minimal information loss. Secondly, a large encoder-decoder was developed to improve transmission electron micrograph signal-to-noise. Thirdly, conditional generative adversarial networks were developed to recover exit wavefunction phases from single images. Phase recovery with deep learning overcomes existing limitations as it is suitable for live applications and does not require microscope modification. To encourage further investigation, scientific publications and their source files, source code, pretrained models, datasets, and other research outputs covered by this thesis are openly accessible.

Preface

This thesis covers a subset of my scientific papers on advances in electron microscopy with deep learning. The papers were prepared while I was a PhD student at the University of Warwick in support of my application for the degree of PhD in Physics. This thesis reflects on my research, unifies covered publications, and discusses future research directions. My papers are available as part of chapters of this thesis, or from their original publication venues with hypertext and other enhancements. This preface covers my initial motivation to investigate deep learning in electron microscopy, structure and content of my thesis, and relationships between included publications. Traditionally, physics PhD theses submitted to the University of Warwick are formatted for physical printing and binding. However, I have also formatted a copy of my thesis for online dissemination to improve readability[37].

I Initial Motivation

When I started my PhD in October 2017, we were unsure if or how machine learning could be applied to electron microscopy. My PhD was funded by EPSRC Studentship 1917382[38] titled "Application of Novel Computing and Data Analysis Methods in Electron Microscopy", which is associated with EPSRC grant EP/N035437/1[39] titled "ADEPT – Advanced Devices by ElectroPlaTing". As part of the grant, our initial plan was for me to spend a couple of days per week using electron microscopes to analyse specimens sent to the University of Warwick from the University of Southampton, and to invest remaining time developing new computational techniques to help with analysis. However, an additional scientist was not needed to analyse specimens, so it was difficult for me to get electron microscopy training. While waiting for training, I was tasked with automating analysis of digital large angle convergent beam electron diffraction[40] (D-LACBED) patterns. However, we did not have a compelling use case for my D-LACBED software[26,41]. Further, a more senior PhD student at the University of Warwick, Alexander Hubert, was already investigating convergent beam electron diffraction[40,42] (CBED).

My first machine learning research began five months after I started my PhD. Without a clear research direction or specimens to study, I decided to develop artificial neural networks (ANNs) to generate artwork. My dubious plan was to create image processing pipelines for the artwork, which I would replace with electron micrographs when I got specimens to study. However, after investigating artwork generation with randomly initialized multilayer perceptrons[43,44], then by style transfer[45,46], and then by fast style transfer[47], there were still no specimens for me to study. Subsequently, I was inspired by NVIDIA's research on semantic segmentation[48] to investigate semantic segmentation with DeepLabv3+[49]. However, I decided that it was unrealistic for me to label a large new electron microscopy dataset for semantic segmentation by myself. Fortunately, I had read about using deep neural networks (DNNs) to reduce image compression artefacts[50], so I wondered if a similar approach based on DeepLabv3+ could improve electron micrograph signal-to-noise. Encouragingly, it would not require time-consuming image labelling. Following a successful investigation into improving signal-to-noise, my first scientific paper[6] (ch. 6) was submitted a few months later, and my experience with deep learning enabled subsequent investigations.

II Thesis Structure

An overview of the first seven chapters in this thesis is presented in fig. 1. The first chapter is introductory and covers a review of deep learning in electron microscopy, which offers a practical perspective aimed at developers with limited familiarity. The next two chapters are ancillary and cover new datasets and an optimization algorithm used in later chapters. The final four chapters before conclusions cover investigations of deep learning in electron microscopy. Each of the first seven chapter covers a combination of journal papers, preprints, and ancillary outputs

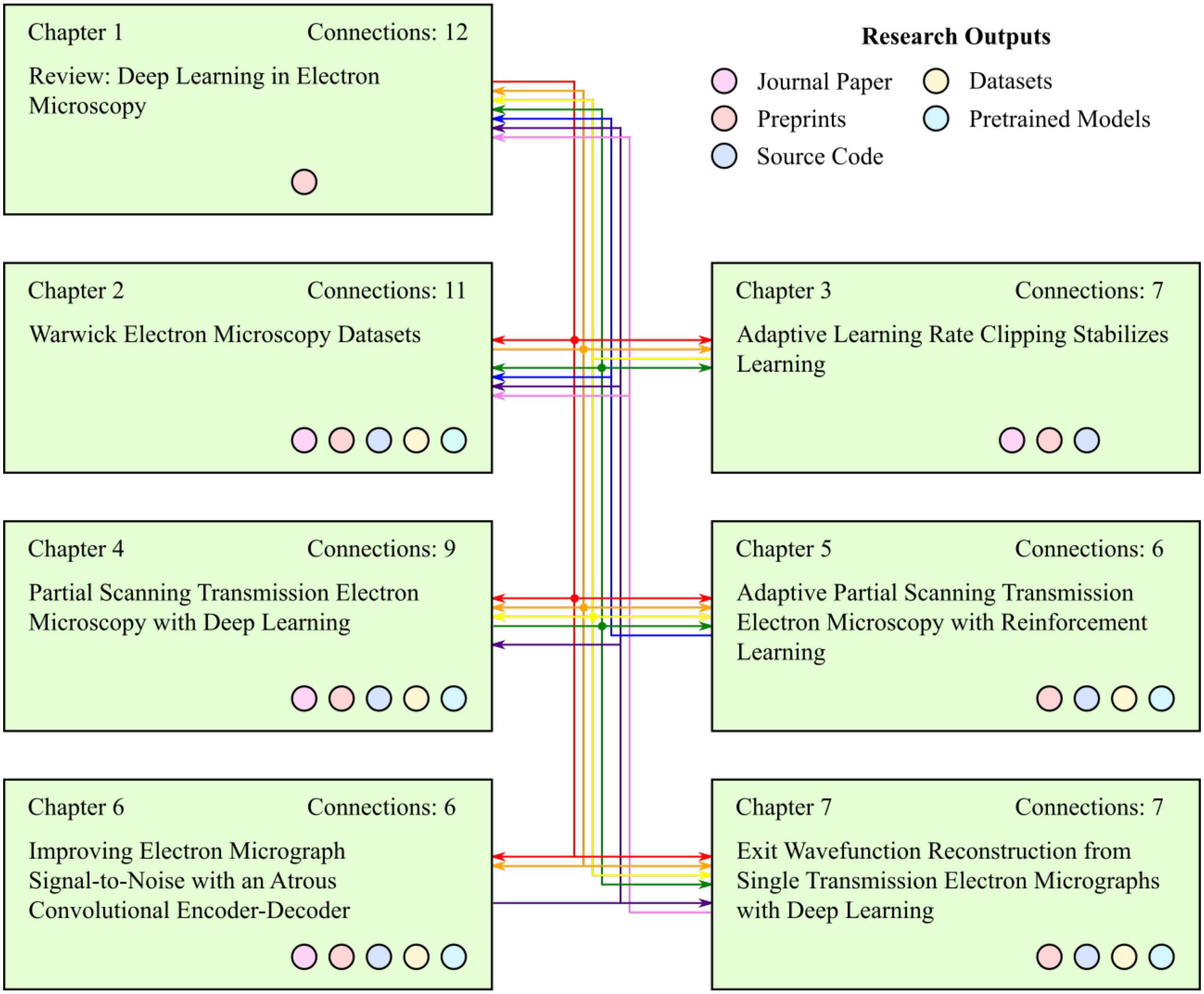

Chapter 1
Connections: 12
Review: Deep Learning in Electron Microscopy
Research Outputs
Journal Paper
Datasets
Preprints
Pretrained Models
Source Code
Chapter 2
Connections: 11
Warwick Electron Microscopy Datasets
Chapter 3
Connections: 7
Adaptive Learning Rate Clipping Stabilizes Learning
Chapter 4
Connections: 9
Partial Scanning Transmission Electron Microscopy with Deep Learning
Chapter 5
Connections: 6
Adaptive Partial Scanning Transmission Electron Microscopy with Reinforcement Learning
Chapter 6
Connections: 6
Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder
Chapter 7
Connections: 7
Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning

Figure 1: Connections between publications covered by chapters of this thesis. An arrow from chapter x to chapter y indicates that results covered by chapter y depend on results covered by chapter x. Labels indicate types of research outputs associated with each chapter, and total connections to and from chapters.

such as source code, datasets, and pretrained models, and supplementary information.

At the University of Warwick, physics PhD theses that cover publications[51,52] are unusual. Instead, most theses are scientific monographs. However, declining impact of monographic theses is long-established[53], and I felt that scientific publishing would push me to produce higher-quality research. Moreover, I think that publishing is an essential part of scientific investigation, and external peer reviews[54–58] often helped me to improve my papers. Open access to PhD theses increases visibility[59,60] and enables their use as data mining resources[60,61], so digital copies of physics PhD theses are archived by the University of Warwick[62]. However, archived theses are usually formatted for physical printing and binding. To improve readability, I have also formatted a copy of my thesis for online dissemination[37], which is published in the arXiv[63,64] with its Latex[65–67] source files.

All my papers were first published as arXiv preprints under Creative Commons Attribution 4.0[68] licenses, then submitted to journals. As discussed in my review[1] (ch. 1), advantages of preprint archives[69–71] include ensuring that research is openly accessible[72], increasing discovery and citations[73–77], inviting timely scientific discussion, and raising awareness to reduce unnecessary duplication of research. Empirically, there are no significant textual differences between arXiv preprints and corresponding journal papers[78]. However, journal papers appear to be slightly higher quality than biomedical preprints[78,79], suggesting that formatting and copyediting practices vary between scientific disciplines. Overall, I think that a lack of differences between journal papers and preprints may be a result of publishers separating language editing into premium services[80–83], rather than including extensive language editing in their usual publication processes. Increasing textual quality is correlated with increasing likelihood that an article will be published[84]. However, most authors appear to be performing copyediting themselves to avoid extra fees.

A secondary benefit of posting arXiv preprints is that their metadata, an article in portable document format[85,86] (PDF), and any Latex source files are openly accessible. This makes arXiv files easy to reuse, especially if they are published under permissive licenses[87]. For example, open accessibility enabled arXiv files to be curated into a large dataset[88] that was used to predict future research trends[89]. Further, although there is no requirement for preprints to peer reviewed, preprints can enable early access to papers that have been peer reviewed. As a case in point, all preprints covered by my thesis have been peer reviewed. Further, the arXiv implicitly supports peer review by providing contact details of authors, and I have both given and received feedback about arXiv papers. In addition, open peer review platforms[90], such as OpenReview[91,92], can be used to explicitly seek peer review. There is also interest in integrating peer review with the arXiv, so a conceptual peer review model has been proposed[93].

Description	Words in Text	Words in Figures	Words in Algorithms	Total Words
Review paper in chapter 1	15156	2680	74	17910
Ancillary paper in chapter 2	4243	1360	0	5603
Ancillary paper in chapter 3	2448	680	344	3472
Paper in chapter 4	3864	1300	0	5164
Paper in chapter 5	3399	900	440	4739
Paper in chapter 6	2933	1100	0	4033
Paper in chapter 7	4396	1240	0	5636
Remainder of the thesis	7950	280	0	8230
Complete thesis	**44389**	**9540**	**858**	**54787**

Table 1: Word counts for papers included in thesis chapters, the remainder of the thesis, and the complete thesis.

This thesis covers a selection of my interconnected scientific papers. Word counts for my papers and covering text are tabulated in table 1. Figures are included in word counts by adding products of nominal word densities and figure areas. However, acknowledgements, references, tables, supplementary information, and similar contents are not included as they do not count towards my thesis length limit of 70000 words. For details, notes on my word counting procedure are openly accessible[28]. Associated research outputs, such as source code and datasets, are not directly included in my thesis due to format restrictions. Nevertheless, my source code is openly accessible from GitHub[94], and archived releases of my source code are openly accessible from Zenodo[95]. In addition, links to openly accessible pretrained models are provided in my source code documentation. Finally, links to openly accessible datasets are in my papers, source code documentation, and datasets paper[2] (ch. 2).

III Connections

Connections between publications covered by my thesis are shown in fig. 1. The most connected chapter covers my review paper[1] (ch. 1). All my papers are connected to my review paper as literature reviews informed their introductions, methodologies, and discussions. My review paper also discusses and builds upon the results of my earlier publications. For example, images published in my earlier papers are reused in my review paper to showcase applications of deep learning in electron microscopy. In addition, my review paper covers Warwick Electron Microscopy Datasets[2] (WEMD, ch. 2), adaptive learning rate clipping[3] (ALRC, ch. 3), sparse scans for compressed sensing in STEM[4] (ch. 4), improving electron microscope signal-to-noise[6] (ch. 6), and EWR[7] (ch. 7).

Finally, compressed sensing with dynamic scan paths that adapt to specimens[5] (ch. 5) motivated my review paper sections on recurrent neural networks (RNNs) and reinforcement learning (RL).

The second most connected chapter, ch. 2, is ancillary and covers WEMD[2], which include large new datasets of experimental transmission electron microscopy (TEM) images, experimental STEM images, and simulated exit wavefunctions. The TEM images were curated to train an ANN to improve signal-to-noise[6] (ch. 6) and motivated the proposition of a new approach to EWR[7] (ch. 7). The STEM images were curated to train ANNs for compressed sensing[4] (ch. 4). Training our ANNs with full-size images was impractical with our limited computational resources, so I created dataset variants containing 512×512 crops from full-size images for both the TEM and STEM datasets. However, 512×512 STEM crops were too large to efficiently train RNNs to adapt scan paths[5] (ch. 5), so I also created 96×96 variants of datasets for rapid initial development. Finally, datasets of exit wavefunctions were simulated as part of our initial investigation into EWR from single TEM images with deep learning[7] (ch. 7).

The other ancillary chapter, ch. 3, covers ALRC[3], which was originally published as an appendix in the first version of our partial STEM preprint[18] (ch. 4). The algorithm was developed to stabilize learning of ANNs being developed for partial STEM, which were destabilized by loss spikes when training with a batch size of 1. My aim was to make experiments[10] easier to compare by preventing learning destabilized by large loss spikes from complicating comparisons. However, ALRC was so effective that I continued to investigate it, increasing the size of the partial STEM appendix. Eventually, the appendix became so large that I decided to turn it into a short paper. To stabilize training with small batch sizes, ALRC was also applied to ANN training for uniformly spaced scans[4,19] (ch. 4). In addition, ALRC inspired adaptive loss clipping to stabilize RNN training for adaptive scans[5] (ch. 5). Finally, I investigated applying ALRC to ANN training for EWR[7] (ch. 7). However, ALRC did not improve EWR as training with a batch size of 32 was not destabilized by loss spikes.

My experiments with compressed sensing showed that ANN performance varies for different scan paths[4] (ch. 4). This motivated the investigation of scan shapes that adapt to specimens as they are scanned[5] (ch. 5). I had found that ANNs for TEM denoising[6] (ch. 6) and uniformly spaced sparse scan completion[19] exhibit significant structured systematic error variation, where errors are higher near output edges. Subsequently, I investigated average partial STEM output errors and found that errors increase with increasing distance from scan paths[4] (ch. 4). In part, structured systematic error variation in partial STEM[4] (ch. 4) motivated my investigation of adaptive scans[5] (ch. 5) as I reasoned that being able to more closely scan regions where errors would otherwise be highest could decrease mean errors.

Most of my publications are connected by their source code as it was partially reused in successive experiments. Source code includes scripts to develop ANNs, plot graphs, create images for papers, and typeset with Latex. Following my publication chronology, I partially reused source code created to improve signal-to-noise[6] (ch. 6) for partial STEM[4] (ch. 4). My partial STEM source code was then partially reused for my other investigations. Many of my publications are also connected because datasets curated for my first investigations were reused in my later investigations. For example, improving signal-to-noise[6] (ch. 6) is connected to EWR[7] (ch. 7) as the availability of my large dataset of TEM images prompted the proposition of, and may enable, a new approach to EWR. Similarly, partial STEM[4] (ch. 4) is connected to adaptive scans[5] (ch. 5) as my large dataset of STEM images was used to derive smaller datasets used to rapidly develop adaptive scan systems.

Chapter 1

Review: Deep Learning in Electron Microscopy

1.1 Scientific Paper

This chapter covers the following paper[1].

> J. M. Ede. Review: Deep Learning in Electron Microscopy. *arXiv preprint arXiv:2009.08328 (accepted by Machine Learning: Science and Technology –* `https://doi.org/10.1088/2632-2153/abd614`*)*, 2020

Review: Deep Learning in Electron Microscopy

Jeffrey M. Ede[1,*]

[1]University of Warwick, Department of Physics, Coventry, CV4 7AL, UK
[*]j.m.ede@warwick.ac.uk

ABSTRACT

Deep learning is transforming most areas of science and technology, including electron microscopy. This review paper offers a practical perspective aimed at developers with limited familiarity. For context, we review popular applications of deep learning in electron microscopy. Following, we discuss hardware and software needed to get started with deep learning and interface with electron microscopes. We then review neural network components, popular architectures, and their optimization. Finally, we discuss future directions of deep learning in electron microscopy.

Keywords: deep learning, electron microscopy, review.

1 Introduction

Following decades of exponential increases in computational capability[1] and widespread data availability[2,3], scientists can routinely develop artificial neural networks[4–11] (ANNs) to enable new science and technology[12–17]. The resulting deep learning revolution[18,19] has enabled superhuman performance in image classification[20–23], games[24–29], medical analysis[30,31], relational reasoning[32], speech recognition[33,34] and many other applications[35,36]. This introduction focuses on deep learning in electron microscopy and is aimed at developers with limited familiarity. For context, we therefore review popular applications of deep learning in electron microscopy. We then review resources available to support researchers and outline electron microscopy. Finally, we review popular ANN architectures and their optimization, or "training", and discuss future trends in artificial intelligence (AI) for electron microscopy.

Deep learning is motivated by universal approximator theorems[37–45], which state that sufficiently deep and wide[37,40,46] ANNs can approximate functions to arbitrary accuracy. It follows that ANNs can always match or surpass the performance of methods crafted by humans. In practice, deep neural networks (DNNs) reliably[47] learn to express[48–51] generalizable[52–59] models without a prior understanding of physics. As a result, deep learning is freeing physicists from a need to devise equations to model complicated phenomena[13,14,16,60,61]. Many modern ANNs have millions of parameters, so inference often takes tens of milliseconds on graphical processing units (GPUs) or other hardware accelerators[62]. It is therefore unusual to develop ANNs to approximate computationally efficient methods with exact solutions, such as the fast Fourier transform[63–65] (FFT). However, ANNs are able to leverage an understanding of physics to accelerate time-consuming or iterative calculations[66–69], improve accuracy of methods[30,31,70], and find solutions that are otherwise intractable[24,71].

1.1 Improving Signal-to-Noise

A popular application of deep learning is to improve signal-to-noise[74,75]. For example, of medical electrical[76,77], medical image[78–80], optical microscopy[81–84], and speech[85–88] signals. There are many traditional denoising algorithms that are not based on deep learning[89–91], including linear[92,93] and non-linear[94–102] spatial domain filters, Wiener[103–105] filters, non-linear[106–111] wavelet domain filters, curvelet transforms[112,113], contourlet transforms[114,115], hybrid algorithms[116–122] that operate in both spatial and transformed domains, and dictionary-based learning[123–127]. However, traditional denoising algorithms are limited by features (often laboriously) crafted by humans and cannot exploit domain-specific context. In perspective, they leverage an ever-increasingly accurate representation of physics to denoise signals. However, traditional algorithms are limited by the difficulty of programmatically describing a complicated reality. As a case in point, an ANN was able to outperform decades of advances in traditional denoising algorithms after training on two GPUs for a week[70].

Definitions of electron microscope noise can include statistical noise[128–135], aberrations[136], scan distortions[137–140], specimen drift[141], and electron beam damage[142]. Statistical noise is often minimized by either increasing electron dose or applying traditional denoising algorithms[143,144]. There are a variety of denoising algorithms developed for electron microscopy, including algorithms based on block matching[145], contourlet transforms[114,115], energy minimization[146], fast patch reorderings[147], Gaussian kernel density estimation[148], Kronecker envelope principal component analysis[149] (PCA), non-local means and Zernike moments[150], singular value thresholding[151], wavelets[152], and other approaches[141,153–156]. Noise that is not statistical is

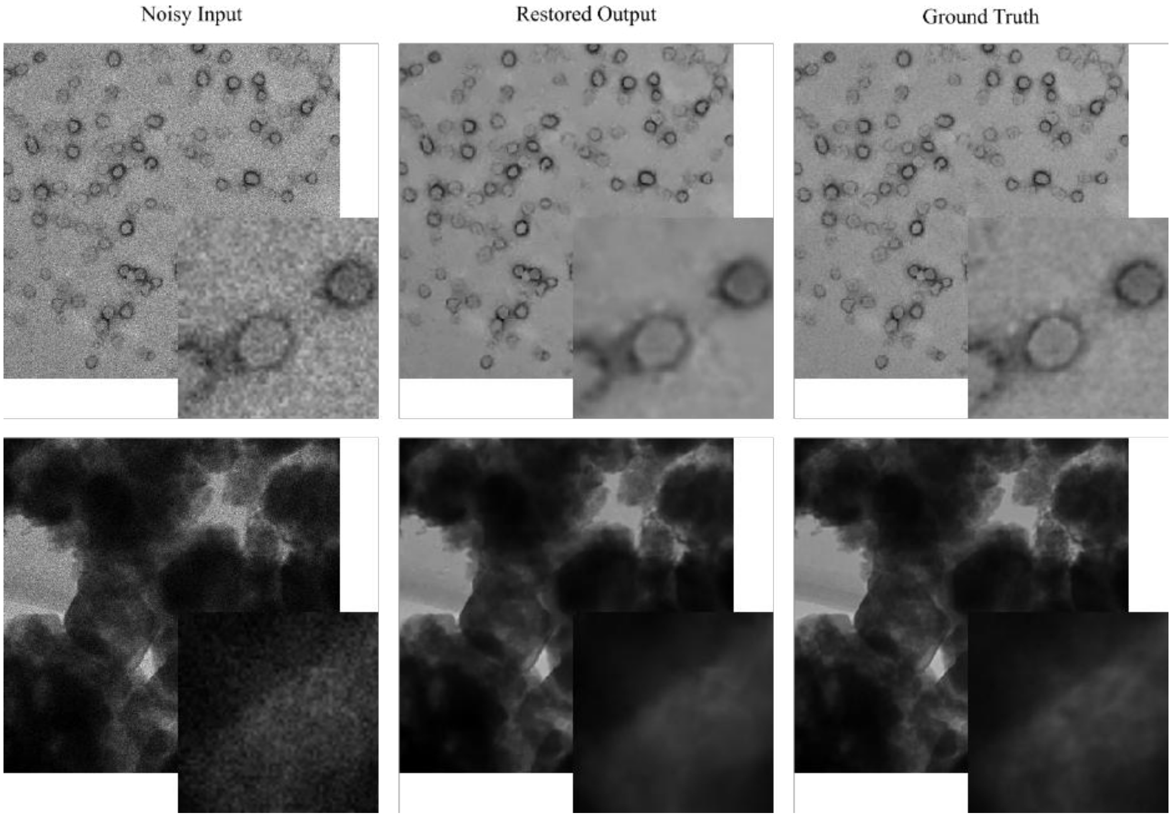


Figure 1. Example applications of a noise-removal DNN to instances of Poisson noise applied to 512×512 crops from TEM images. Enlarged 64×64 regions from the top left of each crop are shown to ease comparison. This figure is adapted from our earlier work[72] under a Creative Commons Attribution 4.0[73] license.

often minimized by hardware. For example, by using aberration correctors[136, 157–159], choosing scanning transmission electron microscopy (STEM) scan shapes and speeds that minimize distortions[138], and using stable sample holders to reduce drift[160]. Beam damage can also be reduced by using minimal electron voltage and electron dose[161–163], or dose-fractionation across multiple frames in multi-pass transmission electron microscopy[164–166] (TEM) or STEM[167].

Deep learning is being applied to improve signal-to-noise for a variety of applications[168–176]. Most approaches in electron microscopy involve training ANNs to either map low-quality experimental[177], artificially deteriorated[70, 178] or synthetic[179–182] inputs to paired high-quality experimental measurements. For example, applications of a DNN trained with artificially deteriorated TEM images are shown in figure 1. However, ANNs have also been trained with unpaired datasets of low-quality and high-quality electron micrographs[183], or pairs of low-quality electron micrographs[184, 185]. Another approach is Noise2Void[168], ANNs are trained from single noisy images. However, Noise2Void removes information by masking noisy input pixels corresponding to target output pixels. So far, most ANNs that improve electron microscope signal-to-noise have been trained to decrease statistical noise[70, 177, 179–181, 181–184, 186] as other approaches have been developed to correct electron microscope scan distortions[187, 188] and specimen drift[141, 188, 189]. However, we anticipate that ANNs will be developed to correct a variety of electron microscopy noise as ANNs have been developed for aberration correction of optical microscopy[190–195] and photoacoustic[196] signals.

1.2 Compressed Sensing

Compressed sensing[203–207] is the efficient reconstruction of a signal from a subset of measurements. Applications include faster medical imaging[208–210], image compression[211, 212], increasing image resolution[213, 214], lower medical radiation exposure[215–217], and low-light vision[218, 219]. In STEM, compressed sensing has enabled electron beam exposure and scan time to be decreased by 10-100× with minimal information loss[201, 202]. Thus, compressed sensing can be essential to investigations where the high current density of electron probes damages specimens[161, 220–226]. Even if the effects of beam damage can be corrected by postprocessing, the damage to specimens is often permanent. Examples of beam-sensitive materials include organic crystals[227], metal-organic frameworks[228], nanotubes[229], and nanoparticle dispersions[230]. In electron microscopy, compressed sensing is

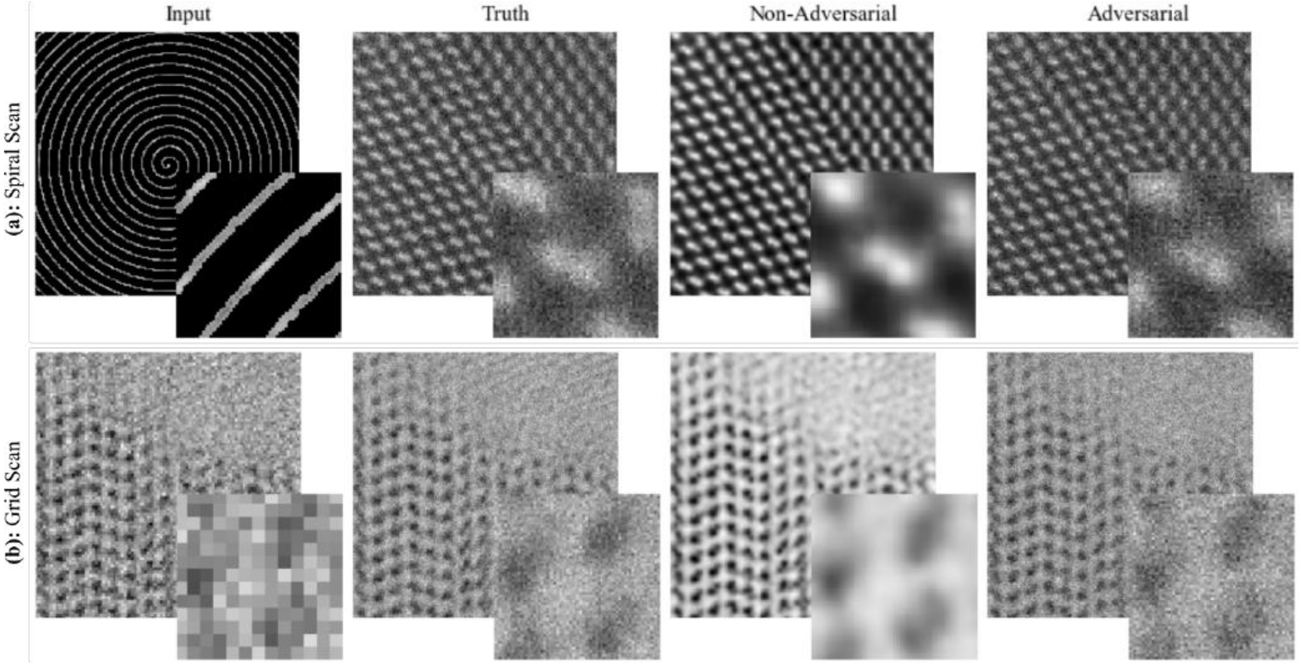


Figure 2. Example applications of DNNs to restore 512×512 STEM images from sparse signals. Training as part of a generative adversarial network[197–200] yields more realistic outputs than training a single DNN with mean squared errors. Enlarged 64×64 regions from the top left of each crop are shown to ease comparison. a) Input is a Gaussian blurred 1/20 coverage spiral[201]. b) Input is a 1/25 coverage grid[202]. This figure is adapted from our earlier works under Creative Commons Attribution 4.0[73] licenses.

especially effective due to high signal redundancy[231]. For example, most electron microscopy images are sampled at 5-10× their Nyquist rates[232] to ease visual inspection, decrease sub-Nyquist aliasing[233], and avoid undersampling.

Perhaps the most popular approach to compressed sensing is upsampling or infilling a uniformly spaced grid of signals[234–236]. Interpolation methods include Lancsoz[234], nearest neighbour[237], polynomial interpolation[238], Wiener[239] and other resampling methods[240–242]. However, a variety of other strategies to minimize STEM beam damage have also been proposed, including dose fractionation[243] and a variety of sparse data collection methods[244]. Perhaps the most intensively investigated approach to the latter is sampling a random subset of pixels, followed by reconstruction using an inpainting algorithm[244–249]. Random sampling of pixels is nearly optimal for reconstruction by compressed sensing algorithms[250]. However, random sampling exceeds the design parameters of standard electron beam deflection systems, and can only be performed by collecting data slowly[138,251], or with the addition of a fast deflection or blanking system[247,252].

Sparse data collection methods that are more compatible with conventional STEM electron beam deflection systems have also been investigated. For example, maintaining a linear fast scan deflection whilst using a widely-spaced slow scan axis with some small random 'jitter'[245,251]. However, even small jumps in electron beam position can lead to a significant difference between nominal and actual beam positions in a fast scan. Such jumps can be avoided by driving functions with continuous derivatives, such as those for spiral and Lissajous scan paths[138,201,247,253,254]. Sang[138,254] considered a variety of scans including Archimedes and Fermat spirals, and scans with constant angular or linear displacements, by driving electron beam deflectors with a field-programmable gate array[255] (FPGA) based system[138]. Spirals with constant angular velocity place the least demand on electron beam deflectors. However, dwell times, and therefore electron dose, decreases with radius. Conversely, spirals created with constant spatial speeds are prone to systematic image distortions due to lags in deflector responses. In practice, fixed doses are preferable as they simplify visual inspection and limit the dose dependence of STEM noise[129].

Deep learning can leverage an understanding of physics to infill images[256–258]. Example applications include increasing scanning electron microscopy[178,259,260] (SEM), STEM[202,261] and TEM[262] resolution, and infilling continuous sparse scans[201]. Example applications of DNNs to complete sparse spiral and grid scans are shown in figure 2. However, caution should be used when infilling large regions as ANNs may generate artefacts if a signal is unpredictable[201]. A popular alternative to deep learning for infilling large regions is exemplar-based infilling[263–266]. However, exemplar-based infilling often leaves artefacts[267] and is usually limited to leveraging information from single images. Smaller regions are often infilled by fast marching[268], Navier-Stokes infilling[269], or interpolation[238].

1.3 Labelling

Deep learning has been the basis of state-of-the-art classification[270–273] since convolutional neural networks (CNNs) enabled a breakthrough in classification accuracy on ImageNet[71]. Most classifiers are single feedforward neural networks (FNNs) that learn to predict discrete labels. In electron microscopy, applications include classifying image region quality[274,275], material structures[276,277], and image resolution[278]. However, siamese[279–281] and dynamically parameterized[282] networks can more quickly learn to recognise images. Finally, labelling ANNs can learn to predict continuous features, such as mechanical properties[283]. Labelling ANNs are often combined with other methods. For example, ANNs can be used to automatically identify particle locations[186,284–286] to ease subsequent processing.

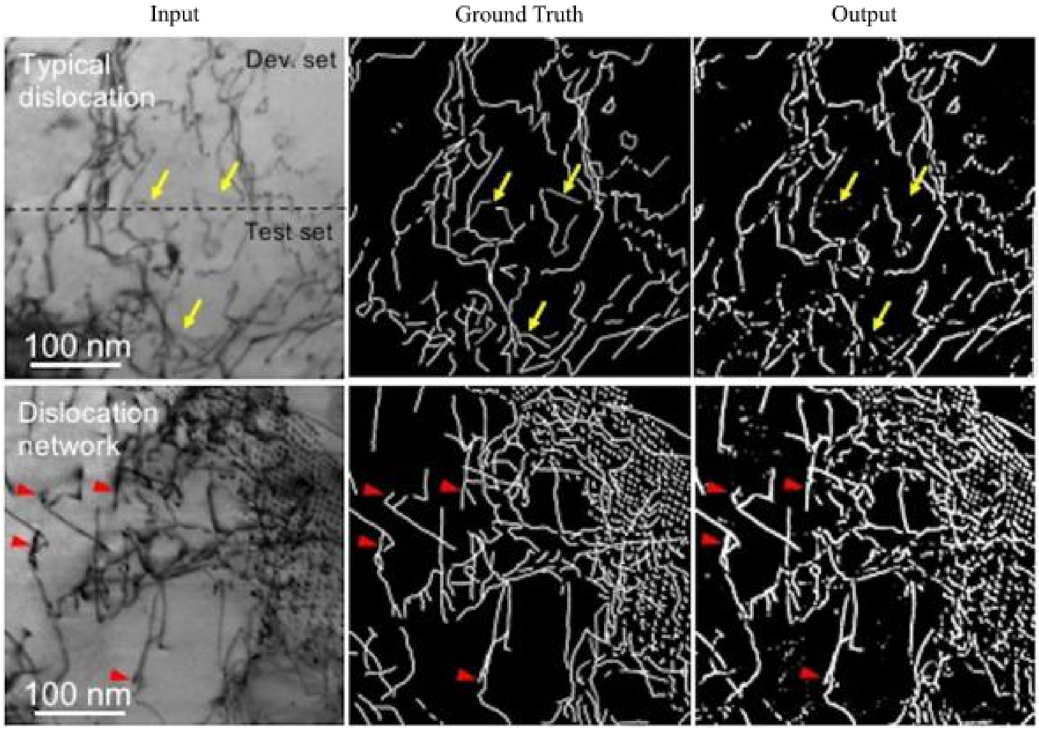


Figure 3. Example applications of a semantic segmentation DNN to STEM images of steel to classify dislocation locations. Yellow arrows mark uncommon dislocation lines with weak contrast, and red arrows indicate that fixed widths used for dislocation lines are sometimes too narrow to cover defects. This figure is adapted with permission[287] under a Creative Commons Attribution 4.0[73] license.

1.4 Semantic Segmentation

Semantic segmentation is the classification of pixels into discrete categories. In electron microscopy, applications include the automatic identification of local features[288,289], such as defects[290,291], dopants[292], material phases[293], material structures[294,295], dynamic surface phenomena[296], and chemical phases in nanoparticles[297]. Early approaches to semantic segmentation used simple rules. However, such methods were not robust to a high variety of data[298]. Subsequently, more adaptive algorithms based on soft-computing[299] and fuzzy algorithms[300] were developed to use geometric shapes as priors. However, these methods were limited by programmed features and struggled to handle the high variety of data.

To improve performance, DNNs have been trained to semantically segment images[301–308]. Semantic segmentation DNNs have been developed for focused ion beam scanning electron microscopy[309–311] (FIB-SEM), SEM[311–314], STEM[287,315], and TEM[286,310,311,316–319]. For example, applications of a DNN to semantic segmentation of STEM images of steel are shown in figure 3. Deep learning based semantic segmentation also has a high variety of applications outside of electron microscopy, including autonomous driving[320–324], dietary monitoring[325,326], magnetic resonance images[327–331], medical images[332–334] such as prenatal ultrasound[335–338], and satellite image translation[339–343]. Most DNNs for semantic segmentation are trained with images segmented by humans. However, human labelling may be too expensive, time-consuming, or inappropriate for sensitive data. Unsupervised semantic segmentation can avoid these difficulties by learning to segment images from an additional dataset of segmented images[344] or image-level labels[345–348]. However, unsupervised semantic segmentation networks are often less accurate than supervised networks.

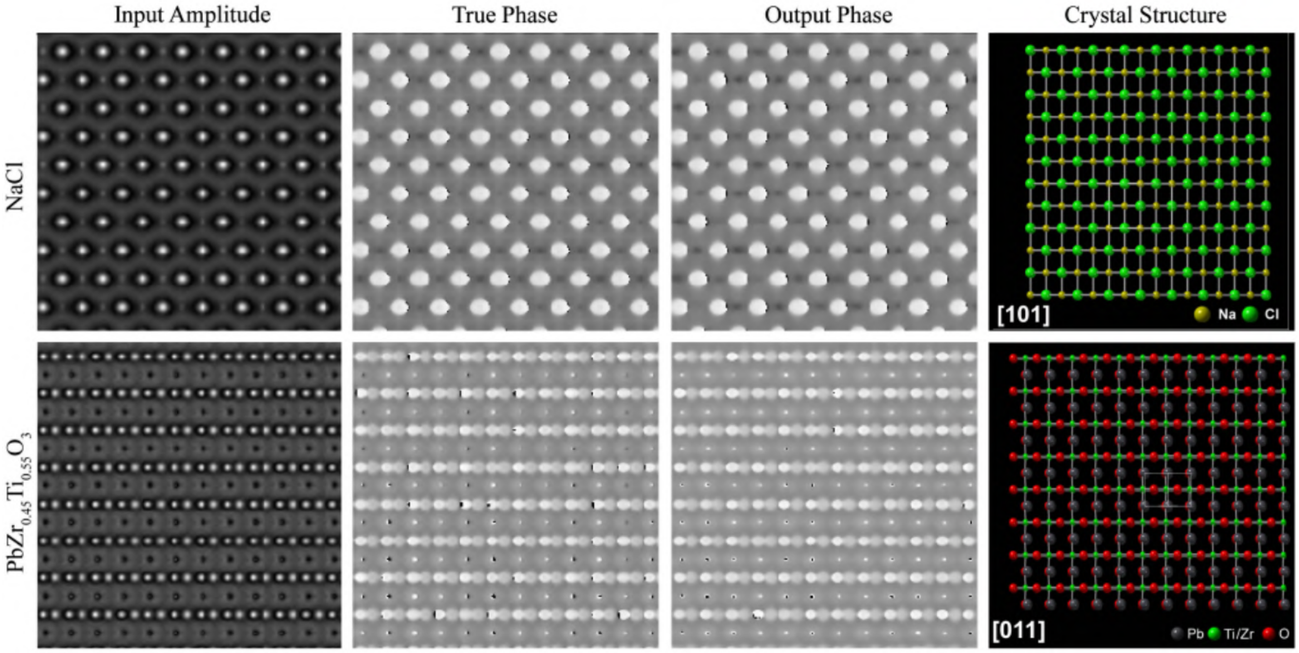


Figure 4. Example applications of a DNN to reconstruct phases of exit wavefunction from intensities of single TEM images. Phases in $[-\pi, \pi)$ rad are depicted on a linear greyscale from black to white, and Miller indices label projection directions. This figure is adapted from our earlier work[349] under a Creative Commons Attribution 4.0[73] license.

1.5 Exit Wavefunction Reconstruction

Electrons exhibit wave-particle duality[350,351], so electron propagation is often described by wave optics[352]. Applications of electron wavefunctions exiting materials[353] include determining projected potentials and corresponding crystal structure information[354,355], information storage, point spread function deconvolution, improving contrast, aberration correction[356], thickness measurement[357], and electric and magnetic structure determination[358,359]. Usually, exit wavefunctions are either iteratively reconstructed from focal series[360–364] or recorded by electron holography[352,363,365]. However, iterative reconstruction is often too slow for live applications, and holography is sensitive to distortions and may require expensive microscope modification.

Non-iterative methods based on DNNs have been developed to reconstruct optical exit wavefunctions from focal series[69] or single images[366–368]. Subsequently, DNNs have been developed to reconstruct exit wavefunctions from single TEM images[349], as shown in figure 4. Indeed, deep learning is increasingly being applied to accelerated quantum mechanics[369–374]. Other examples of DNNs adding new dimensions to data include semantic segmentation described in section 1.4, and reconstructing 3D atomic distortions from 2D images[375]. Non-iterative methods that do not use ANNs to recover phase information from single images have also been developed[376,377]. However, they are limited to defocused images in the Fresnel regime[376], or to non-planar incident wavefunctions in the Fraunhofer regime[377].

2 Resources

Access to scientific resources is essential to scientific enterprise[378]. Fortunately, most resources needed to get started with machine learning are freely available. This section provides directions to various machine learning resources, including how to access deep learning frameworks, a free GPU or tensor processing unit (TPU) to accelerate tensor computations, platforms that host datasets and source code, and pretrained models. To support the ideals of open science embodied by Plan S[378–380], we focus on resources that enhance collaboration and enable open access[381]. We also discuss how electron microscopes can interface with ANNs and the importance of machine learning resources in the context of electron microscopy. However, we expect that our insights into electron microscopy can be generalized to other scientific fields.

2.1 Hardware Acceleration

A DNN is an ANN with multiple layers that perform a sequence of tensor operations. Tensors can either be computed on central processing units (CPUs) or hardware accelerators[62], such as FPGAs[382–385], GPUs[386–388], and TPUs[389–391]. Most benchmarks indicate that GPUs and TPUs outperform CPUs for typical DNNs that could be used for image processing[392–396] in electron microscopy. However, GPU and CPU performance can be comparable when CPU computation is optimized[397]. TPUs often outperform GPUs[394], and FPGAs can outperform GPUs[398,399] if FPGAs have sufficient arithmetic units[400,401]. Typical power consumption per TFLOPS[402] decreases in order CPU, GPU, FPGA, then TPU, so hardware acceleration can help to minimize long-term costs and environmental damage[403].

For beginners, Google Colab[404–407] and Kaggle[408] provide hardware accelerators in ready-to-go deep learning environments. Free compute time on these platforms is limited as they are not intended for industrial applications. Nevertheless, the free compute time is sufficient for some research[409]. For more intensive applications, it may be necessary to get permanent access to hardware accelerators. If so, many online guides detail how to install[410,411] and set up an Nvidia[412] or AMD[413] GPU in a desktop computer for deep learning. However, most hardware comparisons for deep learning[414] focus on Nvidia GPUs as most deep learning frameworks use Nvidia's proprietary Compute Unified Device Architecture (CUDA) Deep Neural Network (cuDNN) primitives for deep learning[415], which are optimized for Nvidia GPUs. Alternatively, hardware accelerators may be accessible from a university or other institutional high performance computing (HPC) centre, or via a public cloud service provider[416–419].

Framework	License	Programming Interfaces
Apache SINGA[420]	Apache 2.0[421]	C++, Java, Python
BigDL[422]	Apache 2.0[423]	Python, Scala
Caffe[424,425]	BSD[426]	C++, MATLAB, Python
Chainer[427]	MIT[428]	Python
Deeplearning4j[429]	Apache 2.0[430]	Clojure, Java, Kotlin, Python, Scala
Dlib[431,432]	BSL[433]	C++
Flux[434]	MIT[435]	Julia
MATLAB Deep Learning Toolbox[436]	Proprietary[437]	MATLAB
Microsoft Cognitive Toolkit[438]	MIT[439]	BrainScript, C++, Python
Apache MXNet[440]	Apache 2.0[441]	C++, Clojure, Go, JavaScript, Julia, Matlab, Perl, Python, R, Scala
OpenNN[442]	GNU LGPL[443]	C++
PaddlePaddle[444]	Apache 2.0[445]	C++
PyTorch[446]	BSD[447]	C++, Python
TensorFlow[448,449]	Apache 2.0[450]	C++, C#, Go, Haskell, Julia, MATLAB, Python, Java, JavaScript, R, Ruby, Rust, Scala, Swift
Theano[451,452]	BSD[453]	Python
Torch[454]	BSD[455]	C, Lua
Wolfram Mathematica[456]	Proprietary[457]	Wolfram Language

Table 1. Deep learning frameworks with programming interfaces. Most frameworks have open source code and many support multiple programming languages.

2.2 Deep Learning Frameworks

A deep learning framework[9,458–464] (DLF) is an interface, library or tool for DNN development. Features often include automatic differentiation[465], heterogeneous computing, pretrained models, and efficient computing[466] with CUDA[467–469], cuDNN[415,470], OpenMP[471,472], or similar libraries. Popular DLFs tabulated in table 1 often have open source code and support multiple programming interfaces. Overall, TensorFlow[448,449] is the most popular DLF[473]. However, PyTorch[446] is the most popular DLF at top machine learning conferences[473,474]. Some DLFs also have extensions that ease development or extend functionality. For example, TensorFlow extensions[475] that ease development include Keras[476], Sonnet[477], Tensor2Tensor[478] and TFLearn[479,480], and extensions that add functionality include Addons[481], Agents[482], Dopamine[483], Federated[484–486], Probability[487], and TRFL[488]. In addition, DLFs are supplemented by libraries for predictive data analysis, such as scikit-learn[489].

A limitation of the DLFs in table 1 is that users must use programming interfaces. This is problematic as many electron microscopists have limited, if any, programming experience. To increase accessibility, a range of graphical user interfaces (GUIs) have been created for ANN development. For example, ANNdotNET[490], Create ML[491], Deep Cognition[492], Deep Network Designer[493], DIGITS[494], ENNUI[495], Expresso[496], Neural Designer[497], Waikato Environment for Knowledge Analysis[498–500] (WEKA) and ZeroCostDL4Mic[501]. The GUIs offer less functionality and scope for customization than programming interfaces. However, GUI-based DLFs are rapidly improving. Moreover, existing GUI functionality is more than sufficient to implement popular FNNs, such as image classifiers[272] and encoder-decoders[305–308,502–504].

2.3 Pretrained Models

Training ANNs is often time-consuming and computationally expensive[403]. Fortunately, pretrained models are available from a range of open access collections[505], such as Model Zoo[506], Open Neural Network Exchange[507–510] (ONNX) Model Zoo[511], TensorFlow Hub[512,513], and TensorFlow Model Garden[514]. Some researchers also provide pretrained models via project repositories[70,201,202,231,349]. Pretrained models can be used immediately or to transfer learning[515–521] to new applications. For example, by fine-tuning and augmenting the final layer of a pretrained model[522]. Benefits of transfer learning can include decreasing training time by orders of magnitude, reducing training data requirements, and improving generalization[520,523].

Using pretrained models is complicated by ANNs being developed with a variety of DLFs in a range of programming languages. However, most DLFs support interoperability. For example, by supporting the saving of models to a common format or to formats that are interoperable with the Neural Network Exchange Format[524] (NNEF) or ONNX formats. Many DLFs also support saving models to HDF5[525,526], which is popular in the pycroscopy[527,528] and HyperSpy[529,530] libraries used by electron microscopists. The main limitation of interoperability is that different DLFs may not support the same functionality. For example, Dlib[431,432] does not support recurrent neural networks[531–536] (RNNs).

2.4 Datasets

Randomly initialized ANNs[537] must be trained, validated, and tested with large, carefully partitioned datasets to ensure that they are robust to general use[538]. Most ANN training starts from random initialization, rather than transfer learning[515–521], as:

1. Researchers may be investigating modifications to ANN architecture or ability to learn.
2. Pretrained models may be unavailable or too difficult to find.
3. Models may quickly achieve sufficient performance from random initialization. For example, training an encoder-decoder based on Xception[539] to improve electron micrograph signal-to-noise[70] can require less training than for PASCAL VOC 2012[540] semantic segmentation[305].
4. There may be a high computing budget, so transfer learning is unnecessary[541,542].

There are millions of open access datasets[543,544] and a range of platforms that host[545–549] or aggregate[550–553] machine learning datasets. Openly archiving datasets drives scientific enterprise by reducing need to repeat experiments[554–558], enabling new applications through data mining[559,560], and standardizing performance benchmarks[561]. For example, popular datasets used to standardize image classification performance benchmarks include CIFAR-10[562,563], MNIST[564] and ImageNet[565]. A high range of both domain-specific and general platforms that host scientific data for free are listed by the Open Access Directory[566] and Nature Scientific Data[567]. For beginners, we recommend Zenodo[568] as it is free, open access, has an easy-to-use interface, and will host an unlimited number of datasets smaller than 50 GB for at least 20 years[569].

There are a range of platforms dedicated to hosting electron microscopy datasets, including the Caltech Electron Tomography Database[570] (ETDB-Caltech), Electron Microscopy Data Bank[571–576] (EMDataBank), and the Electron Microscopy Public Image Archive[577] (EMPIAR). However, most electron microscopy datasets are small, esoteric or are not partitioned for machine learning[231]. Nevertheless, a variety of large machine learning datasets for electron microscopy are being published in independent repositories[231,578,579], including Warwick Electron Microscopy Datasets[231] (WEMD) that we curated. In addition, a variety of databases host information that supports electron microscopy. For example, crystal structure databases provide data in standard formats[580,581], such as Crystallography Information Files[582–585] (CIFs). Large crystal structure databases[586–588] containing over 10^5 crystal structures include the Crystallography Open Database[589–594] (COD), Inorganic Crystal Structure Database[595–599] (ICSD), and National Institute of Standards and Technology (NIST) Crystal Data[600,601].

To achieve high performance, it may be necessary to curate a large dataset for ANN training[2]. However, large datasets like DeepMind Kinetics[602], ImageNet[565], and YouTube 8M[603] may take a team months to prepare. As a result, it may not be practical to divert sufficient staff and resources to curate a high-quality dataset, even if curation is partially automated[603–610]. To curate data, human capital can be temporarily and cheaply increased by using microjob services[611]. For example, through microjob platforms tabulated in table 2. Increasingly, platforms are emerging that specialize in data preparation for machine

Platform	Website	For Machine Learning
Amazon Mechanical Turk	`https://www.mturk.com`	General tasks
Appen	`https://appen.com`	Machine learning data preparation
Clickworker	`https://www.clickworker.com`	Machine learning data preparation
Fiverr	`https://www.fiverr.com`	General tasks
Hive	`https://thehive.ai`	Machine learning data preparation
iMerit	`https://imerit.net`	Machine learning data preparation
JobBoy	`https://www.jobboy.com`	General tasks
Minijobz	`https://minijobz.com`	General tasks
Microworkers	`https://www.microworkers.com`	General tasks
OneSpace	`https://freelance.onespace.com`	General tasks
Playment	`https://playment.io`	Machine learning data preparation
RapidWorkers	`https://rapidworkers.com`	General tasks
Scale	`https://scale.com`	Machine learning data preparation
Smart Crowd	`https://thesmartcrowd.lionbridge.com`	General tasks
Trainingset.ai	`https://www.trainingset.ai`	Machine learning data preparation
ySense	`https://www.ysense.com`	General tasks

Table 2. Microjob service platforms. The size of typical tasks varies for different platforms and some platforms specialize in preparing machine learning datasets.

learning. Nevertheless, microjob services may be inappropriate for sensitive data or tasks that require substantial domain-specific knowledge.

2.5 Source Code

Software is part of our cultural, industrial, and scientific heritage[612]. Source code should therefore be archived where possible. For example, on an open source code platform such as Apache Allura[613], AWS CodeCommit[614], Beanstalk[615], BitBucket[616], GitHub[617], GitLab[618], Gogs[619], Google Cloud Source Repositories[620], Launchpad[621], Phabricator[622], Savannah[623] or SourceForge[624]. These platforms enhance collaboration with functionality that helps users to watch[625] and contribute improvements[626–632] to source code. The choice of platform is often not immediately important for small electron microscopy projects as most platforms offer similar functionality. Nevertheless, functionality comparisons of open source platforms are available[633–635]. For beginners, we recommend GitHub as it is actively developed, scalable to large projects and has an easy-to-use interface.

2.6 Finding Information

Most web traffic[636,637] goes to large-scale web search engines[638–642] such as Bing, DuckDuckGo, Google, and Yahoo. This includes searches for scholarly content[643–645]. We recommend Google for electron microscopy queries as it appears to yield the best results for general[646–648], scholarly[644,645] and other[649] queries. However, general search engines can be outperformed by dedicated search engines for specialized applications. For example, for finding academic literature[650–652], data[653], jobs[654,655], publication venues[656], patents[657–660], people[661–663], and many other resources. The use of search engines is increasingly political[664–666] as they influence which information people see. However, most users appear to be satisfied with their performance[667].

Introductory textbooks are outdated[668,669] insofar that most information is readily available online. We find that some websites are frequent references for up-to-date and practical information:

1. Stack Overflow[670–675] is a source of working code snippets and a useful reference when debugging code.
2. Papers With Code State-of-the-Art[561] leaderboards rank the highest performing ANNs with open source code for various benchmarks.
3. Medium[676] and its subsidiaries publish blogs with up-to-date and practical advice about machine learning.
4. The Machine Learning subreddit[677] hosts discussions about machine learning. In addition, there is a Learn Machine Learning subreddit[678] aimed at beginners.
5. Dave Mitchell's DigitalMicrograph Scripting Website[679,680] hosts a collection of scripts and documentation for programming electron microscopes.
6. The Internet Archive[681,682] maintains copies of software and media, including webpages via its Wayback Machine[683–685].

7. Distill[686] is a journal dedicated to providing clear explanations about machine learning. Monetary prizes are awarded for excellent communication and refinement of ideas.

This list enumerates popular resources that we find useful, so it may introduce personal bias. However, alternative guides to useful resources are available[687–689]. We find that the most common issues finding information are part of an ongoing reproducibility crisis[690,691] where machine learning researchers do not publish their source code or data. Nevertheless, third party source code is sometimes available. Alternatively, ANNs can reconstruct source code from some research papers[692].

2.7 Scientific Publishing

The number of articles published per year in reputable peer-reviewed[693–697] scientific journals[698,699] has roughly doubled every nine years since the beginning of modern science[700]. There are now over 25000 peer-reviewed journals[699] with varying impact factors[701–703], scopes and editorial policies. Strategies to find the best journal to publish in include using online journal finders[704], seeking the advice of learned colleagues, and considering where similar research has been published. Increasingly, working papers are also being published in open access preprint archives[705–707]. For example, the arXiv[708,709] is a popular preprint archive for computer science, mathematics, and physics. Advantages of preprints include ensuring that research is openly available, increasing discovery and citations[710–714], inviting timely scientific discussion, and raising awareness to reduce unnecessary duplication of research. Many publishers have adapted to the popularity of preprints[705] by offering open access publication options[715–718] and allowing, and in some cases encouraging[719], the prior publication of preprints. Indeed, some journals are now using the arXiv to host their publications[720].

A variety of software can help authors prepare scientific manuscripts[721]. However, we think the most essential software is a document preparation system. Most manuscripts are prepared with Microsoft Word[722] or similar software[723]. However, Latex[724–726] is a popular alternative among computer scientists, mathematicians and physicists[727]. Most electron microscopists at the University of Warwick appear to prefer Word. A 2014 comparison of Latex and Word found that Word is better at all tasks other than typesetting equations[728]. However, in 2017 it become possible to use Latex to typeset equations within Word[727]. As a result, Word appears to be more efficient than Latex for most manuscript preparation. Nevertheless, Latex may still be preferable to authors who want fine control over typesetting[729,730]. As a compromise, we use Overleaf[731] to edit Latex source code, then copy our code to Word as part of proofreading to identify issues with grammar and wording.

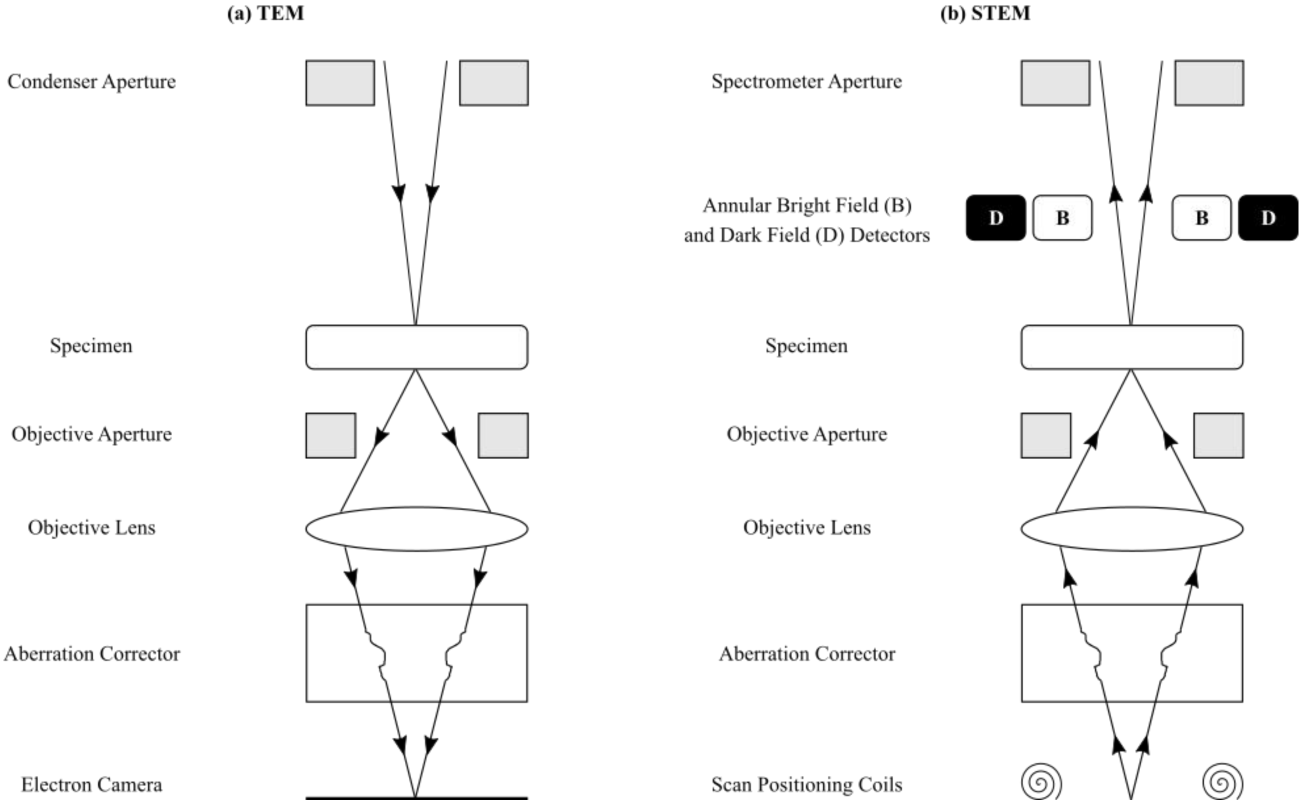


Figure 5. Reciprocity of TEM and STEM electron optics.

3 Electron Microscopy

An electron microscope is an instrument that uses electrons as a source of illumination to enable the study of small objects. Electron microscopy competes with a large range of alternative techniques for material analysis[732–734], including atomic force microscopy[735–737] (AFM); Fourier transformed infrared (FTIR) spectroscopy[738,739]; nuclear magnetic resonance[740–743] (NMR); Raman spectroscopy[744–750]; and x-ray diffraction[751,752] (XRD), dispersion[753], fluorescence[754,755] (XRF), and photoelectron spectroscopy[756,757] (XPS). Quantitative advantages of electron microscopes can include higher resolution and depth of field, and lower radiation damage than light microscopes[758]. In addition, electron microscopes can record images, enabling visual interpretation of complex structures that may otherwise be intractable. This section will briefly introduce varieties of electron microscopes, simulation software, and how electron microscopes can interface with ANNs.

3.1 Microscopes

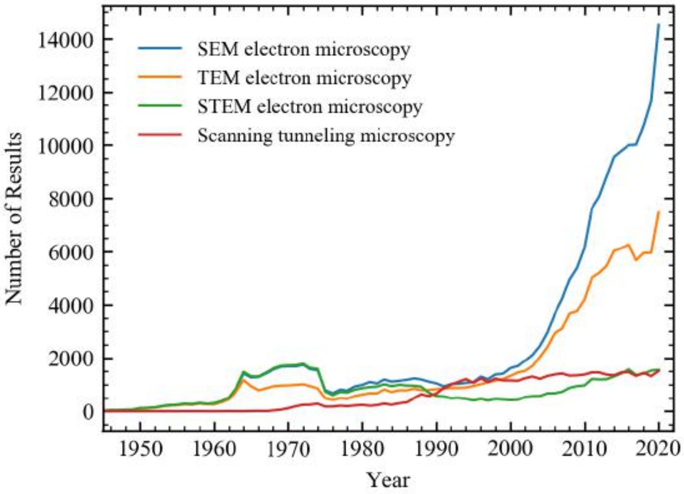


Figure 6. Numbers of results per year returned by Dimensions.ai abstract searches for SEM, TEM, STEM, STM and REM qualitate their popularities. The number of results for 2020 is extrapolated using the mean rate before 14th July 2020.

There are a variety of electron microscopes that use different illumination mechanisms. For example, reflection electron microscopy[759,760] (REM), scanning electron microscopy[761,762] (SEM), scanning transmission electron microscopy[763,764] (STEM), scanning tunnelling microscopy[765,766] (STM), and transmission electron microscopy[767–769] (TEM). To roughly gauge popularities of electron microscope varieties, we performed abstract searches with Dimenions.ai[651,770–772] for their abbreviations followed by “electron microscopy” e.g. “REM electron microscopy”. Numbers of results per year in figure 6 qualitate that popularity increases in order REM, STM, STEM, TEM, then SEM. It may be tempting to attribute the popularity of SEM over TEM to the lower cost of SEM[773], which increases accessibility. However, a range of considerations influence the procurement of electron microscopes[774] and hourly pricing at universities[775–779] is similar for SEM and TEM.

In SEM, material surfaces are scanned by sequential probing with a beam of electrons, which are typically accelerated to 0.2-40 keV. The SEM detects quanta emitted from where the beam interacts with the sample. Most SEM imaging uses low-energy secondary electrons. However, reflection electron microscopy[759,760] (REM) uses elastically backscattered electrons and is often complimented by a combination of reflection high-energy electron diffraction[780–782] (RHEED), reflection high-energy electron loss spectroscopy[783,784] (RHEELS) and spin-polarized low-energy electron microscopy[785–787] (SPLEEM). Some SEMs also detect Auger electrons[788,789]. To enhance materials characterization, most SEMs also detect light. The most common light detectors are for cathodoluminescence and energy dispersive r-ray[790,791] (EDX) spectroscopy. Nonetheless, some SEMs also detect Bremsstrahlung radiation[792].

Alternatively, TEM and STEM detect electrons transmitted through specimens. In conventional TEM, a single region is exposed to a broad electron beam. In contrast, STEM uses a fine electron beam to probe a series of discrete probing locations. Typically, electrons are accelerated across a potential difference to kinetic energies, E_k, of 80-300 keV. Electrons also have rest energy $E_\mathrm{e} = m_\mathrm{e}c^2$, where m_e is electron rest mass and c is the speed of light. The total energy, $E_t = E_\mathrm{e} + E_k$, of free electrons is related to their rest mass energy by a Lorentz factor, γ,

$$E_t = \gamma m_e c^2 \,, \quad (1)$$
$$\gamma = (1 - v^2/c^2)^{1/2} \,, \quad (2)$$

where v is the speed of electron propagation in the rest frame of an electron microscope. Electron kinetic energies in TEM and STEM are comparable to their rest energy, $E_e = 511$ keV[793], so relativistic phenomena[794,795] must be considered to accurately describe their dynamics.

Electrons exhibit wave-particle duality[350,351]. Thus, in an ideal electron microscope, the maximum possible detection angle, θ, between two point sources separated by a distance, d, perpendicular to the electron propagation direction is diffraction-limited. The resolution limit for imaging can be quantified by Rayleigh's criterion[796–798]

$$\theta \simeq 1.22\frac{\lambda}{d}, \quad (3)$$

where resolution increases with decreasing wavelength, λ. Electron wavelength increases with increasing accelerating voltage, as described by the relativistic de Broglie relation[799–801],

$$\lambda = hc\left(E_k^2 + 2E_e E_k\right)^{-1/2} \,, \quad (4)$$

where h is Planck's constant[793]. Electron wavelengths for typical acceleration voltages tabulated by JEOL are in picometres[802]. In comparison, Cu K-α x-rays, which are often used for XRD, have wavelengths near 0.15 nm[803]. In theory, electrons can therefore achieve over 100× higher resolution than x-rays. Electrons and x-rays are both ionizing; however, electrons often do less radiation damage to thin specimens than x-rays[758]. Tangentially, TEM and STEM often achieve over 10 times higher resolution than SEM[804] as transmitted electrons in TEM and STEM are easier to resolve than electrons returned from material surfaces in SEM.

In practice, TEM and STEM are also limited by incoherence[805–807] introduced by inelastic scattering, electron energy spread, and other mechanisms. TEM and STEM are related by an extension of Helmholtz reciprocity[808,809] where the source plane in a TEM corresponds to the detector plane in a STEM[810], as shown in figure 5. Consequently, TEM coherence is limited by electron optics between the specimen and image, whereas STEM coherence is limited by the illumination system. For conventional TEM and STEM imaging, electrons are normally incident on a specimen[811]. Advantages of STEM imaging can include higher contrast and resolution than TEM imaging, and lower radiation damage[812]. As a result, STEM is increasing being favoured over TEM for high-resolution studies. However, we caution that definitions of TEM and STEM resolution can be disparate[813].

In addition to conventional imaging, TEM and STEM include a variety of operating modes for different applications. For example, TEM operating configurations include electron diffraction[814]; convergent beam electron diffraction[815–817] (CBED); tomography[818–826]; and bright field[768,827–829], dark field[768,829] and annular dark field[830] imaging. Similarly, STEM operating configurations include differential phase contrast[831–834]; tomography[818,820,822,823]; and bright field[835,836] or dark field[837] imaging. Further, electron cameras[838,839] are often supplemented by secondary signal detectors. For example, elemental composition is often mapped by EDX spectroscopy, electron energy loss spectroscopy[840,841] (EELS) or wavelength dispersive spectroscopy[842,843] (WDS). Similarly, electron backscatter diffraction[844–846] (EBSD) can detect strain[847–849] and crystallization[850–852].

3.2 Contrast Simulation

The propagation of electron wavefunctions though electron microscopes can be described by wave optics[136]. Following, the most popular approach to modelling measurement contrast is multislice simulation[853,854], where an electron wavefunction is iteratively perturbed as it travels through a model of a specimen. Multislice software for electron microscopy includes ACEM[854–856], clTEM[857,858], cudaEM[859], Dr. Probe[860,861], EMSoft[862,863], JEMS[864], JMULTIS[865], MULTEM[866–868], NCEMSS[869,870], NUMIS[871], Prismatic[872–874], QSTEM[875], SimulaTEM[876], STEM-CELL[877], Tempas[878], and xHREM[879–884]. We find that most multislice software is a recreation and slight modification of common functionality, possibly due to a publish-or-perish culture in academia[885–887]. Bloch-wave simulation[854,888–892] is an alternative to multislice simulation that can reduce computation time and memory requirements for crystalline materials[893].

3.3 Automation

Most modern electron microscopes support Gatan Microscopy Suite (GMS) Software[894]. GMS enables electron microscopes to be programmed by DigitalMicrograph Scripting, a propriety Gatan programming language akin to a simplified version of C++.

A variety of DigitalMicrograph scripts, tutorials and related resources are available from Dave Mitchell's DigitalMicrograph Scripting Website[679,680], FELMI/ZFE's Script Database[895] and Gatan's Script library[896]. Some electron microscopists also provide DigitalMicrograph scripting resources on their webpages[897–899]. However, DigitalMicrograph scripts are slow insofar that they are interpreted at runtime, and there is limited native functionality for parallel and distributed computing. As a result, extensions to DigitalMicrograph scripting are often developed in other programming languages that offer more functionality.

Historically, most extensions were developed in C++[900]. This was problematic as there is limited documentation, the standard approach used outdated C++ software development kits such as Visual Studio 2008, and programming expertise required to create functions that interface with DigitalMicrograph scripts limited accessibility. To increase accessibility, recent versions of GMS now support python[901]. This is convenient as it enables ANNs developed with python to readily interface with electron microscopes. For ANNs developed with C++, users have the option to either create C++ bindings for DigitalMicrograph script or for python. Integrating ANNs developed in other programming languages is more complicated as DigitalMicrograph provides almost no support. However, that complexity can be avoided by exchanging files from DigitalMicrograph script to external libraries via a random access memory (RAM) disk[902] or secondary storage[903].

Increasing accessibility, there are collections of GMS plugins with GUIs for automation and analysis[897–899,904]. In addition, various individual plugins are available[905–909]. Some plugins are open source, so they can be adapted to interface with ANNs. However, many high-quality plugins are proprietary and closed source, limiting their use to automation of data collection and processing. Plugins can also be supplemented by a variety of libraries and interfaces for electron microscopy signal processing. For example, popular general-purpose software includes ImageJ[910], pycroscopy[527,528] and HyperSpy[529,530]. In addition, there are directories for tens of general-purpose and specific electron microscopy programs[911–913].

4 Components

Most modern ANNs are configured from a variety of DLF components. To take advantage of hardware accelerators[62], most ANNs are implemented as sequences of parallelizable layers of tensor operations[914]. Layers are often parallelized across data and may be parallelized across other dimensions[915]. This section introduces popular nonlinear activation functions, normalization layers, convolutional layers, and skip connections. To add insight, we provide comparative discussion and address some common causes of confusion.

4.1 Nonlinear Activation

In general, DNNs need multiple layers to be universal approximators[37–45]. Nonlinear activation functions[916,917] are therefore essential to DNNs as successive linear layers can be contracted to a single layer. Activation functions separate artificial neurons, similar to biological neurons[918]. To learn efficiently, most DNNs are tens or hundreds of layers deep[47,919–921]. High depth increases representational capacity[47], which can help training by gradient descent as DNNs evolve as linear models[922] and nonlinearities can create suboptimal local minima where data cannot be fit by linear models[923]. There are infinitely many possible activation functions. However, most activation functions have low polynomial order, similar to physical Hamiltonians[47].

Most ANNs developed for electron microscopy are for image processing, where the most popular nonlinearities are rectifier linear units[924,925] (ReLUs). The ReLU activation, $f(x)$, of an input, x, and its gradient, $\partial_x f(x)$, are

$$f(x) = \max(0, x) \tag{5a}$$

$$\frac{\partial f(x)}{\partial x} = \begin{cases} 0, & \text{if } x \leq 0 \\ 1, & \text{if } x > 0 \end{cases} \tag{5b}$$

Popular variants of ReLUs include Leaky ReLU[926],

$$f(x) = \max(\alpha x, x) \tag{6a}$$

$$\frac{\partial f(x)}{\partial x} = \begin{cases} \alpha, & \text{if } x \leq 0 \\ 1, & \text{if } x > 0 \end{cases} \tag{6b}$$

where α is a hyperparameter, parametric ReLU[22] (PreLU) where α is a learned parameter, dynamic ReLU where α is a learned function of inputs[927], and randomized leaky ReLU[928] (RReLU) where α is chosen randomly. Typically, learned PreLU α are higher the nearer a layer is to ANN inputs[22]. Motivated by limited comparisons that do not show a clear performance difference between ReLU and leaky ReLU[929], some blogs[930] argue against using leaky ReLU due to its higher computational requirements and complexity. However, an in-depth comparison found that leaky ReLU variants consistently slightly outperform ReLU[928]. In addition, the non-zero gradient of leaky ReLU for $x \leq 0$ prevents saturating, or "dying", ReLU[931–933], where the zero gradient of ReLUs stops learning.

There are a variety of other piecewise linear ReLU variants that can improve performance. For example, ReLUh activations are limited to a threshold[934], h, so that

$$f(x) = \min(\max(0,x),h) \tag{7a}$$

$$\frac{\partial f(x)}{\partial x} = \begin{cases} 0, & \text{if } x \leq 0 \\ 1, & \text{if } 0 < x \leq h \\ 0, & \text{if } x > h \end{cases} \tag{7b}$$

Thresholds near $h = 6$ are often effective, so popular choice is ReLU6. Another popular activation is concatenated ReLU[935] (CReLU), which is the concatenation of ReLU(x) and ReLU$(-x)$. Other ReLU variants include adaptive convolutional[936], bipolar[937], elastic[938], and Lipschitz[939] ReLUs. However, most ReLU variants are uncommon as they are more complicated than ReLU and offer small, inconsistent, or unclear performance gains. Moreover, it follows from the universal approximator theorems[37–45] that disparity between ReLU and its variants approaches zero as network depth increases.

In shallow networks, curved activation functions with non-zero Hessians often accelerate convergence and improve performance. A popular activation is the exponential linear unit[940] (ELU),

$$f(x) = \begin{cases} \alpha(\exp(x) - 1), & \text{if } x \leq 0 \\ x, & \text{if } x \geq 0 \end{cases} \tag{8a}$$

$$\frac{\partial f(x)}{\partial x} = \begin{cases} \alpha \exp(x), & \text{if } x \leq 0 \\ 1, & \text{if } x \geq 0 \end{cases} \tag{8b}$$

where α is a learned parameter. Further, a scaled ELU[941] (SELU),

$$f(x) = \begin{cases} \lambda\alpha(\exp(x) - 1), & \text{if } x \leq 0 \\ \lambda x, & \text{if } x \geq 0 \end{cases} \tag{9a}$$

$$\frac{\partial f(x)}{\partial x} = \begin{cases} \lambda\alpha \exp(x), & \text{if } x \leq 0 \\ \lambda, & \text{if } x \geq 0 \end{cases} \tag{9b}$$

with fixed $\alpha = 1.67326$ and scale factor $\lambda = 1.0507$ can be used to create self-normalizing neural networks (SNNs). A SNN cannot be derived from ReLUs or most other activation functions. Activation functions with curvature are especially common in ANNs with only a couple of layers. For example, activation functions in radial basis function (RBF) networks[942–945], which are efficient universal approximators, are often Gaussians, multiquadratics, inverse multiquadratics, or square-based RBFs[946]. Similarly, support vector machines[947–949] (SVMs) often use RBFs, or sigmoids,

$$f(x) = \frac{1}{1 + \exp(-x)} \tag{10a}$$

$$\frac{\partial f(x)}{\partial x} = f(x)\,(1 - f(x)) \tag{10b}$$

Sigmoids can also be applied to limit the support of outputs. Unscaled, or "logistic", sigmoids are often denoted $\sigma(x)$ and are related to tanh by $\tanh(x) = 2\sigma(2x) - 1$. To avoid expensive $\exp(-x)$ in the computation of tanh, we recommend K-tanH[950], LeCun tanh[951], or piecewise linear approximation[952,953].

The activation functions introduced so far are scalar functions than can be efficiently computed in parallel for each input element. However, functions of vectors, $\mathbf{x} = \{x_1, x_2, \ldots\}$, are also popular. For example, softmax activation[954],

$$f(\mathbf{x}) = \frac{\exp(\mathbf{x})}{\text{sum}(\exp(\mathbf{x}))} \tag{11a}$$

$$\frac{f(\mathbf{x})}{\partial x_j} = \sum_i f(\mathbf{x})_i(\delta_{ij} - f(\mathbf{x})_j) \tag{11b}$$

is often applied before computing cross-entropy losses for classification networks. Similarly, Ln vector normalization,

$$f(\mathbf{x}) = \frac{\mathbf{x}}{||\mathbf{x}||_n} \tag{12a}$$

$$\frac{f(\mathbf{x})}{\partial x_j} = \frac{1}{||\mathbf{x}||_n}\left(1 - \frac{x_j^n}{||\mathbf{x}||_n^n}\right) \tag{12b}$$

is often applied to n-dimensional vectors to ensure that they lie on a unit n-sphere[349]. Finally, max pooling[955,956],

$$f(\mathbf{x}) = \max(\mathbf{x}) \tag{13a}$$

$$\frac{f(\mathbf{x})}{\partial x_j} = \begin{cases} 1, & \text{if } j = \text{argmax}(\mathbf{x}) \\ 0, & \text{if } j \neq \text{argmax}(\mathbf{x}) \end{cases} \tag{13b}$$

is another popular multivariate activation function that is often used for downsampling. However, max pooling has fallen out of favour as it is often outperformed by strided convolutional layers[957]. Other vector activation functions include squashing nonlinearities for dynamic routing by agreement in capsule networks[958] and cosine similarity[959].

There is a range of other activation functions that are not detailed here for brevity. Further, finding new activation functions is an active area of research[960,961]. Notable variants include choosing activation functions from a set before training[962,963] and learning activation functions[962,964–967]. Activation functions can also encode probability distributions[968–970] or include noise[953]. Finally, there are a variety of other deterministic activation functions[961,971]. In electron microscopy, most ANNs enable new or enhance existing applications. Subsequently, we recommend using computationally efficient and established activation functions unless there is a compelling reason to use a specialized activation function.

4.2 Normalization

Normalization[972–974] standardizes signals, which can accelerate convergence by gradient descent and improve performance. Batch normalization[975–980] is the most popular normalization layer in image processing DNNs trained with minibatches of N examples. Technically, a "batch" is an entire training dataset and a "minibatch" is a subset; however, the "mini" is often omitted where meaning is clear from context. During training, batch normalization applies a transform,

$$\mu_B = \frac{1}{N} \sum_{i=1}^{N} x_i \,, \tag{14}$$

$$\sigma_B^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu_B)^2 \,, \tag{15}$$

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \mu_B}{(\sigma_B^2 + \varepsilon)^{1/2}} \,, \tag{16}$$

$$\text{BatchNorm}(\mathbf{x}) = \gamma \hat{\mathbf{x}} + \beta \,, \tag{17}$$

where $\mathbf{x} = \{x_1, ..., x_N\}$ is a batch of layer inputs, γ and β are a learnable scale and shift, and ε is a small constant added for numerical stability. During inference, batch normalization applies a transform,

$$\text{BatchNorm}(\mathbf{x}) = \frac{\gamma}{(\text{Var}[x] + \varepsilon)^{1/2}} \mathbf{x} + \left(\beta - \frac{\gamma \text{E}[x]}{(\text{Var}[x] + \varepsilon)^{1/2}} \right) , \tag{18}$$

where E[x] and Var[x] are expected batch means and variances. For convenience, E[x] and Var[x] are often estimated with exponential moving averages that are tracked during training. However, E[x] and Var[x] can also be estimated by propagating examples through an ANN after training.

Increasing batch size stabilizes learning by averaging destabilizing loss spikes over batches[261]. Batched learning also enables more efficient utilization of modern hardware accelerators. For example, larger batch sizes improve utilization of GPU memory bandwidth and throughput[391,981,982]. Using large batches can also be more efficient than many small batches when distributing training across multiple CPU clusters or GPUs due to communication overheads. However, the performance benefits of large batch sizes can come at the cost of lower test accuracy as training with large batches tends to converge to sharper minima[983,984]. As a result, it often best not to use batch sizes higher than $N \approx 32$ for image classification[985]. However, learning rate scaling[541] and layer-wise adaptive learning rates[986] can increase accuracy of training with fixed larger batch sizes. Batch size can also be increased throughout training without compromising accuracy[987] to exploit effective learning rates being inversely proportional to batch size[541,987]. Alternatively, accuracy can be improved by creating larger batches from replicated instances of training inputs with different data augmentations[988].

There are a few caveats to batch normalization. Originally, batch normalization was applied before activation[976]. However, applying batch normalization after activation often slightly improves performance[989,990]. In addition, training can be sensitive to the often-forgotten ε hyperparameter[991] in equation 16. Typically, performance decreases as ε is increased above $\varepsilon \approx 0.001$; however, there is a sharp increase in performance around $\varepsilon = 0.01$ on ImageNet. Finally, it is often assumed that batches are representative of the training dataset. This is often approximated by shuffling training data to sample independent and identically distributed (i.i.d.) samples. However, performance can often be improved by prioritizing sampling[992,993]. We observe that batch normalization is usually effective if batch moments, μ_B and σ_B, have similar values for every batch.

Batch normalization is less effective when training batch sizes are small, or do not consist of independent samples. To improve performance, standard moments in equation 16 can be renormalized[994] to expected means, μ, and standard deviations, σ,

$$\hat{\mathbf{x}} \leftarrow r\hat{\mathbf{x}} + d\,, \tag{19}$$

$$r = \text{clip}_{[1/r_\text{max}, r_\text{max}]}\left(\frac{\sigma_B}{\sigma}\right)\,, \tag{20}$$

$$d = \text{clip}_{[-d_\text{max}, d_\text{max}]}\left(\frac{\mu_B - \mu}{\sigma}\right)\,, \tag{21}$$

where gradients are not backpropagated with respect to (w.r.t.) the renormalization parameters, r and d. Moments, μ and σ are tracked by exponential moving averages and clipping to r_max and d_max improves learning stability. Usually, clipping values are increased from starting values of $r_\text{max} = 1$ and $d_\text{max} = 0$, which correspond to batch normalization, as training progresses. Another approach is virtual batch normalization[995] (VBN), which estimates μ and σ from a reference batch of samples and does not require clipping. However, VBN is computationally expensive as it requires computing a second batch of statistics at every training iteration. Finally, online[996] and streaming[974] normalization enable training with small batch sizes by replace μ_B and σ_B in equation 16 with their exponential moving averages.

There are alternatives to the L_2 batch normalization of equations 14-18 that standardize to different Euclidean norms. For example, L_1 batch normalization[997] computes

$$s_1 = \frac{1}{N}\sum_{i=1}^{N} |x_i - \mu_B|\,, \tag{22}$$

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \mu_B}{C_{L_1} s_1}\,, \tag{23}$$

where $C_{L_1} = (\pi/2)^{1/2}$. Although the C_{L_1} factor could be learned by ANN parameters, its inclusion accelerates convergence of the original implementation of L_1 batch normalization[997]. Another alternative is L_∞ batch normalization[997], which computes

$$s_\infty = \text{mean}(\text{top}_k(|\mathbf{x} - \mu_B|))\,, \tag{24}$$

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \mu_B}{C_{L_\infty} s_\infty}\,, \tag{25}$$

where C_{L_∞} is a scale factor, and $\text{top}_k(\mathbf{x})$ returns the k highest elements of $\mathbf{x}$. Hoffer *et al* suggest $k = 10$[997]. Some L_1 batch normalization proponents claim that L_1 batch normalization outperforms[975] or achieves similar performance[997] to L_2 batch normalization. However, we found that L_1 batch normalization often lowers performance in our experiments. Similarly, L_∞ batch normalization often lowers performance[997]. Overall, L_1 and L_∞ batch normalization do not appear to offer a substantial advantage over L_2 batch normalization.

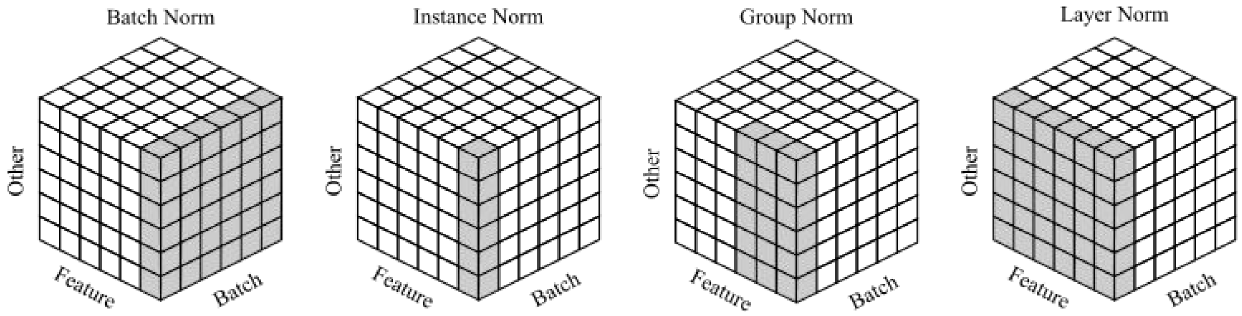


Figure 7. Visual comparison of various normalization methods highlighting regions that they normalize. Regions can be normalized across batch, feature and other dimensions, such as height and width.

A variety of layers normalize samples independently, including layer, instance, and group normalization. They are compared with batch normalization in figure 7. Layer normalization[998,999] is a transposition of batch normalization that is computed across feature channels for each training example, instead of across batches. Batch normalization is ineffective in RNNs; however, layer normalization of input activations often improves accuracy[998]. Instance normalization[1000] is an extreme version of layer normalization that standardizes each feature channel for each training example. Instance normalization was developed for style transfer[1001–1005] and makes ANNs insensitive to input image contrast. Group normalization[1006] is intermediate to instance and layer normalization insofar that it standardizes groups of channels for each training example.

The advantages of a set of multiple different normalization layers, Ω, can be combined by switchable normalization[1007,1008], which standardizes to

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \sum\limits_{z \in \Omega} \lambda_z^\mu \mu_z}{\sum\limits_{z \in \Omega} \lambda_z^\sigma \sigma_z}, \tag{26}$$

where μ_z and σ_z are means and standard deviations computed by normalization layer z, and their respective importance ratios, λ_z^μ and λ_z^σ, are trainable parameters that are softmax activated to sum to unity. Combining batch and instance normalization statistics outperforms batch normalization for a range of computer vision tasks[1009]. However, most layers strongly weighted either batch or instance normalization, with most preferring batch normalization. Interestingly, combining batch, instance and layer normalization statistics[1007,1008] results in instance normalization being preferred in earlier layers, whereas layer normalization was preferred in the later layers, and batch normalization was preferred in the middle layers. Smaller batch sizes lead to a preference towards layer normalization and instance normalization. Limitingly, using multiple normalization layers increases computation. To limit expense, we therefore recommend either defaulting to batch normalization, or progressively using single instance, batch or layer normalization layers.

A significant limitation of batch normalization is that it is not effective in RNNs. This is a limited issue as most electron microscopists are developing CNNs for image processing. However, we anticipate that RNNs may become more popular in electron microscopy following the increasing popularity of reinforcement learning[1010]. In addition to general-purpose alternatives to batch normalization that are effective in RNNs, such as layer normalization, there are a variety of dedicated normalization schemes. For example, recurrent batch normalization[1011,1012] uses distinct normalization layers for each time step. Alternatively, batch normalized RNNs[1013] only have normalization layers between their input and hidden states. Finally, online[996] and streaming[974] normalization are general-purpose solutions that improve the performance of batch normalization in RNNs by applying batch normalization based on a stream of past batch statistics.

Normalization can also standardize trainable weights, $\mathbf{w}$. For example, weight normalization[1014],

$$\text{WeightNorm}(\mathbf{w}) = \frac{g}{||\mathbf{w}||_2}\mathbf{w}, \tag{27}$$

decouples the L2 norm, g, of a variable from its direction. Similarly, weight standardization[1015] subtracts means from variables and divides them by their standard deviations,

$$\text{WeightStd}(\mathbf{w}) = \frac{\mathbf{w} - \text{mean}(\mathbf{w})}{\text{std}(\mathbf{w})}, \tag{28}$$

similar to batch normalization. Weight normalization often outperforms batch normalization at small batch sizes. However, batch normalization consistently outperforms weight normalization at larger batch sizes used in practice[1016]. Combining weight normalization with running mean-only batch normalization can accelerate convergence[1014]. However, similar final accuracy can be achieved without mean-only batch normalization at the cost of slower convergence, or with the use of zero-mean preserving activation functions[937,997]. To achieve similar performance to batch normalization, norm-bounded weight normalization[997] can be applied to DNNs with scale-invariant activation functions, such as ReLU. Norm-bounded weight normalization fixes g at initialization to avoid learning instability[997,1016], and scales outputs with the final DNN layer.

Limitedly, weight normalization encourages the use of a small number of features to inform activations[1017]. To maximize feature utilization, spectral normalization[1017],

$$\text{SpectralNorm}(\mathbf{w}) = \frac{\mathbf{w}}{\sigma(\mathbf{w})}, \tag{29}$$

divides tensors by their spectral norms, $\sigma(\mathbf{w})$. Further, spectral normalization limits Lipschitz constants[1018], which often improves generative adversarial network[197–200] (GAN) training by bounding backpropagated discriminator gradients[1017]. The spectral norm of $\mathbf{v}$ is the maximum value of a diagonal matrix, Σ, in the singular value decomposition[1019–1022] (SVG),

$$\mathbf{v} = \mathbf{U}\Sigma\mathbf{V}^*, \tag{30}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices of orthonormal eigenvectors for $\mathbf{v}\mathbf{v}^T$ and $\mathbf{v}^T\mathbf{v}$, respectively. To minimize computation, $\sigma(\mathbf{w})$ is often approximated by the power iteration method[1023, 1024],

$$\hat{\mathbf{v}} \leftarrow \frac{\mathbf{w}^{\mathrm{T}}\hat{\mathbf{u}}}{||\mathbf{w}^{\mathrm{T}}\hat{\mathbf{u}}||_2}\,, \tag{31}$$

$$\hat{\mathbf{u}} \leftarrow \frac{\mathbf{w}\hat{\mathbf{v}}}{||\mathbf{w}\hat{\mathbf{v}}||_2}\,, \tag{32}$$

$$\sigma(\mathbf{w}) \simeq \hat{\mathbf{u}}^T\mathbf{w}\hat{\mathbf{v}}\,, \tag{33}$$

where one iteration of equations 31-32 per training iteration is usually sufficient.

Parameter normalization can complement or be combined with signal normalization. For example, scale normalization[1025],

$$\mathrm{ScaleNorm}(\mathbf{x}) = \frac{g}{||\mathbf{x}||_2}\mathbf{x}\,, \tag{34}$$

learns scales, g, for activations, and is often combined with weight normalization[1014, 1026] in transformer networks. Similarly, cosine normalization[959],

$$\mathrm{CosineNorm}(\mathbf{x}) = \frac{\mathbf{w}}{||\mathbf{w}||_2} \cdot \frac{\mathbf{x}}{||\mathbf{x}||_2}\,, \tag{35}$$

computes products of L2 normalized parameters and signals. Both scale and cosine normalization can outperform batch normalization.

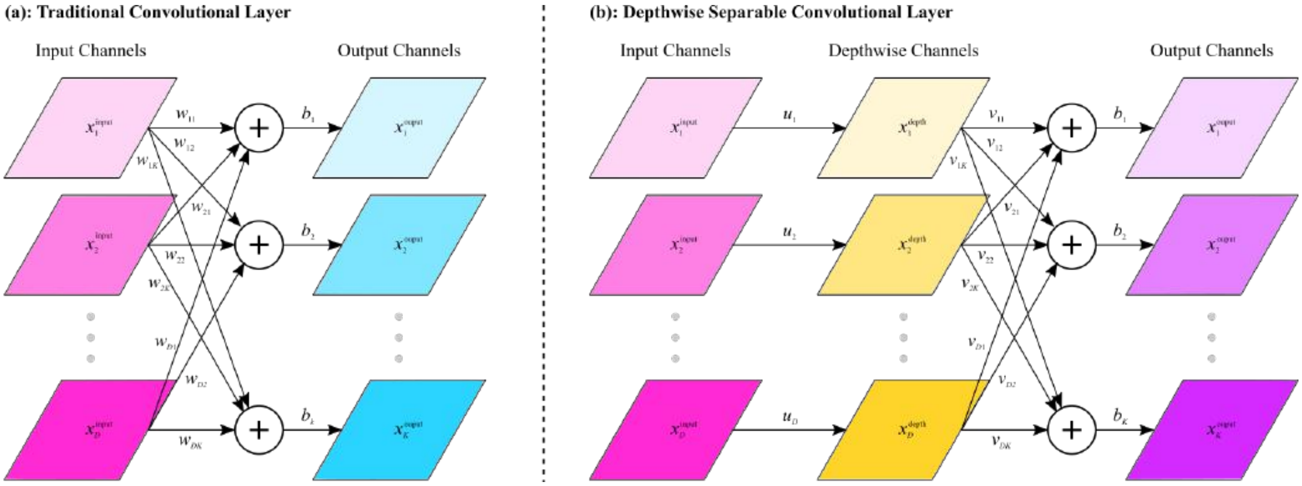


Figure 8. Visualization of convolutional layers. a) Traditional convolutional layer where output channels are sums of biases and convolutions of weights with input channels. b) Depthwise separable convolutional layer where depthwise convolutions compute one convolution with weights for each input channel. Output channels are sums of biases and pointwise convolutions weights with depthwise channels.

4.3 Convolutional Layers

A convolutional neural network[1027–1030] (CNN) is trained to weight convolutional kernels to exploit local correlations, such as spatial correlations in electron micrographs[231]. Historically, the development of CNNs was inspired by primate visual cortices[1031], where partially overlapping neurons are only stimulated by visual stimuli within their receptive fields. Based on this idea, Fukushima published his Neocognitron[1032–1035] in 1980. Convolutional formulations were then published by Atlas *et al* in 1988 for a single-layer CNN[1036], and LeCun *et al* in 1998 for a multi-layer CNN[1037, 1038]. Following, GPUs were applied to accelerate convolutions in 2010[1039], leading to a breakthrough in classification performance on ImageNet with AlexNet in 2012[71]. Indeed, the deep learning era is often partitioned into before and after AlexNet[19]. Deep CNNs are now ubiquitous. For example, there are review papers on applications of CNNs to action recognition in videos[1040], cytometry[1041], image and video compression[1042, 1043], image background subtraction[1044], image classification[272], image style transfer[1001], medical image analysis[332–334, 1045–1052], object detection[1053, 1054], semantic image segmentation[304, 332–334], and text classification[1055].

In general, the convolution of two functions, f and g, is

$$(f*g)(x) := \int_{s\in\Omega} f(s)g(x-s)\,\mathrm{d}s\,, \tag{36}$$

and their cross-correlation is

$$(f\circ g)(x) := \int_{s\in\Omega} f(s)g(x+s)\,\mathrm{d}s\,, \tag{37}$$

where integrals have unlimited support, Ω. In a CNN, convolutional layers sum convolutions of feature channels with trainable kernels, as shown in figure 8. Thus, f and g are discrete functions and the integrals in equations 36-37 can be replaced with limited summations. Since cross-correlation is equivalent to convolution if the kernel is flipped in every dimension, and CNN kernels are usually trainable, convolution and cross-correlation is often interchangeable in deep learning. For example, a TensorFlow function named "tf.nn.convolution" computes cross-correlations[1056]. Nevertheless, the difference between convolution and cross-correlation can be source of subtle errors if convolutional layers from a DLF are used in an image processing pipeline with static asymmetric kernels.

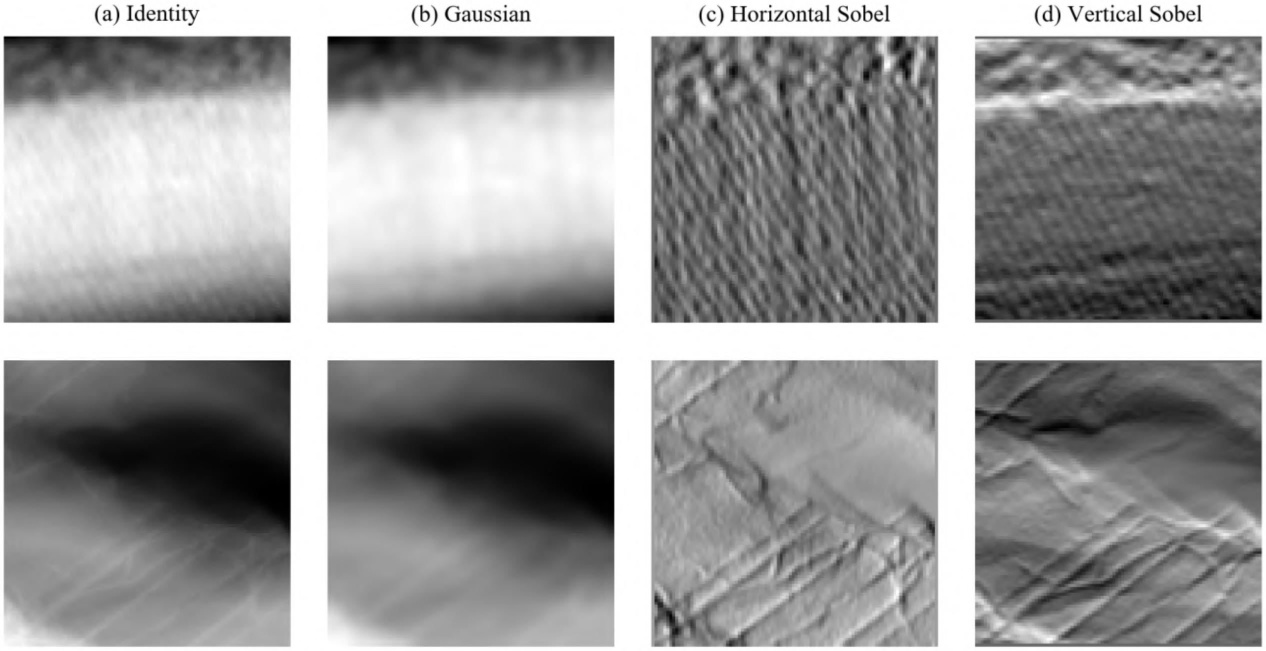


Figure 9. Two 96×96 electron micrographs a) unchanged, and filtered by b) a 5×5 symmetric Gaussian kernel with a 2.5 px standard deviation, c) a 3×3 horizontal Sobel kernel, and d) a 3×3 vertical Sobel kernel. Intensities in a) and b) are in [0, 1], whereas intensities in c) and d) are in [-1, 1].

Kernels designed by humans[1057] are often convolved in image processing pipelines. For example, convolutions of electron micrographs with Gaussian and Sobel kernels are shown in figure 9. Gaussian kernels compute local averages, blurring images and suppressing high-frequency noise. For example, a 5×5 symmetric Gaussian kernel with a 2.5 px standard deviation is

$$\begin{bmatrix} 0.1689 \\ 0.2148 \\ 0.2326 \\ 0.2148 \\ 0.1689 \end{bmatrix} \begin{bmatrix} 0.1689 & 0.2148 & 0.2326 & 0.2148 & 0.1689 \end{bmatrix} = \begin{bmatrix} 0.0285 & 0.0363 & 0.0393 & 0.0363 & 0.0285 \\ 0.0363 & 0.0461 & 0.0500 & 0.0461 & 0.0363 \\ 0.0393 & 0.0500 & 0.0541 & 0.0500 & 0.0393 \\ 0.0363 & 0.0461 & 0.0500 & 0.0461 & 0.0363 \\ 0.0285 & 0.0363 & 0.0393 & 0.0363 & 0.0285 \end{bmatrix}. \tag{38}$$

Alternatives to Gaussian kernels for image smoothing[1058] include mean, median and bilateral filters. Sobel kernels compute horizontal and vertical spatial gradients that can be used for edge detection[1059]. For example, 3×3 Sobel kernels are

$$\begin{bmatrix}1\\2\\1\end{bmatrix}\begin{bmatrix}1 & 0 & -1\end{bmatrix}=\begin{bmatrix}1 & 0 & -1\\2 & 0 & -2\\1 & 0 & -1\end{bmatrix} \tag{39a}$$

$$\begin{bmatrix}1\\0\\-1\end{bmatrix}\begin{bmatrix}1 & 2 & 1\end{bmatrix}=\begin{bmatrix}1 & 2 & 1\\0 & 0 & 0\\-1 & -2 & -1\end{bmatrix} \tag{39b}$$

Alternatives to Sobel kernels offer similar utility, and include extended Sobel[1060], Scharr[1061, 1062], Kayyali[1063], Roberts cross[1064] and Prewitt[1065] kernels. Two-dimensional Gaussian and Sobel kernels are examples of linearly separable, or "flattenable", kernels, which can be split into two one-dimensional kernels, as shown in equations 38-39b. Kernel separation can decrease computation in convolutional layers by convolving separated kernels in series, and CNNs that only use separable convolutions are effective[1066–1068]. However, serial convolutions decrease parallelization and separable kernels have fewer degrees of freedom, decreasing representational capacity. Following, separated kernels are usually at least 5×5, and separated 3×3 kernels are unusual. Even-sized kernels, such as 2×2 and 4×4, are rare as symmetric padding is needed to avoid information erosion caused by spatial shifts of feature maps[1069].

A traditional 2D convolutional layer maps inputs, x^{input}, with height H, width, W, and depth, D, to

$$x_{kij}^{\text{output}} = b_k + \sum_{d=1}^{D}\sum_{m=1}^{M}\sum_{n=1}^{N} w_{dkmn} x_{d(i+m-1)(j+n-1)}^{\text{input}} \,, i \in [1, H-M+1]\,, j \in [1, W-N+1]\,, \tag{40}$$

where K output channels are indexed by $k \in [1,K]$, is the sum of a bias, b, and convolutions of each input channel with $M \times N$ kernels with weights, w. For clarity, a traditional convolutional layer is visualized in figure 8a. Convolutional layers for 1D, 3D and higher-dimensional kernels[1070] have a similar form to 2D kernels, where kernels are convolved across each dimension. Most inputs to convolutional layers are padded[1071, 1072] to avoid reducing spatial resolutions by kernel sizes, which could remove all resolution in deep networks. Padding is computationally inexpensive and eases implementations of ANNs that would otherwise combine layers with different sizes, such as FractalNet[1073], Inception[1074–1076], NASNet[1077], recursive CNNs[1078, 1079], and ResNet[1080]. Pre-padding inputs results in higher performance than post-padding outputs[1081]. Following AlexNet[71], most convolutional layers are padded with zeros for simplicity. Reflection and replication padding achieve similar results to zero padding[1072]. However, padding based on partial convolutions[1082] consistently outperforms other methods[1072].

Convolutional layers are similar to fully connected layers used in multilayer perceptrons[1083, 1084] (MLPs). For comparison with equation 40, a fully connected, or "dense", layer in a MLP computes

$$x_k^{\text{output}} = b_k + \sum_{d=1}^{D} w_{dk} x_d^{\text{input}} \,, \tag{41}$$

where every input element is connected to every output element. Convolutional layers reduce computation by making local connections within receptive fields of convolutional kernels, and by convolving kernels rather than using different weights at each input position. Intermediately, fully connected layers can be regularized to learn local connections[1085]. Fully connected layers are sometimes used at the middle of encoder-decoders[1086]. However, such fully connected layers can often be replaced by multiscale atrous, or "holey", convolutions[955] in an atrous spatial pyramid pooling[305, 306] (ASPP) module to decrease computation without a significant decrease in performance. Alternatively, weights in fully connected layers can be decomposed into multiple smaller tensors to decrease computation without significantly decreasing performance[1087, 1088].

Convolutional layers can perform a variety of convolutional arithmetic[955]. For example, strided convolutions[1089] usually skip computation of outputs that are not at multiples of an integer spatial stride. Most strided convolutional layers are applied throughout CNNs to sequentially decrease spatial extent, and thereby decrease computational requirements. In addition, strided convolutions are often applied at the start of CNNs[539, 1074–1076] where most input features can be resolved at a lower resolution than the input. For simplicity and computational efficiency, stride is typically constant within a convolutional layer; however, increasing stride away from the centre of layers can improve performance[1090]. To increase spatial resolution, convolutional layers often use reciprocals of integer strides[1091]. Alternatively, spatial resolution can be increased by combining interpolative upsampling with an unstrided convolutional layer[1092, 1093], which can help to minimize output artefacts.

Convolutional layers couple the computation of spatial and cross-channel convolutions. However, partial decoupling of spatial and cross-channel convolutions by distributing inputs across multiple convolutional layers and combining outputs can improve performance. Partial decoupling of convolutions is prevalent in many seminal DNN architectures, including FractalNet[1073], Inception[1074–1076], NASNet[1077]. Taking decoupling to an extreme, depthwise separable convolutions[539, 1094, 1095] shown in figure 8 compute depthwise convolutions,

$$x_{dij}^{\text{depth}} = \sum_{m=1}^{M}\sum_{n=1}^{N} u_{dmn} x_{d(i+m-1)(j+n-1)}^{\text{input}} \,, i \in [1, H-M+1]\,, j \in [1, W-N+1]\,, \tag{42}$$

then compute pointwise 1×1 convolutions for D intermediate channels,

$$x_{kij}^{\text{output}} = b_k + \sum_{d=1}^{D} v_{dk}^{\text{point}} x_{dij}^{\text{depth}}\,, \tag{43}$$

where K output channels are indexed by $k \in [1,K]$. Depthwise convolution kernels have weights, u, and the depthwise layer is often followed by extra batch normalization before pointwise convolution to improve performance and accelerate convergence[1094]. Increasing numbers of channels with pointwise convolutions can increase accuracy[1094], at the cost of increased computation. Pointwise convolutions are a special case of traditional convolutional layers in equation 40 and have convolution kernel weights, v, and add biases, b. Naively, depthwise separable convolutions require fewer weight multiplications than traditional convolutions[1096, 1097]. However, extra batch normalization and serialization of one convolutional layer into depthwise and pointwise convolutional layers mean that depthwise separable convolutions and traditional convolutions have similar computing times[539, 1097].

Most DNNs developed for computer vision use fixed-size inputs. Although fixed input sizes are often regarded as an artificial constraint, it is similar to animalian vision where there is an effectively constant number of retinal rods and cones[1098–1100]. Typically, the most practical approach to handle arbitrary image shapes is to train a DNN with crops so that it can be tiled across images. In some cases, a combination of cropping, padding and interpolative resizing can also be used. To fully utilize unmodified variable size inputs, a simple is approach to train convolutional layers on variable size inputs. A pooling layer, such as global average pooling, can then be applied to fix output size before fully connected or other layers that might require fixed-size inputs. More involved approaches include spatial pyramid pooling[1101] or scale RNNs[1102]. Typical electron micrographs are much larger than 300×300, which often makes it unfeasible for electron microscopists with a few GPUs to train high-performance DNNs on full-size images. For comparison, Xception was trained on 300×300 images with 60 K80 GPUs for over one month.

The Fourier transform[1103], $\hat{f}(k_1,...,k_N)$, at an N-dimensional Fourier space vector, $\{k_1,...,k_N\}$, is related to a function, $f(x_1,...,x_N)$, of an N-dimensional signal domain vector, $\{x_1,...,x_N\}$, by

$$\hat{f}(k_1,...,k_N) = \left(\frac{|b|}{(2\pi)^{1-a}}\right)^{N/2} \int_{-\infty}^{\infty} ... \int_{-\infty}^{\infty} f(x_1,...,x_N)\exp(+ibk_1x_i + ... + ibk_Nx_N)\,\mathrm{d}x_1...\mathrm{d}x_N\,, \tag{44}$$

$$f(x_1,...,x_N) = \left(\frac{|b|}{(2\pi)^{1+a}}\right)^{N/2} \int_{-\infty}^{\infty} ... \int_{-\infty}^{\infty} \hat{f}(k_1,...,k_N)\exp(-ibk_1x_i - ... - ibk_Nx_N)\,\mathrm{d}k_1...\mathrm{d}k_N\,, \tag{45}$$

where $\pi = 3.141...$, and $i = (-1)^{1/2}$ is the imaginary number. Two parameters, a and b, can parameterize popular conventions that relate the Fourier and inverse Fourier transforms. Mathematica documentation nominates conventions[1104] for general applications (a,b), pure mathematics $(1,-1)$, classical physics $(-1,1)$, modern physics $(0,1)$, systems engineering $(1,-1)$, and signal processing $(0,2\pi)$. We observe that most electron microscopists follow the modern physics convention of $a=0$ and $b=1$; however, the choice of convention is arbitrary and does not matter if it is consistent within a project. For discrete functions, Fourier integrals are replaced with summations that are limited to the support of a function.

Discrete Fourier transforms of uniformly spaced inputs are often computed with a fast Fourier transform (FFT) algorithm, which can be parallelized for CPUs[1105] or GPUs[65, 1106–1108]. Typically, the speedup of FFTs on GPUs over CPUs is higher for larger signals[1109, 1110]. Most popular FFTs are based on the Cooley-Turkey algorithm[1111, 1112], which recursively divides FFTs into smaller FFTs. We observe that some electron microscopists consider FFTs to be limited to radix-2 signals that can be recursively halved; however, FFTs can use any combination of factors for the sizes of recursively smaller FFTs. For example, clFFT[1113] FFT algorithms support signal sizes that are any sum of powers of 2, 3, 5, 7, 11 and 13.

Convolution theorems can decrease computation by enabling convolution in the Fourier domain[1114]. To ease notation, we denote the Fourier transform of a signal, $\mathbf{I}$, by $\text{FT}(\mathbf{I})$, and the inverse Fourier transform by $\text{FT}^{-1}(\mathbf{I})$. Following, the convolution theorems for two signals, $\mathbf{I}_1$ and $\mathbf{I}_2$, are[1115]

$$\text{FT}(\mathbf{I}_1 * \mathbf{I}_2) = \text{FT}(\mathbf{I}_1)\cdot\text{FT}(\mathbf{I}_2)\,, \tag{46}$$

$$\text{FT}(\mathbf{I}_1 \cdot \mathbf{I}_2) = \text{FT}(\mathbf{I}_1) * \text{FT}(\mathbf{I}_2)\,, \tag{47}$$

where the signals can be feature channels and convolutional kernels. Fourier domain convolutions, $\mathbf{I}_1 * \mathbf{I}_2 = \text{FT}^{-1}\left(\text{FT}(\mathbf{I}_1)\cdot\text{FT}(\mathbf{I}_2\right.$ are increasingly efficient, relative to signal domain convolutions, as kernel and image sizes increase[1114]. Indeed, Fourier domain convolutions are exploited to enable faster training with large kernels in Fourier CNNs[1114, 1116]. However, Fourier CNNs are rare as most researchers use small 3×3 kernels, following University of Oxford Visual Geometry Group (VGG) CNNs[1117].

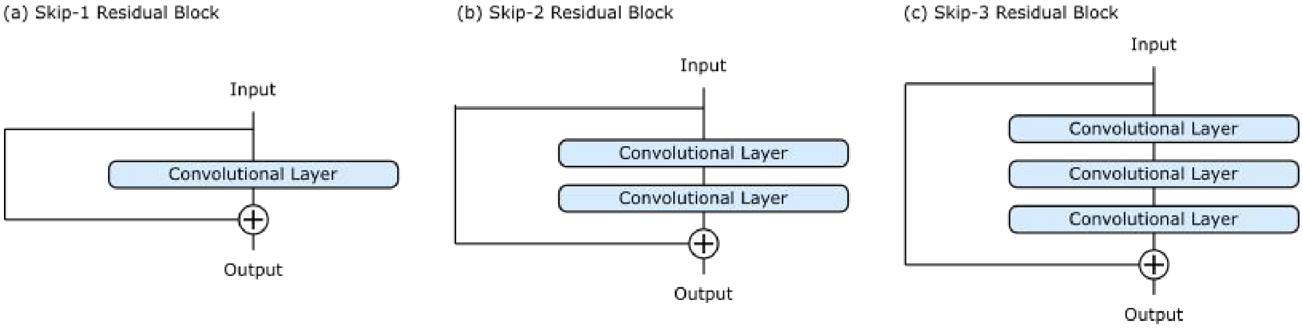


Figure 10. Residual blocks where a) one, b) two, and c) three convolutional layers are skipped. Typically, convolutional layers are followed by batch normalization then activation.

4.4 Skip Connections

Residual connections[1080] add a signal after skipping ANN layers, similar to cortical skip connections[1118,1119]. Residuals improve DNN performance by preserving gradient norms during backpropagation[537,1120] and avoiding bad local minima[1121] by smoothing DNN loss landscapes[1122]. In practice, residuals enable DNNs to behave like an ensemble of shallow networks[1123] that learn to iteratively estimate outputs[1124]. Mathematically, a residual layer learns parameters, $\mathbf{w}_l$, of a perturbative function, $f_l(\mathbf{x}_l, \mathbf{w}_l)$, that maps a signal, $\mathbf{x}_l$, at depth l to depth $l+1$,

$$\mathbf{x}_{l+1} = \mathbf{x}_l + f_l(\mathbf{x}_l, \mathbf{w}_l) \,. \tag{48}$$

Residuals were developed for CNNs[1080], and examples of residual connections that skip one, two and three convolutional layers are shown in figure 10. Nonetheless, residuals are also used in MLPs[1125] and RNNs[1126–1128]. Representational capacity of perturbative functions increases as the number of skipped layers increases. As result, most residuals skip two or three layers. Skipping one layer rarely improves performance due to its low representational capacity[1080].

There are a range of residual connection variants that can improve performance. For example, highway networks[1129,1130] apply a gating function to skip connections, and dense networks[1131–1133] use a high number of residual connections from multiple layers. Another example is applying a 1×1 convolutional layer to x_l before addition[539,1080] where $f_l(x_l, w_l)$ spatially resizes or changes numbers of feature channels. However, resizing with norm-preserving convolutional layers[1120] before residual blocks can often improve performance. Finally, long additive[1134] residuals that connect DNN inputs to outputs are often applied to DNNs that learn perturbative functions.

A limitation of preserving signal information with residuals[1135,1136] is that residuals make DNNs learn perturbative functions, which can limit accuracy of DNNs that learn non-perturbative functions if they do not have many layers. Feature channel concatenation is an alternative approach that not perturbative, and that supports combination of layers with different numbers of feature channels. In encoder-decoders, a typical example is concatenating features computed near the start with layers near the end to help resolve output features[305,306,308,316]. Concatenation can also combine embeddings of different[1137,1138] or variants of[366] input features by multiple DNNs. Finally, peephole connections in RNNs can improve performance by using concatenation to combine cell state information with other cell inputs[1139,1140].

5 Architecture

There is a high variety of ANN architectures[4–7] that are trained to minimize losses for a range of applications. Many of the most popular ANNs are also the simplest, and information about them is readily available. For example, encoder-decoder[305–308,502–504] or classifier[272] ANNs usually consist of single feedforward sequences of layers that map inputs to outputs. This section introduces more advanced ANNs used in electron microscopy, including actor-critics, GANs, RNNs, and variational autoencoders (VAEs). These ANNs share weights between layers or consist of multiple subnetworks. Other notable architectures include recursive CNNs[1078,1079], Network-in-Networks[1141] (NiNs), and transformers[1142,1143]. Although they will not be detailed in this review, their references may be good starting points for research.

5.1 Actor-Critic

Most ANNs are trained by gradient descent using backpropagated gradients of a differentiable loss function cf. section 6.1. However, some losses are not differentiable. Examples include losses of actors directing their vision[1144,1145], and playing competitive[24] or score-based[1146,1147] computer games. To overcome this limitation, a critic[1148] can be trained to predict

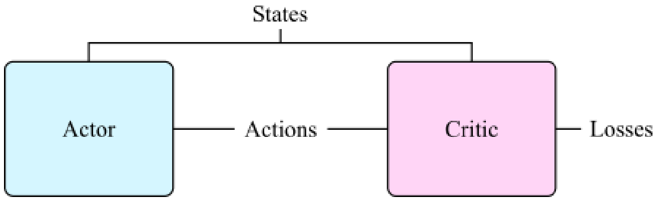


Figure 11. Actor-critic architecture. An actor outputs actions based on input states. A critic then evaluates action-state pairs to predict losses.

differentiable losses from action and state information, as shown in figure 11. If the critic does not depend on states, it is a surrogate loss function[1149,1150]. Surrogates are often fully trained before actor optimization, whereas critics that depend on actor-state pairs are often trained alongside actors to minimize the impact of catastrophic forgetting[1151] by adapting to changing actor policies and experiences. Alternatively, critics can be trained with features output by intermediate layers of actors to generate synthetic gradients for backpropagation[1152].

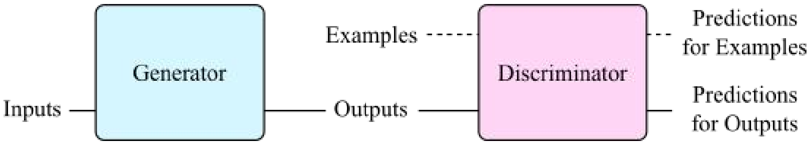


Figure 12. Generative adversarial network architecture. A generator learns to produce outputs that look realistic to a discriminator, which learns to predict whether examples are real or generated.

5.2 Generative Adversarial Network

Generative adversarial networks[197–200] (GANs) consist of generator and discriminator subnetworks that play an adversarial game, as shown in figure 12. Generators learn to generate outputs that look realistic to discriminators, whereas discriminators learn to predict whether examples are real or generated. Most GANs are developed to generate visual media with realistic characteristics. For example, partial STEM images infilled with a GAN are less blurry than images infilled with a non-adversarial generator trained to minimize MSEs[201] cf. figure 2. Alternatively, computationally inexpensive loss functions designed by humans, such as structural similarity index measures[1153] (SSIMs) and Sobel losses[231], can improve generated output realism. However, it follows from the universal approximator theorems[37–45] that training with ANN discriminators can often yield more realistic outputs.

There are many popular GAN loss functions and regularization mechanisms[1154–1158]. Traditionally, GANs were trained to minimize logarithmic discriminator, D, and generator, G, losses[1159],

$$L_D = -\log D(\mathbf{x}) - \log(1 - D(G(\mathbf{z}))) \,, \tag{49}$$
$$L_G = \log(1 - D(G(\mathbf{z}))) \,, \tag{50}$$

where $\mathbf{z}$ are generator inputs, $G(\mathbf{z})$ are generated outputs, and $\mathbf{x}$ are example outputs. Discriminators predict labels, $D(\mathbf{x})$ and $D(G(\mathbf{z}))$, where target labels are 0 and 1 for generated and real examples, respectively. Limitedly, logarithmic losses are numerically unstable for $D(\mathbf{x}) \to 0$ or $D(G(\mathbf{z})) \to 1$, as the denominator, $f(x)$, in $\partial_x \log f(x) = \partial_x f(x)/f(x)$ vanishes. In addition, discriminators must be limited to $D(\mathbf{x}) > 0$ and $D(G(\mathbf{z})) < 1$, so that logarithms are not complex. To avoid these issues, we recommend training discriminators with squared difference losses[1160,1161],

$$L_D = (D(\mathbf{x}) - 1)^2 + D(G(\mathbf{z}))^2 \,, \tag{51}$$
$$L_G = (D(G(\mathbf{z})) - 1)^2 \,. \tag{52}$$

However, there are a variety of other alternatives to logarithmic loss functions that are also effective[1154,1155].

A variety of methods have been developed to improve GAN training[995,1162]. The most common issues are catastrophic forgetting[1151] of previous learning, and mode collapse[1163] where generators only output examples for a subset of a target domain. Mode collapse often follows discriminators becoming Lipschitz discontinuous. Wasserstein GANs[1164] avoid mode collapse by clipping trainable variables, albeit often at the cost of 5-10 discriminator training iterations per generator training iteration. Alternatively, Lipschitz continuity can be imposed by adding a gradient penalty[1165] to GAN losses, such as differences of L2 norms of discriminator gradients from unity,

$$
\begin{aligned}
\tilde{x} &= G(\mathbf{z})\,, && (53)\\
\hat{\mathbf{x}} &= \varepsilon\mathbf{x} + (1-\varepsilon)\tilde{\mathbf{x}}\,, && (54)\\
L_D &= D(\tilde{\mathbf{x}}) - D(\mathbf{x}) + \lambda(||\partial_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})||_2 - 1)^2\,, && (55)\\
L_G &= -D(G(\mathbf{z}))\,, && (56)
\end{aligned}
$$

where $\varepsilon \in [0,1]$ is a uniform random variate, λ weights the gradient penalty, and $\tilde{\mathbf{x}}$ is an attempt to generate x. However, using a gradient penalty introduces additional gradient backpropagation that increases discriminator training time. There are also a variety of computationally inexpensive tricks that can improve training, such as adding noise to labels[995,1075,1166] or balancing discriminator and generator learning rates[349]. These tricks can help to avoid discontinuities in discriminator output distributions that can lead to mode collapse; however, we observe that these tricks do not reliably stabilize GAN training.

Instead, we observe that spectral normalization[1017] reliably stabilizes GAN discriminator training in our electron microscopy research[201,202,349]. Spectral normalization controls Lipschitz constants of discriminators by fixing the spectral norms of their weights, as introduced in section 4.2. Advantages of spectral normalization include implementations based on the power iteration method[1023,1024] being computationally inexpensive, not adding a regularizing loss function that could detrimentally compete[1167,1168] with discrimination losses, and being effective with one discriminator training iterations per generator training iteration[1017,1169]. Spectral normalization is popular in GANs for high-resolution image synthesis, where it is also applied in generators to stabilize training[1170].

There are a variety of GAN architectures[1171]. For high-resolution image synthesis, computation can be decreased by training multiple discriminators to examine image patches at different scales[201,1172]. For domain translation characterized by textural differences, a cyclic GAN[1004,1173] consisting of two GANs can map from one domain to the other and vice versa. Alternatively, two GANs can share intermediate layers to translate inputs via a shared embedding domain[1174]. Cyclic GANs can also be combined with a siamese network[279–281] for domain translation beyond textural differences[1175]. Finally, discriminators can introduce auxiliary losses to train DNNs to generalize to examples from unseen domains[1176–1178].

5.3 Recurrent Neural Network

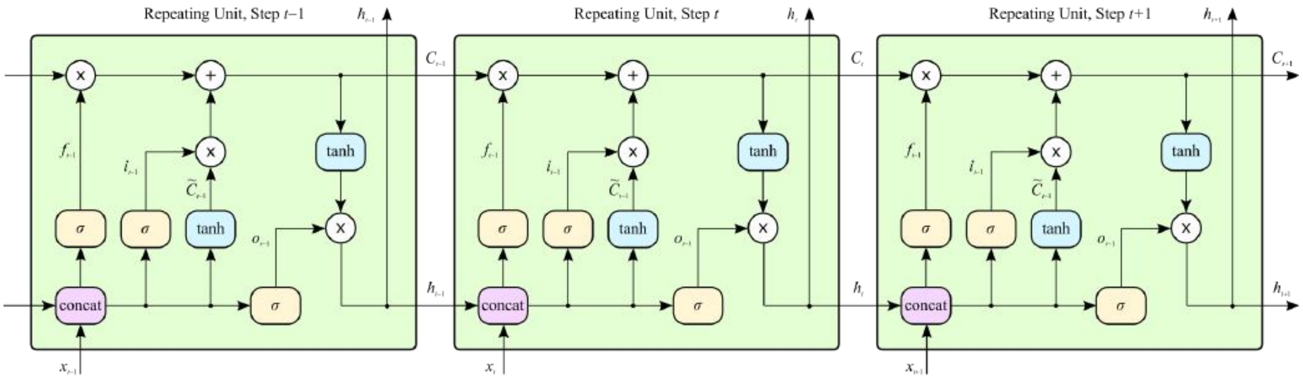
Recurrent neural networks[531–536] reuse an ANN cell to process each step of a sequence. Most RNNs learn to model long-term dependencies by gradient backpropagation through time[1179] (BPTT). The ability of RNNs to utilize past experiences enables them to model partially observed and variable length Markov decision processes[1180,1181] (MDPs). Applications of RNNs include directing vision[1144,1145], image captioning[1182,1183], language translation[1184], medicine[77], natural language processing[1185,1186], playing computer games[24], text classification[1055], and traffic forecasting[1187]. Many RNNs are combined with CNNs to embed visual media[1145] or words[1188,1189], or to process RNN outputs[1190,1191]. RNNs can also be combined with MLPs[1144], or text embeddings[1192] such as BERT[1192,1193], continuous bag-of-words[1194–1196] (CBOW), doc2vec[1197,1198], GloVe[1199], and word2vec[1194,1200].

The most popular RNNs consist of long short-term memory[1201–1204] (LSTM) cells or gated recurrent units[1202,1205–1207] (GRUs). LSTMs and GRUs are popular as they solve the vanishing gradient problem[537,1208,1209] and have consistently high performance[1210–1215]. Their architectures are shown in figure 13. At step t, an LSTM outputs a hidden state, h_t, and cell state, C_t, given by

$$
\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{w}_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f)\,, && (57)\\
\mathbf{i}_t &= \sigma(\mathbf{w}_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i)\,, && (58)\\
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{w}_C \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_C)\,, && (59)\\
\mathbf{C}_t &= \mathbf{f}_t \mathbf{C}_{t-1} + \mathbf{i}_t \tilde{\mathbf{C}}_t\,, && (60)\\
\mathbf{o}_t &= \sigma(\mathbf{w}_o \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o)\,, && (61)\\
\mathbf{h}_t &= \mathbf{o}_t \tanh(\mathbf{C}_t)\,, && (62)
\end{aligned}
$$

where $\mathbf{C}_{t-1}$ is the previous cell state, $\mathbf{h}_{t-1}$ is the previous hidden state, $\mathbf{x}_t$ is the step input, and σ is a logistic sigmoid function of equation 10a, $[\mathbf{x}, \mathbf{y}]$ is the concatenation of $\mathbf{x}$ and $\mathbf{y}$ channels, and $(\mathbf{w}_f, \mathbf{b}_f)$, $(\mathbf{w}_i, \mathbf{b}_i)$, $(\mathbf{w}_C, \mathbf{b}_C)$ and $(\mathbf{w}_o, \mathbf{b}_o)$ are pairs of weights

(a): Long Short-Term Memory



(b): Gated Recurrent Unit

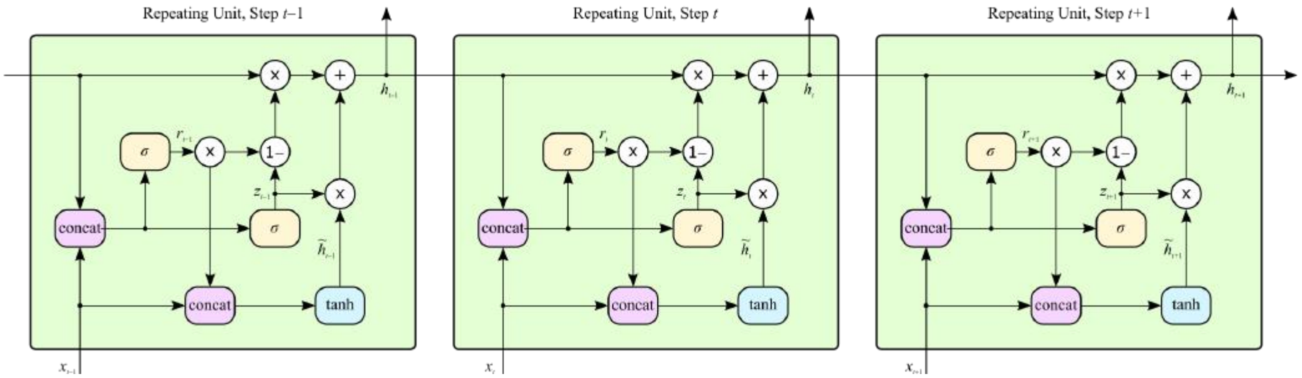


Figure 13. Architectures of recurrent neural networks with a) long short-term memory (LSTM) cells, and b) gated recurrent units (GRUs).

and biases. A GRU performs fewer computations than an LSTM and does not have separate cell and hidden states,

$$\mathbf{z}_t = \sigma(\mathbf{w}_z \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_z)\,, \tag{63}$$

$$\mathbf{r}_t = \sigma(\mathbf{w}_r \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_r)\,, \tag{64}$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{w}_h \cdot [\mathbf{r}_t \mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_h)\,, \tag{65}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t)\mathbf{h}_{t-1} + \mathbf{z}_t \tilde{\mathbf{h}}_t\,, \tag{66}$$

where $(\mathbf{w}_z, \mathbf{b}_z)$, $(\mathbf{w}_r, \mathbf{b}_r)$, and $(\mathbf{w}_h, \mathbf{b}_h)$ are pairs of weights and biases. Minimal gated units (MGUs) can further reduce computation[1216]. A large-scale analysis of RNN architectures for language translation found that LSTMs consistently outperform GRUs[1210]. GRUs struggle with simple languages that are learnable by LSTMs as the combined hidden and cell states of GRUs make it more difficult for GRUs to perform unbounded counting[1214]. However, further investigations found that GRUs can outperform LSTMs on tasks other than language translation[1211], and that GRUs can outperform LSTMs on some datasets[1212,1213,1217]. Overall, LSTM performance is usually comparable to that of GRUs.

There are a variety of alternatives to LSTM and GRUs. Examples include continuous time RNNs[1218–1222] (CTRNNs), Elman[1223] and Jordan[1224] networks, independently RNNs[1225] (IndRNNs), Hopfield networks[1226], recurrent MLPs[1227] (RMLPs). However, none of the variants offer consistent performance benefits over LSTMs for general sequence modelling. Similarly, augmenting LSTMs with additional connections, such as peepholes[1139,1140] and projection layers[1228], does not consistently improve performance. For electron microscopy, we recommend defaulting to LSTMs as we observe that their performance is more consistently high than performance of other RNNs. However, LSTM and GRU performance is often comparable, so GRUs are also a good choice to reduce computation.

There are a variety of architectures based on RNNs. Popular examples include deep RNNs[1229] that stack RNN cells to increase representational ability, bidirectional RNNs[1230–1233] that process sequences both forwards and in reverse to improve input utilization, and using separate encoder and decoder subnetworks[1205,1234] to embed inputs and generate outputs. Hierarchical RNNs[1235–1239] are more complex models that stack RNNs to efficiently exploit hierarchical sequence information, and include multiple timescale RNNs[1240, 1241] (MTRNNs) that operate at multiple sequence length scales. Finally, RNNs can be augmented with additional functionality to enable new capabilities. For example, attention[1182, 1242–1244] mechanisms can enable more efficient input utilization. Further, creating a neural Turing machine (NTMs) by augmenting a RNN with dynamic external memory[1245, 1246] can make it easier for an agent to solve dynamic graphs.

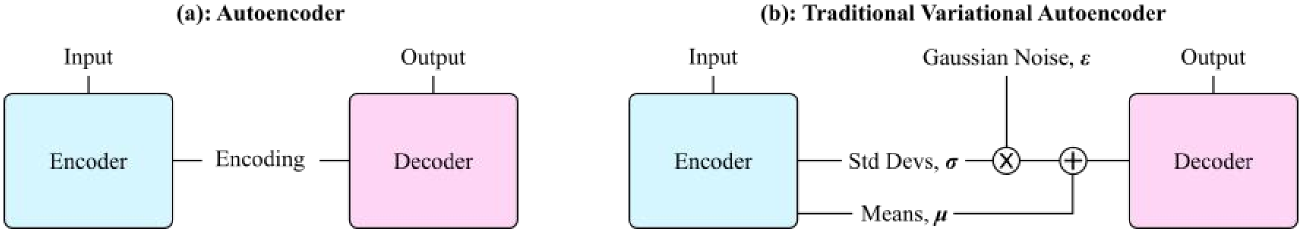


Figure 14. Architectures of autoencoders where an encoder maps an input to a latent space and a decoder learns to reconstruct the input from the latent space. a) An autoencoder encodes an input in a deterministic latent space, whereas a b) traditional variational autoencoder encodes an input as means, μ, and standard deviations, σ, of Gaussian multivariates, $\mu + \sigma \cdot \varepsilon$, where ε is a standard normal multivariate.

5.4 Autoencoders

Autoencoders[1247–1249] (AEs) learn to efficiently encode inputs, $\mathbf{I}$, without supervision. An AE consists of a encoder, E, and decoder, D, as shown in figure 14a. Most encoders and decoders are jointly trained[1250] to restore inputs from encodings, $E(\mathbf{I})$, to minimize a MSE loss,

$$L_{\text{AE}} = \text{MSE}(D(E(\mathbf{I})), \mathbf{I}) , \tag{67}$$

by gradient descent. In practice, DNN encoders and decoders yield better compression[1248] than linear techniques, such as principal component analysis[1251] (PCA), or shallow ANNs. Indeed, deep AEs can outperform JPEG image compression[1252]. Denoising autoencoders[1253–1257] (DAEs) are a popular AE variant that can learn to remove artefacts by artificially corrupting inputs inside encoders. Alternatively, contractive autoencoders[1258, 1259] (CAEs) can decrease sensitivity to input values by adding a loss to minimize gradients w.r.t. inputs. Most DNNs that improve electron micrograph signal-to-noise are DAEs.

In general, semantics of AE outputs are pathological functions of encodings. To generate outputs with well-behaved semantics, traditional VAEs[969, 1260, 1261] learn to encode means, μ, and standard deviations, σ, of Gaussian multivariates. Meanwhile, decoders learn to reconstruct inputs from sampled multivariates, $\mu + \sigma \cdot \varepsilon$, where ε is a standard normal multivariate. Traditional VAE architecture is shown in figure 14b. Usually, VAE encodings are regularized by adding Kullback-Leibler (KL) divergence of encodings from standard multinormals to an AE loss function,

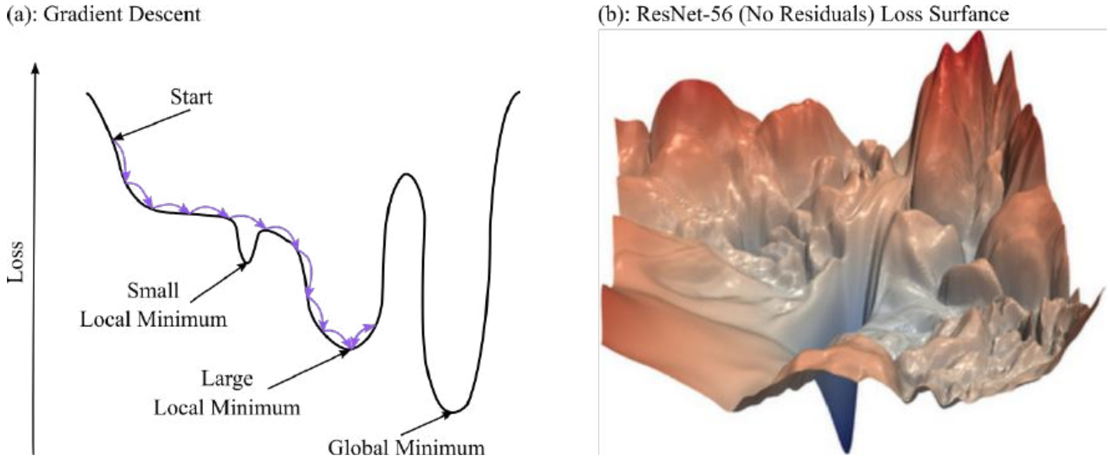
$$L_{\text{VAE}} = \text{MSE}(D(\mu + \sigma \cdot \varepsilon), \mathbf{I}) + \frac{\lambda_{\text{KL}}}{2Bu} \sum_{i=1}^{B} \sum_{j=1}^{u} \mu_{ij}^2 + \sigma_{ij}^2 - \log(\sigma_{ij}^2) - 1 , \tag{68}$$

where λ_{KL} weights the contribution of the KL divergence loss for a batch size of B, and a latent space with u degrees of freedom. However, variants of Gaussian regularization can improve clustering[231], and sparse autoencoders[1262–1265] (SAEs) that regularize encoding sparsity can encode more meaningful features. To generate realistic outputs, a VAE can be combined with a GAN to create a VAE-GAN[1266–1268]. Adding a loss to minimize differences between gradients of generated and target outputs is computationally inexpensive alternative that can generate realistic outputs for some applications[231].

A popular application of VAEs is data clustering. For example, VAEs can encode hash tables[1269–1273] for search engines, and we use VAEs as the basis of our electron micrograph search engines[231]. Encoding clusters visualized by tSNE can be labelled to classify data[231], and encoding deviations from clusters can be used for anomaly detection[1274–1278]. In addition, learning encodings with well-behaved semantics enables encodings to be used for semantic manipulation[1278, 1279]. Finally, VAEs can be used as generative models to create synthetic populations[1280, 1281], develop new chemicals[1282–1285], and synthesize underrepresented data to reduce imbalanced learning[1286].

6 Optimization

Training, testing, deployment and maintenance of machine learning systems is often time-consuming and expensive[1287–1290]. The first step is usually preparing training data and setting up data pipelines for ANN training and evaluation. Typically, ANN parameters are randomly initialized for optimization by gradient descent, possibly as part of an automatic machine learning algorithm. Reinforcement learning is a special optimization case where the loss is a discounted future reward. During training, ANN components are often regularized to stabilize training, accelerate convergence, or improve performance. Finally, trained models can be streamlined for efficient deployment. This section introduces each step. We find that electron microscopists can be apprehensive about robustness and interpretability of ANNs, so we also provide subsections on model evaluation and interpretation.


(a): Gradient Descent

(b): ResNet-56 (No Residuals) Loss Surface

Figure 15. Gradient descent. a) Arrows depict steps across one dimension of a loss landscape as a model is optimized by gradient descent. In this example, the optimizer traverses a small local minimum; however, it then gets trapped in a larger sub-optimal local minimum, rather than reaching the global minimum. b) Experimental DNN loss surface for two random directions in parameter space showing many local minima[1122]. The image in part b) is reproduced with permission under an MIT license[1291].

Algorithm 1 Optimization by gradient descent.

```
Initialize a model, $f(\mathbf{x})$, with trainable parameters, $\theta_1$.
for training step $t = 1, T$ do
    Forwards propagate a randomly sampled batch of inputs, $\mathbf{x}$, through the model to compute outputs, $\mathbf{y} = f(\mathbf{x})$.
    Compute loss, $L_t$, for outputs.
    Use the differentiation chain rule[1292] to backpropagate gradients of the loss to trainable parameters, $\theta_{t-1}$.
    Apply an optimizer to the gradients to update $\theta_{t-1}$ to $\theta_t$.
end for
```

6.1 Gradient Descent

Most ANNs are iteratively trained by gradient descent[465, 1303–1307], as described by algorithm 1 and shown in figure 15. To minimize computation, results at intermediate stages of forward propagation, where inputs are mapped to outputs, are often stored in memory. Storing the forwards pass in memory enables backpropagation memoization by sequentially computing gradients w.r.t. trainable parameters. To reduce memory costs for large ANNs, a subset of intermediate forwards pass results can be saved as starting points to recompute other stages during backpropagation[1308, 1309]. Alternatively, forward pass computations can be split across multiple devices[1310]. Optimization by gradient descent plausibly models learning in some biological systems[1311]. However, gradient descent is not generally an accurate model of biological learning[1312–1314].

There are many popular gradient descent optimizers for deep learning[1303–1305]. Update rules for eight popular optimizers are summarized in figure 1. Other optimizers include AdaBound[1315], AMSBound[1315], AMSGrad[1316], Lookahead[1317], NADAM[1318], Nostalgic Adam[1319], Power Gradient Descent[1320], Rectified ADAM[1321] (RADAM), and trainable optimizers[1322–1326]. Gradient descent is effective in the high-dimensional optimization spaces of overparameterized ANNs[1327] as the probability of getting trapped in a sub-optimal local minima decreases as the number of dimensions increases. The simplest optimizer is "vanilla" stochastic gradient descent (SGD), where a trainable parameter perturbation, $\Delta\theta_t = \theta_t - \theta_{t-1}$, is the product of a learning rate, η, and derivative of a loss, L_t, w.r.t. the trainable parameter, $\partial_\theta L_t$. However, vanilla SGD convergence is often limited by

Vanilla SGD[1293,1294] $[\eta]$

$$\theta_t = \theta_{t-1} - \eta \partial_\theta L_t \tag{69}$$

Momentum[1295] $[\eta, \gamma]$

$$v_t = \gamma v_{t-1} + \eta \partial_\theta L_t \tag{70}$$

$$\theta_t = \theta_{t-1} - v_t \tag{71}$$

Nesterov momentum[1296–1298] $[\eta, \gamma]$

$$\phi = \theta_{t-1} + \eta \gamma v_{t-1} \tag{72}$$

$$v_t = \gamma v_{t-1} + \partial_\theta L_t \tag{73}$$

$$\theta_t = \phi - \eta v_t (1 + \gamma) \tag{74}$$

Quasi-hyperbolic momentum[1299] $[\eta, \beta, \nu]$

$$g_t = \beta g_{t-1} + (1 - \beta) \partial_\theta L_t \tag{75}$$

$$\theta_t = \theta_{t-1} - \eta (\nu g_t + (1 - \nu) \partial_\theta L_t) \tag{76}$$

AggMo[1300] $[\eta, \beta^{(1)}, \ldots, \beta^{(K)}]$

$$v_t^{(i)} = \beta^{(i)} v_{t-1}^{(i)} - (\partial_\theta L_t) \tag{77}$$

$$\theta_t = \theta_{t-1} + \frac{\eta}{K} \sum_{i=1}^{K} v_t^{(i)} \tag{78}$$

RMSProp[1301] $[\eta, \beta, \varepsilon]$

$$v_t = \beta v_{t-1} + (1 - \beta)(\partial_\theta L_t)^2 \tag{79}$$

$$\theta_t = \theta_{t-1} - \frac{\eta}{(v_t + \varepsilon)^{1/2}} \partial_\theta L_t \tag{80}$$

ADAM[1302] $[\eta, \beta_1, \beta_2, \varepsilon]$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \partial_\theta L_t \tag{81}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\partial_\theta L_t)^2 \tag{82}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{83}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{84}$$

$$\theta_t = \theta_{t-1} - \frac{\eta}{\hat{v}_t^{1/2} + \varepsilon} \hat{m}_t \tag{85}$$

AdaMax[1302] $[\eta, \beta_1, \beta_2]$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \partial_\theta L_t \tag{86}$$

$$u_t = \max(\beta_2 u_{t-1}, |\partial_\theta L_t|) \tag{87}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{88}$$

$$\theta_t = \theta_{t-1} - \frac{\eta}{u_t} \hat{m}_t \tag{89}$$

Algorithms 1. Update rules of various gradient descent optimizers for a trainable parameter, θ_t, at iteration t, gradients of losses w.r.t. the parameter, $\partial_\theta L_t$, and learning rate, η. Hyperparameters are listed in square brackets.

unstable parameter oscillations as it a low-order local optimization method[1328]. Further, vanilla SGD has no mechanism to adapt to varying gradient sizes, which vary effective learning rates as $\Delta\theta \propto \partial_\theta L_t$.

To accelerate convergence, many optimizers introduce a momentum term that weights an average of gradients with past gradients[1296,1329,1330]. Momentum-based optimizers in figure 1 are momentum, Nesterov momentum[1296,1297], quasi-hyperbolic momentum[1299], AggMo[1300], ADAM[1302], and AdaMax[1302]. To standardize effective learning rates for every layer, adaptive optimizers normalize updates based on an average of past gradient sizes. Adaptive optimizers in figure 1 are RMSProp[1301], ADAM[1302], and AdaMax[1302], which usually result in faster convergence and higher accuracy than other optimizers[1331,1332]. However, adaptive optimizers can be outperformed by vanilla SGD due to overfitting[1333], so some researchers adapt adaptive learning rates to their variance[1321] or transition from adaptive optimization to vanilla SGD as training progresses[1315]. For electron microscopy we recommend adaptive optimization with Nadam[1318], which combines ADAM with Nesterov momentum, as it is well-established and a comparative analysis of select gradient descent optimizers found that it often achieves higher performance than other popular optimizers[1334]. Limitingly, most adaptive optimizers slowly adapt to changing gradient sizes e.g. a default value for ADAM β_2 is 0.999[1302]. To prevent learning being destabilized by spikes in gradient sizes, adaptive optimizers can be combined with adaptive learning rate[261,1315] or gradient[1208,1335,1336] clipping.

For non-adaptive optimizers, effective learning rates are likely to vary due to varying magnitudes of gradients w.r.t. trainable parameters. Similarly, learning by biological neurons varies as stimuli usually activate a subset of neurons[1337]. However, all neuron outputs are usually computed for ANNs. Thus, not effectively using all weights to inform decisions is computational inefficient. Further, inefficient weight updates can limit representation capacity, slow convergence, and decrease training stability. A typical example is effective learning rates varying between layers. Following the chain rule, gradients backpropagated to the ith layer of a DNN from its start are

$$\frac{\partial L_t}{\partial \mathbf{x}_i} = \left(\prod_{l=i}^{L-1} \frac{\partial \mathbf{x}_{l+1}}{\partial \mathbf{x}_l} \right) \frac{\partial L_t}{\partial \mathbf{x}_L}, \tag{90}$$

for a DNN with L layers. Vanishing gradients[537,1208,1209] occur when many layers have $\partial x_{l+1}/\partial x_l \ll 1$. For example, DNNs with logistic sigmoid activations often exhibit vanishing gradients as their maximum gradient is $1/4$ cf. equation 10b. Similarly, exploding gradients[537, 1208, 1209] occur when many layers have $\partial x_{l+1}/\partial x_l \gg 1$. Adaptive optimizers alleviate vanishing and exploding gradients by dividing gradients by their expected sizes. Nevertheless, it is essential to combine adaptive optimizers with appropriate initialization and architecture to avoid numerical instability.

Optimizers have a myriad of hyperparameters to be initialized and varied throughout training to optimize performance[1338] cf. figure 1. For example, stepwise exponentially decayed learning rates are often theoretically optimal[1339]. There are also various heuristics that are often effective, such as using a DEMON decay schedule for an ADAM first moment of the momentum decay rate[1340],

$$\beta_1 = \frac{1 - t/T}{(1 - \beta_{\text{init}}) + \beta_{\text{init}}(1 - t/T)} \beta_{\text{init}} \,, \tag{91}$$

where β_{init} is the initial value of β_1, t is the iteration number, and T is the final iteration number. Developers often optimize ANN hyperparameters by experimenting with a range of heuristic values. Hyperparameter optimization algorithms[1341–1346] can automate optimizer hyperparameter selection. However, automatic hyperparameter optimizers may not yield sufficient performance improvements relative to well-established heuristics to justify their use, especially in initial stages of development.

Alternatives to gradient descent[1347] are rarely used for parameter optimization as they are not known to consistently improve upon gradient descent. For example, simulated annealing[1348, 1349] has been applied to CNN training[1350, 1351], and can be augmented with momentum to accelerate convergence in deep learning[1352]. Simulated annealing can also augment gradient descent to improve performance[1353]. Other approaches include evolutionary[1354, 1355] and genetic[1356, 1357] algorithms, which can be a competitive alternative to deep reinforcement learning where convergence is slow[1358]. Indeed, recent genetic algorithms have outperformed a popular deep reinforcement learning algorithm[1359]. Another direction is to augment genetic algorithms with ANNs to accelerate convergence[1360–1363]. Other alternatives to backpropagation include direct search[1364], the Moore-Penrose Pseudo Inverse[1365]; particle swarm optimization[1366–1369] (PSO); and echo-state networks[1370–1372] (ESNs) and extreme learning machines[1373–1379] (ELMs), where some randomly initialized weights are never updated.

6.2 Reinforcement Learning

Reinforcement learning[1380–1386] (RL) is where a machine learning system, or "actor", is trained to perform a sequence of actions. Applications include autonomous driving[1387–1389], communications network control[1390, 1391], energy and environmental management[1392, 1393], playing games[24–29, 1146, 1394], and robotic manipulation[1395, 1396]. To optimize a MDP[1180, 1181], a discounted future reward, Q_t, at step t in a MDP with T steps is usually calculated from step rewards, r_t, with Bellman's equation,

$$Q_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} \,, \tag{92}$$

where $\gamma \in [0, 1)$ discounts future step rewards. To be clear, multiplying Q_t by -1 yields a loss that can be minimized using the methods in section 6.1.

In practice, many MDPs are partially observed or have non-differentiable losses that may make it difficult to learn a good policy from individual observations. However, RNNs can often learn a model of their environments from sequences of observations[1147]. Alternatively, FNNs can be trained with groups of observations that contain more information than individual observations[1146, 1394]. If losses are not differentiable, a critic can learn to predict differentiable losses for actor training cf. section 5.1. Alternatively, actions can be sampled from a differentiable probability distribution[1144, 1397] as training losses given by products of losses and sampling probabilities are differentiable. There are also a variety of alternatives to gradient descent introduced at the end of section 6.1 that do not require differentiable loss functions.

There are a variety of exploration strategies for RL[1398, 1399]. Adding Ornstein-Uhlenbeck[1400] (OU) noise to actions is effective for continuous control tasks optimized by deep deterministic policy gradients[1146] (DDPG) or recurrent deterministic policy gradients[1147] (RDPG) RL algorithms. Adding Gaussian noise achieves similar performance for optimization by TD3[1401] or D4PG[1402] RL algorithms. However, a comparison of OU and Gaussian noise across a variety of tasks[1403] found that OU noise usually achieves similar performance to or outperforms Gaussian noise. Similarly, exploration can be induced by adding noise to ANN parameters[1404, 1405]. Other approaches to exploration include rewarding actors for increasing action entropy[1405–1407] and intrinsic motivation[1408–1410], where ANNs are incentified to explore actions that they are unsure about.

RL algorithms are often partitioned into online learning[1411, 1412], where training data is used as it is acquired; and offline learning[1413, 1414], where a static training dataset has already been acquired. However, many algorithms operate in an intermediate regime, where data collected with an online policy is stored in an experience replay[1415–1417] buffer for offline learning. Training data is often sampled at random from a replay. However, prioritizing the replay of data with high losses[993] or data that results in high policy improvements[992] often improves actor performance. A default replay buffer size of around 10^6 examples is often used; however, training is sensitive to replay buffer size[1418]. If the replay is too small, changes in actor policy may destabilize training; whereas if the replay is too large, convergence may be slowed by delays before the actor learns from policy changes.

6.3 Automatic Machine Learning

There are a variety of automatic machine learning[1419–1423] (AutoML) algorithms that can create and optimize ANN architectures and learning policies for a dataset of input and target output pairs. Most AutoML algorithms are based on RL or evolutionary algorithms. Examples of AutoML algorithms include AdaNet[1424,1425], Auto-DeepLab[1426], AutoGAN[1427], Auto-Keras[1428], auto-sklearn[1429], DARTS+[1430], EvoCNN[271], H2O[1431], Ludwig[1432], MENNDL[1433,1434], NASBOT[1435], XNAS[1436], and others[1437–1441]. AutoML is becoming increasingly popular as it can achieve higher performance than human developers[1077,1442] and enables human developer time to be traded for potentially cheaper computer time. Nevertheless, AutoML is currently limited to established ANN architectures and learning policies. Following, we recommend that researchers either focus on novel ANN architectures and learning policies or developing ANNs for novel applications.

6.4 Initialization

How ANN trainable parameters are initialized[537,1443] is related to model capacity[1444]. Further, initializing parameters with values that are too small or large can cause slow learning or divergence[537]. Careful initialization can also prevent training by gradient descent being destabilized by vanishing or exploding gradients[537,1208,1209], or high variance of length scales across layers[537]. Finally, careful initialization can enable momentum to accelerate convergence and improve performance[1296]. Most trainable parameters are multiplicative weights or additive biases. Initializing parameters with constant values would result in every parameter in a layer receiving the same updates by gradient descent, reducing model capacity. Thus, weights are often randomly initialized. Following, biases are often initialized with constant values due to symmetry breaking by the weights.

Consider the projection of n_in inputs, $\mathbf{x}^\text{input} = \{x_1^\text{input}, ..., x_{n_\text{in}}^\text{input}\}$, to n_out outputs, $\mathbf{x}^\text{output} = \{x_1^\text{output}, ..., x_{n_\text{out}}^\text{output}\}$, by an $n_\text{in} \times n_\text{out}$ weight matrix, $\mathbf{w}$. The expected variance of an output element is[1443]

$$\text{Var}(\mathbf{x}^\text{output}) = n_\text{in}\text{E}(\mathbf{x}^\text{input})^2\text{Var}(\mathbf{w}) + n_\text{in}\text{E}(\mathbf{w})^2\text{Var}(\mathbf{x}^\text{input}) + n_\text{in}\text{Var}(\mathbf{w})\text{Var}(\mathbf{x}^\text{input}), \tag{93}$$

where $\text{E}(\mathbf{x})$ and $\text{Var}(\mathbf{x})$ denote the expected mean and variance of elements of $\mathbf{x}$, respectively. For similar length scales across layers, $\text{Var}(\mathbf{x}^\text{output})$ should be constant. Initially, similar variances can be achieved by normalizing ANN inputs to have zero mean, so that $\text{E}(\mathbf{x}^\text{input}) = 0$, and initializing weights so that $\text{E}(\mathbf{w}) = 0$ and $\text{Var}(\mathbf{w}) = 1/n_\text{in}$. However, parameters can shift during training, destabilizing learning. To compensate for parameter shift, popular normalization layers like batch normalization often impose $\text{E}(\mathbf{x}^\text{input}) = 0$ and $\text{Var}(\mathbf{x}^\text{input}) = 1$, relaxing need for $\text{E}(\mathbf{x}^\text{input}) = 0$ or $\text{E}(\mathbf{w}) = 0$. Nevertheless, training will still be sensitive to the length scale of trainable parameters.

There are a variety of popular weight initializers that adapt weights to ANN architecture. One of the oldest methods is LeCun initialization[941,951], where weights are initialized with variance,

$$\text{Var}(\mathbf{w}) = \frac{1}{n_\text{in}}, \tag{94}$$

which is argued to produce outputs with similar length scales in the previous paragraph. However, a similar argument can be made for initializing with $\text{Var}(\mathbf{w}) = 1/n_\text{out}$ to produce similar gradients at each layer during the backwards pass[1443]. As a compromise, Xavier initialization[1445] computes an average,

$$\text{Var}(\mathbf{w}) = \frac{2}{n_\text{in} + n_\text{out}}. \tag{95}$$

However, adjusting weights for n_out is not necessary for adaptive optimizers like ADAM, which divide gradients by their length scales, unless gradients will vanish or explode. Finally, He initialization[22] doubles the variance of weights to

$$\text{Var}(\mathbf{w}) = \frac{2}{n_\text{in}}, \tag{96}$$

and is often used in ReLU networks to compensate for activation functions halving variances of their outputs[22,1443,1446]. Most trainable parameters are initialized from either a zero-centred Gaussian or uniform distribution. For convenience, the limits of such a uniform distribution are $\pm(3\text{Var}(\mathbf{w}))^{1/2}$. Uniform initialization can outperform Gaussian initialization in DNNs due to Gaussian outliers harming learning[1443]. However, issues can be avoided by truncating Gaussian initialization, often to two standard deviations, and rescaling to its original variance.

Some initializers are mainly used for RNNs. For example, orthogonal initialization[1447] often improves RNN training[1448] by reducing susceptibility to vanishing and exploding gradients. Similarly, identity initialization[1449,1450] can help RNNs to learn long-term dependencies. In most ANNs, biases are initialized with zeros. However, the forget gates of LSTMs are often initialized with ones to decrease forgetting at the start of training[1211]. Finally, the start states of most RNNs are initialized with zeros or other constants. However, random multivariate or trainable variable start states can improve performance[1451].

There are a variety of alternatives to initialization from random multivariates. Weight normalized[1014] ANNs are a popular example of data-dependent initialization, where randomly initialized weight magnitudes and biases are chosen to counteract variances and means of an initial batch of data. Similarly, layer-sequential unit-variance (LSUV) initialization[1452] consists of orthogonal initialization followed by adjusting the magnitudes of weights to counteract variances of an initial batch of data. Other approaches standardize the norms of backpropagated gradients. For example, random walk initialization[1453] (RWI) finds scales for weights to prevent vanishing or exploding gradients in deep FNNs, albeit with varied success[1452]. Alternatively, MetaInit[1454] scales the magnitudes of randomly initialized weights to minimize changes in backpropagated gradients per iteration of gradient descent.

6.5 Regularization

There are a variety of regularization mechanisms[1455–1458] that modify learning algorithms to improve ANN performance. One of the most popular is LX regularization, which decays weights by adding a loss,

$$L_X = \lambda_X \sum_i \frac{|\theta_i|^X}{X} \,, \tag{97}$$

weighted by λ_X to each trainable variable, θ_i. L2 regularization[1459–1461] is preferred[1462] for most DNN optimization as subtraction of its gradient, $\partial_{\theta_i} L_2 = \lambda_2 \theta_i$, is equivalent to computationally-efficient multiplicative weight decay. Nevertheless, L1 regularization is better at inducing model sparsity[1463] than L2 regularization, and L1 regularization achieves higher performance in some applications[1464]. Higher performance can also be achieved by adding both L1 and L2 regularization in elastic nets[1465]. LX regularization is most effective at the start of training and becomes less important near convergence[1459]. Finally, L1 and L2 regularization are closely related to lasso[1466] and ridge[1467] regularization, respectively, whereby trainable parameters are adjusted to limit L_1 and L_2 losses.

Gradient clipping[1336, 1468–1470] accelerates learning by limiting large gradients, and is most commonly applied to RNNs. A simple approach is to clip gradient magnitudes to a threshold hyperparameter. However, it is more common to scale gradients, $\mathbf{g}_i$, at layer i if their norm is above a threshold, u, so that[1208, 1469]

$$\mathbf{g}_i \leftarrow \begin{cases} \mathbf{g}_i, & \text{if } ||\mathbf{g}_i||_n \leq u \\ \frac{u}{||\mathbf{g}_i||_n} \mathbf{g}_i, & \text{if } ||\mathbf{g}_i||_n > u \end{cases} \tag{98}$$

where $n = 2$ is often chosen to minimize computation. Similarly, gradients can be clipped if they are above a global norm,

$$g_{\text{norm}} = \left(\sum_{i=1}^{L} ||\mathbf{g}_i||_n^n \,, \right)^{1/n} \tag{99}$$

computed with gradients at L layers. Scaling gradient norms is often preferable to clipping to a threshold as scaling is akin to adapting layer learning rates and does not affect the directions of gradients. Thresholds for gradient clipping are often set based on average norms of backpropagated gradients during preliminary training[1471]. However, thresholds can also be set automatically and adaptively[1335, 1336]. In addition, adaptive gradient clipping algorithms can skip training iterations if gradient norms are anomalously high[1472], which often indicates an imminent gradient explosion.

Dropout[1473–1477] often reduces overfitting by only using a fraction, p_i, of layer i outputs during training, and multiplying all outputs by p_i for inference. However, dropout often increases training time, can be sensitive to p_i, and sometimes lowers performance[1478]. Improvements to dropout at the structural level, such as applying it to convolutional channels, paths, and layers, rather than random output elements, can improve performance[1479]. For example, DropBlock[1480] improves performance by dropping contiguous regions of feature maps to prevent dropout being trivially circumvented by using spatially correlated neighbouring outputs. Similarly, PatchUp[1481] swaps or mixes contiguous regions with regions for another sample. Dropout is often outperformed by Shakeout[1482, 1483], a modification of dropout that randomly enhances or reverses contributions of outputs to the next layer.

Noise often enhances ANN training by decreasing susceptibility to spurious local minima[1484]. Adding noise to trainable parameters can improve generalization[1485, 1486], or exploration for RL[1404]. Parameter noise is usually additive as it does not change an objective function being learned, whereas multiplicative noise can change the objective[1487]. In addition, noise can be added to inputs[1253, 1488], hidden layers[1158, 1489], generated outputs[1490] or target outputs[995, 1491]. However, adding noise to signals does not always improve performance[1217]. Finally, modifying usual gradient noise[1492] by adding noise to gradients can improve performance[1493]. Typically, additive noise is annealed throughout training, so that that final training is with a noiseless model that will be used for inference.

There are a variety of regularization mechanisms that exploit extra training data. A simple approach is to create extra training examples by data augmentation[1494–1496]. Extra training data can also be curated, or simulated for training by domain adaption[1176–1178]. Alternatively, semi-supervised learning[1497–1502] can generate target outputs for a dataset of unpaired inputs to augment training with a dataset of paired inputs and target outputs. Finally, multitask learning[1503–1507] can improve performance by introducing additional loss functions. For instance, by adding an auxiliary classifier to predict image labels from features generated by intermediate DNN layers[1508–1511]. Losses are often manually balanced; however, their gradients can also be balanced automatically and adaptively[1167,1168].

6.6 Data Pipeline

A data pipeline prepares data to be input to an ANN. Efficient pipelines often parallelize data preparation across multiple CPU cores[1512]. Small datasets can be stored in RAM to decrease data access times, whereas large dataset elements are often loaded from files. Loaded data can then be preprocessed and augmented[1494,1495,1513–1515]. For electron micrographs, preprocessing often includes replacing non-finite elements, such as NaN and inf, with finite values; linearly transforming intensities to a common range, such as $[-1,1]$ or zero mean and unit variance; and performing a random combination of flips and 90° to augment data by a factor of eight[70,201,202,231,349]. Preprocessed examples can then be combined into batches. Typically, multiple batches that are ready to be input are prefetched and stored in RAM to avoid delays due to fluctuating CPU performance.

To efficiently utilize data, training datasets are often reiterated over for multiple training epochs. Usually, training datasets are reiterated over about 10^2 times. Increasing epochs can maximize utilization of potentially expensive training data; however, increasing epochs can lower performance due to overfitting[1516,1517] or be too computationally expensive[539]. Naively, batches of data can be randomly sampled with replacement during training by gradient descent. However, convergence can be accelerated by reinitializing a training dataset at the start of each training epoch and randomly sampling data without replacement[1518–1522]. Most modern DLFs, such as TensorFlow, provide efficient and easy-to-use functions to control data sampling[1523].

6.7 Model Evaluation

There are a variety of methods for ANN performance evaluation[538]. However, most ANNs are evaluated by 1-fold validation, where a dataset is partitioned into training, validation, and test sets. After ANN optimization with a training set, ability to generalize is measured with a validation set. Multiple validations may be performed for training with early stopping[1516,1517] or ANN learning policy and architecture selection, so final performance is often measured with a test set to avoid overfitting to the validation set. Most researchers favour using single training, validation, and test sets to simplify standardization of performance benchmarks[231]. However, multiple-fold validation[538] or multiple validation sets[1524] can improve performance characterization. Alternatively, models can be bootstrap aggregated[1525] (bagged) from multiple models trained on different subsets of training data. Bagging is usually applied to random forests[1526–1528] or other lightweight models, and enables model uncertainly to be gauged from the variance of model outputs.

For small datasets, model performance is often sensitive to split of data between training and validation sets[1529]. Increasing training set size usually increases model accuracy, whereas increasing validation set size decreases performance uncertainty. Indeed, a scaling law can be used to estimate an optimal tradeoff[1530] between training and validation set sizes. However, most experimenters follow a Pareto[1531] splitting heuristic. For example, we often use a 75:15:10 training-validation-test split[231]. Heuristic splitting is justified for ANN training with large datasets insofar that sensitivity to splitting ratios decreases with increasing dataset size[2].

6.8 Deployment

If an ANN is deployed[1532–1534] on multiple different devices, such as various electron microscopes, a separate model can be trained for each device[403], Alternatively, a single model can be trained and specialized for different devices to decrease training requirements[1535]. In addition, ANNs can remotely service requests from cloud containers[1536–1538]. Integration of multiple ANNs can be complicated by different servers for different DLFs supporting different backends; however, unified interfaces are available. For example, GraphPipe[1539] provides simple, efficient reference model servers for Tensorflow, Caffe2, and ONNX; a minimalist machine learning transport specification based on FlatBuffers[1540]; and efficient client implementations in Go, Python, and Java. In 2020, most ANNs developed researchers were not deployed. However, we anticipate that deployment will become a more prominent consideration as the role of deep learning in electron microscopy matures.

Most ANNs are optimized for inference by minimizing parameters and operations from training time, like MobileNets[1094]. However, less essential operations can also be pruned after training[1541,1542]. Another approach is quantization, where ANN bit depths are decreased, often to efficient integer instructions, to increase inference throughput[1543,1544]. Quantization often decreases performance; however, the amount of quantization can be adapted to ANN components to optimize performance-throughput tradeoffs[1545]. Alternatively, training can be modified to minimize the impact of quantization on performance[1546–1548].

Another approach is to specialize bit manipulation for deep learning. For example, signed brain floating point (bfloat16) often improves accuracy on TPUs by using an 8 bit mantissa and 7 bit exponent, rather than a usual 5 bit mantissa and 10 bit exponent[1549]. Finally, ANNs can be adaptively selected from a set of ANNs based on available resources to balance tradeoff of performance and inference time[1550], similar to image optimization for web applications[1551,1552].

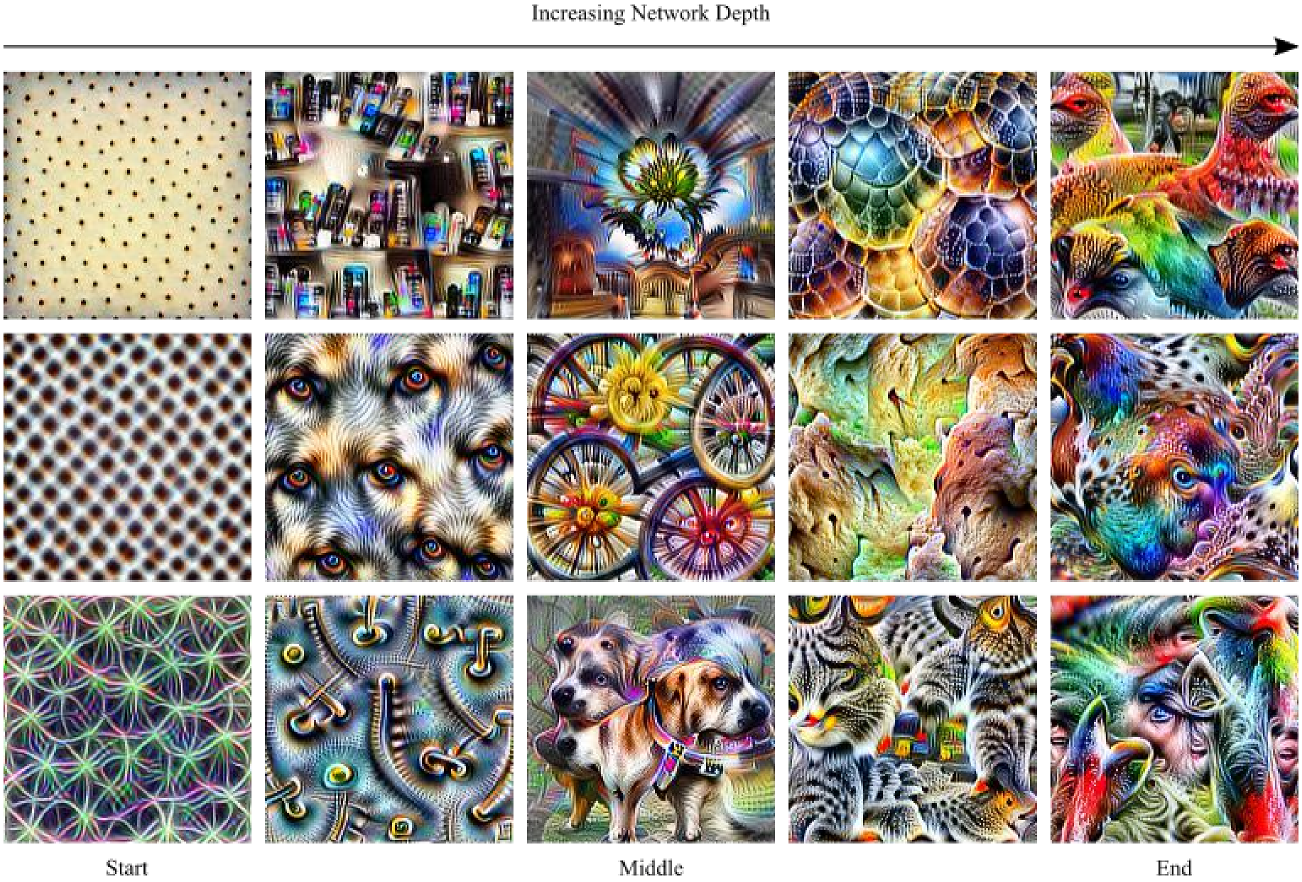


Figure 16. Inputs that maximally activate channels in GoogLeNet[1076] after training on ImageNet[71]. Neurons in layers near the start have small receptive fields and discern local features. Middle layers discern semantics recognisable by humans, such as dogs and wheels. Finally, layers at the end of the DNN, near its logits, discern combinations of semantics that are useful for labelling. This figure is adapted with permission[1553] under a Creative Commons Attribution 4.0[73] license.

6.9 Interpretation

We find that some electron microscopists are apprehensive about working with ANNs due to a lack of interpretability, irrespective of rigorous ANN validation. We try to address uncertainty by providing loss visualizations in some of our electron microscopy papers[70,201,202]. However, there are a variety of popular approaches to explainable artificial intelligence[1554–1560] (XAI). One of the most popular approaches to XAI is saliency[1561–1564], where gradients of outputs w.r.t. inputs correlate with their importance. Saliency is often computed by gradient backpropagation[1565–1567]. For example, with Grad-CAM[1568] or its variants[1569–1572]. Alternatively, saliency can be predicted by ANNs[1054,1573,1574] or a variety of methods inspired by Grad-CAM[1575–1577]. Applications of saliency include selecting useful features from a model[1578], and locating regions in inputs corresponding to ANN outputs[1579].

There are a variety of other approaches to XAI. For example, feature visualization via optimization[1553,1580–1583] can find inputs that maximally activate parts of an ANN, as shown in figure 16. Another approach is to cluster features, e.g. by tSNE[1584,1585] with the Barnes-Hut algorithm[1586,1587], and examine corresponding clustering of inputs or outputs[231]. Finally, developers can view raw features and gradients during forward and backward passes of gradient descent, respectively. For example, CNN explainer[1588,1589] is an interactive visualization tool designed for non-experts to learn and experiment with CNNs. Similarly, GAN Lab[1590] is an interactive visualization tool for non-experts to learn and experiment with GANs.

7 Discussion

We introduced a variety of electron microscopy applications in section 1 that have been enabled or enhanced by deep learning. Nevertheless, the greatest benefit of deep learning in electron microscopy may be general-purpose tools that enable researchers to be more effective. Search engines based on deep learning are almost essential to navigate an ever-increasing number of scientific publications[700]. Further, machine learning can enhance communication by filtering spam and phishing attacks[1591–1593], and by summarizing[1594–1596] and classifying[1055, 1597–1599] scientific documents. In addition, machine learning can be applied to education to automate and standardize scoring[1600–1603], detect plagiarism[1604–1606], and identify at-risk students[1607].

Creative applications of deep learning[1608, 1609] include making new art by style transfer[1001–1005], composing music[1610–1612], and storytelling[1613, 1614]. Similar DNNs can assist programmers[1615, 1616]. For example, by predictive source code completion[1617–1622], and by generating source code to map inputs to target outputs[1623] or from labels describing desired source code[1624]. Text generating DNNs can also help write scientific papers. For example, by drafting scientific passages[1625] or drafting part of a paper from a list of references[1626]. Papers generated by early prototypes for automatic scientific paper generators, such as SciGen[1627], are realistic insofar that they have been accepted by scientific venues.

An emerging application of deep learning is mining scientific resources to make new scientific discoveries[1628]. Artificial agents are able to effectively distil latent scientific knowledge as they can parallelize examination of huge amounts of data, whereas information access by humans[1629–1631] is limited by human cognition[1632]. High bandwidth bi-directional brain-machine interfaces are being developed to overcome limitations of human cognition[1633]; however, they are in the early stages of development and we expect that they will depend on substantial advances in machine learning to enhance control of cognition. Eventually, we expect that ANNs will be used as scientific oracles, where researchers who do not rely on their services will no longer be able to compete. For example, an ANN trained on a large corpus of scientific literature predicted multiple advances in materials science before they were reported[1634]. ANNs are already used for financial asset management[1635, 1636] and recruiting[1637–1640], so we anticipate that artificial scientific oracle consultation will become an important part of scientific grant[1641, 1642] reviews.

A limitation of deep learning is that it can introduce new issues. For example, DNNs are often susceptible to adversarial attacks[1643–1647], where small perturbations to inputs cause large errors. Nevertheless, training can be modified to improve robustness to adversarial attacks[1648–1652]. Another potential issue is architecture-specific systematic errors. For example, CNNs often exhibit structured systematic error variation[70, 201, 202, 1092, 1093, 1653], including higher errors nearer output edges[70, 201, 202]. However, structured systematic error variation can be minimized by GANs incentifying the generation of realistic outputs[201]. Finally, ANNs can be difficult to use as they often require downloading code with undocumented dependencies, downloading a pretrained model, and may require hardware accelerators. These issues can be avoided by serving ANNs from cloud containers. However, it may not be practical for academics to acquire funding to cover cloud service costs.

Perhaps the most important aspect of deep learning in electron microscopy is that it presents new challenges that can lead to advances in machine learning. Simple benchmarks like CIFAR-10[562, 563] and MNIST[564] have been solved. Following, more difficult benchmarks like Fashion-MNIST[1654] have been introduced. However, they only partially address issues with solved datasets as they do not present fundamentally new challenges. In contrast, we believe that new problems often invite new solutions. For example, we developed adaptive learning rate clipping[261] (ALRC) to stabilize training of DNNs for partial scanning transmission electron microscopy[201]. The challenge was that we wanted to train a large model for high-resolution images; however, training was unstable if we used small batches needed to fit it in GPU memory. Similar challenges abound and can lead to advances in both machine learning and electron microscopy.

Data Availability

No new data were created or analysed in this study.

Acknowledgements

Thanks go to Jeremy Sloan and Martin Lotz for internally reviewing this article. In addition, part of the text in section 1.2 is adapted from our earlier work with permission[201] under a Creative Commons Attribution 4.0[73] license. Finally, the author acknowledges funding from EPSRC grant EP/N035437/1 and EPSRC Studentship 1917382.

Competing Interests

The author declares no competing interests.

References

1. Leiserson, C. E. *et al.* There's Plenty of Room at the Top: What Will Drive Computer Performance After Moore's Law? *Science* **368** (2020).
2. Sun, C., Shrivastava, A., Singh, S. & Gupta, A. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In *Proceedings of the IEEE International Conference on Computer Vision*, 843–852 (2017).
3. Hey, T., Butler, K., Jackson, S. & Thiyagalingam, J. Machine Learning and Big Scientific Data. *Philos. Transactions Royal Soc. A* **378**, 20190054 (2020).
4. Sengupta, S. *et al.* A Review of Deep Learning with Special Emphasis on Architectures, Applications and Recent Trends. *Knowledge-Based Syst.* **4**, 105596 (2020).
5. Shrestha, A. & Mahmood, A. Review of Deep Learning Algorithms and Architectures. *IEEE Access* **7**, 53040–53065 (2019).
6. Dargan, S., Kumar, M., Ayyagari, M. R. & Kumar, G. A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning. *Arch. Comput. Methods Eng.* **27**, 1071–1092 (2019).
7. Alom, M. Z. *et al.* A State-of-the-Art Survey on Deep Learning Theory and Architectures. *Electronics* **8**, 292 (2019).
8. Zhang, Q., Yang, L. T., Chen, Z. & Li, P. A Survey on Deep Learning for Big Data. *Inf. Fusion* **42**, 146–157 (2018).
9. Hatcher, W. G. & Yu, W. A Survey of Deep Learning: Platforms, Applications and Emerging Research Trends. *IEEE Access* **6**, 24411–24432 (2018).
10. LeCun, Y., Bengio, Y. & Hinton, G. Deep Learning. *Nature* **521**, 436–444 (2015).
11. Schmidhuber, J. Deep Learning in Neural Networks: An Overview. *Neural Networks* **61**, 85–117 (2015).
12. Ge, M., Su, F., Zhao, Z. & Su, D. Deep Learning Analysis on Microscopic Imaging in Materials Science. *Mater. Today Nano* **11**, 100087 (2020).
13. Carleo, G. *et al.* Machine Learning and the Physical Sciences. *Rev. Mod. Phys.* **91**, 045002 (2019).
14. Wei, J. *et al.* Machine Learning in Materials Science. *InfoMat* **1**, 338–358 (2019).
15. Barbastathis, G., Ozcan, A. & Situ, G. On the Use of Deep Learning for Computational Imaging. *Optica* **6**, 921–943 (2019).
16. Schleder, G. R., Padilha, A. C., Acosta, C. M., Costa, M. & Fazzio, A. From DFT to Machine Learning: Recent Approaches to Materials Science – A Review. *J. Physics: Mater.* **2**, 032001 (2019).
17. von Lilienfeld, O. A. Introducing Machine Learning: Science and Technology. *Mach. Learn. Sci. Technol.* **1**, 010201 (2020).
18. Sejnowski, T. J. *The Deep Learning Revolution* (MIT Press, 2018).
19. Alom, M. Z. *et al.* The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches. *arXiv preprint arXiv:1803.01164* (2018).
20. Wang, Y. & Kosinski, M. Deep Neural Networks are More Accurate than Humans at Detecting Sexual Orientation from Facial Images. *J. Pers. Soc. Psychol.* **114**, 246 (2018).
21. Kheradpisheh, S. R., Ghodrati, M., Ganjtabesh, M. & Masquelier, T. Deep Networks can Resemble Human Feed-Forward Vision in Invariant Object Recognition. *Sci. Reports* **6**, 32672 (2016).
22. He, K., Zhang, X., Ren, S. & Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of the IEEE International Conference on Computer Vision*, 1026–1034 (2015).
23. Lu, C. & Tang, X. Surpassing Human-Level Face Verification Performance on LFW with GaussianFace. In *Twenty-Ninth AAAI Conference on Artificial Intelligence* (2015).
24. Vinyals, O. *et al.* AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. Online: https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/ (2019).
25. Firoiu, V., Whitney, W. F. & Tenenbaum, J. B. Beating the World's Best at Super Smash Bros. with Deep Reinforcement Learning. *arXiv preprint arXiv:1702.06230* (2017).
26. Lample, G. & Chaplot, D. S. Playing FPS Games with Deep Reinforcement Learning. In *Thirty-First AAAI Conference on Artificial Intelligence* (2017).
27. Silver, D. *et al.* Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* **529**, 484–489 (2016).

28. Mnih, V. *et al.* Playing Atari with Deep Reinforcement Learning. *arXiv preprint arXiv:1312.5602* (2013).

29. Tesauro, G. Programming Backgammon Using Self-Teaching Neural Nets. *Artif. Intell.* **134**, 181–199 (2002).

30. Han, S. S. *et al.* Deep Neural Networks Show an Equivalent and Often Superior Performance to Dermatologists in Onychomycosis Diagnosis: Automatic Construction of Onychomycosis Datasets by Region-Based Convolutional Deep Neural Network. *PLOS ONE* **13**, e0191493 (2018).

31. Wang, D., Khosla, A., Gargeya, R., Irshad, H. & Beck, A. H. Deep Learning for Identifying Metastatic Breast Cancer. *arXiv preprint arXiv:1606.05718* (2016).

32. Santoro, A. *et al.* A Simple Neural Network Module for Relational Reasoning. In *Advances in Neural Information Processing Systems*, 4967–4976 (2017).

33. Xiong, W. *et al.* Achieving Human Parity in Conversational Speech Recognition. *arXiv preprint arXiv:1610.05256* (2016).

34. Weng, C., Yu, D., Seltzer, M. L. & Droppo, J. Single-Channel Mixed Speech Recognition Using Deep Neural Networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5632–5636 (IEEE, 2014).

35. Lee, K., Zung, J., Li, P., Jain, V. & Seung, H. S. Superhuman Accuracy on the SNEMI3D Connectomics Challenge. *arXiv preprint arXiv:1706.00120* (2017).

36. Weyand, T., Kostrikov, I. & Philbin, J. Planet-Photo Geolocation with Convolutional Neural Networks. In *European Conference on Computer Vision*, 37–55 (Springer, 2016).

37. Kidger, P. & Lyons, T. Universal Approximation with Deep Narrow Networks. *arXiv preprint arXiv:1905.08539* (2019).

38. Lin, H. & Jegelka, S. ResNet with One-Neuron Hidden Layers is a Universal Approximator. In *Advances in Neural Information Processing Systems*, 6169–6178 (2018).

39. Hanin, B. & Sellke, M. Approximating Continuous Functions by ReLU Nets of Minimal Width. *arXiv preprint arXiv:1710.11278* (2017).

40. Lu, Z., Pu, H., Wang, F., Hu, Z. & Wang, L. The Expressive Power of Neural Networks: A View from the Width. In *Advances in Neural Information Processing Systems*, 6231–6239 (2017).

41. Pinkus, A. Approximation Theory of the MLP Model in Neural Networks. *Acta Numer.* **8**, 143–195 (1999).

42. Leshno, M., Lin, V. Y., Pinkus, A. & Schocken, S. Multilayer Feedforward Networks with a Nonpolynomial Activation Function can Approximate any Function. *Neural Networks* **6**, 861–867 (1993).

43. Hornik, K. Approximation Capabilities of Multilayer Feedforward Networks. *Neural Networks* **4**, 251–257 (1991).

44. Hornik, K., Stinchcombe, M. & White, H. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks* **2**, 359–366 (1989).

45. Cybenko, G. Approximation by Superpositions of a Sigmoidal Function. *Math. Control. Signals Syst.* **2**, 303–314 (1989).

46. Johnson, J. Deep, Skinny Neural Networks are not Universal Approximators. *arXiv preprint arXiv:1810.00393* (2018).

47. Lin, H. W., Tegmark, M. & Rolnick, D. Why Does Deep and Cheap Learning Work so Well? *J. Stat. Phys.* **168**, 1223–1247 (2017).

48. Gühring, I., Raslan, M. & Kutyniok, G. Expressivity of Deep Neural Networks. *arXiv preprint arXiv:2007.04759* (2020).

49. Raghu, M., Poole, B., Kleinberg, J., Ganguli, S. & Sohl-Dickstein, J. On the Expressive Power of Deep Neural Networks. In *International Conference on Machine Learning*, 2847–2854 (2017).

50. Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J. & Ganguli, S. Exponential Expressivity in Deep Neural Networks Through Transient Chaos. In *Advances in Neural Information Processing Systems*, 3360–3368 (2016).

51. Hanin, B. & Rolnick, D. Deep ReLU Networks Have Surprisingly Few Activation Patterns. In *Advances in Neural Information Processing Systems*, 361–370 (2019).

52. Cao, Y. & Gu, Q. Generalization Error Bounds of Gradient Descent for Learning Over-Parameterized Deep ReLU Networks. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 3349–3356 (2020).

53. Geiger, M. *et al.* Scaling Description of Generalization with Number of Parameters in Deep Learning. *J. Stat. Mech. Theory Exp.* **2020**, 023401 (2020).

54. Dziugaite, G. K. *Revisiting Generalization for Deep Learning: PAC-Bayes, Flat Minima, and Generative Models*. Ph.D thesis, University of Cambridge (2020).

55. Cao, Y. & Gu, Q. Generalization Bounds of Stochastic Gradient Descent for Wide and Deep Neural Networks. In *Advances in Neural Information Processing Systems*, 10836–10846 (2019).

56. Xu, Z. J. Understanding Training and Generalization in Deep Learning by Fourier Analysis. *arXiv preprint arXiv:1808.04295* (2018).

57. Neyshabur, B., Bhojanapalli, S., McAllester, D. & Srebro, N. Exploring Generalization in Deep Learning. In *Advances in Neural Information Processing systems*, 5947–5956 (2017).

58. Wu, L., Zhu, Z. *et al.* Towards Understanding Generalization of Deep Learning: Perspective of Loss Landscapes. *arXiv preprint arXiv:1706.10239* (2017).

59. Kawaguchi, K., Kaelbling, L. P. & Bengio, Y. Generalization in Deep Learning. *arXiv preprint arXiv:1710.05468* (2017).

60. Iten, R., Metger, T., Wilming, H., Del Rio, L. & Renner, R. Discovering Physical Concepts with Neural Networks. *Phys. Rev. Lett.* **124**, 010508 (2020).

61. Wu, T. & Tegmark, M. Toward an Artificial Intelligence Physicist for Unsupervised Learning. *Phys. Rev. E* **100**, 033311 (2019).

62. Chen, Y., Xie, Y., Song, L., Chen, F. & Tang, T. A Survey of Accelerator Architectures for Deep Neural Networks. *Engineering* **6**, 264–274 (2020).

63. Garrido, M., Qureshi, F., Takala, J. & Gustafsson, O. Hardware Architectures for the Fast Fourier Transform. In *Handbook of Signal Processing Systems*, 613–647 (Springer, 2019).

64. Velik, R. Discrete Fourier Transform Computation Using Neural Networks. In *2008 International Conference on Computational Intelligence and Security*, 120–123 (IEEE, 2008).

65. Moreland, K. & Angel, E. The FFT on a GPU. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, 112–119 (Eurographics Association, 2003).

66. Breen, P. G., Foley, C. N., Boekholt, T. & Zwart, S. P. Newton Versus the Machine: Solving the Chaotic Three-Body Problem Using Deep Neural Networks. *Mon. Notices Royal Astron. Soc.* **494**, 2465–2470 (2020).

67. Ryczko, K., Strubbe, D. A. & Tamblyn, I. Deep Learning and Density-Functional Theory. *Phys. Rev. A* **100**, 022512 (2019).

68. Sinitskiy, A. V. & Pande, V. S. Deep Neural Network Computes Electron Densities and Energies of a Large Set of Organic Molecules Faster than Density Functional Theory (DFT). *arXiv preprint arXiv:1809.02723* (2018).

69. Zhang, G. *et al.* Fast Phase Retrieval in Off-Axis Digital Holographic Microscopy Through Deep Learning. *Opt. Express* **26**, 19388–19405 (2018).

70. Ede, J. M. & Beanland, R. Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder. *Ultramicroscopy* **202**, 18–25 (2019).

71. Krizhevsky, A., Sutskever, I. & Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, 1097–1105 (2012).

72. Ede, J. M. Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder. *arXiv preprint arXiv:1807.11234* (2018).

73. Creative Commons Attribution 4.0 International (CC BY 4.0). Online: https://creativecommons.org/licenses/by/4.0 (2020).

74. Liu, B. & Liu, J. Overview of Image Denoising Based on Deep Learning. In *Journal of Physics: Conference Series*, vol. 1176, 022010 (IOP Publishing, 2019).

75. Tian, C. *et al.* Deep Learning on Image Denoising: An Overview. *arXiv preprint arXiv:1912.13171* (2019).

76. Yoon, D., Lim, H. S., Jung, K., Kim, T. Y. & Lee, S. Deep Learning-Based Electrocardiogram Signal Noise Detection and Screening Model. *Healthc. Informatics Res.* **25**, 201–211 (2019).

77. Antczak, K. Deep Recurrent Neural Networks for ECG Signal Denoising. *arXiv preprint arXiv:1807.11551* (2018).

78. Bai, T., Nguyen, D., Wang, B. & Jiang, S. Probabilistic Self-Learning Framework for Low-Dose CT Denoising. *arXiv preprint arXiv:2006.00327* (2020).

79. Jifara, W., Jiang, F., Rho, S., Cheng, M. & Liu, S. Medical Image Denoising Using Convolutional Neural Network: A Residual Learning Approach. *The J. Supercomput.* **75**, 704–718 (2019).

80. Feng, D., Wu, W., Li, H. & Li, Q. Speckle Noise Removal in Ultrasound Images Using a Deep Convolutional Neural Network and a Specially Designed Loss Function. In *International Workshop on Multiscale Multimodal Medical Imaging*, 85–92 (Springer, 2019).

81. de Haan, K., Rivenson, Y., Wu, Y. & Ozcan, A. Deep-Learning-Based Image Reconstruction and Enhancement in Optical Microscopy. *Proc. IEEE* **108**, 30–50 (2019).

82. Manifold, B., Thomas, E., Francis, A. T., Hill, A. H. & Fu, D. Denoising of Stimulated Raman Scattering Microscopy Images via Deep Learning. *Biomed. Opt. Express* **10**, 3860–3874 (2019).

83. Devalla, S. K. *et al.* A Deep Learning Approach to Denoise Optical Coherence Tomography Images of the Optic Nerve Head. *Sci. Reports* **9**, 1–13 (2019).

84. Choi, G. *et al.* Cycle-Consistent Deep Learning Approach to Coherent Noise Reduction in Optical Diffraction tomography. *Opt. Express* **27**, 4927–4943 (2019).

85. Azarang, A. & Kehtarnavaz, N. A Review of Multi-Objective Deep Learning Speech Denoising Methods. *Speech Commun.* (2020).

86. Choi, H.-S., Heo, H., Lee, J. H. & Lee, K. Phase-Aware Single-Stage Speech Denoising and Dereverberation with U-Net. *arXiv preprint arXiv:2006.00687* (2020).

87. Alamdari, N., Azarang, A. & Kehtarnavaz, N. Self-Supervised Deep Learning-Based Speech Denoising. *arXiv* arXiv–1904 (2019).

88. Han, K. *et al.* Learning Spectral Mapping for Speech Dereverberation and Denoising. *IEEE/ACM Transactions on Audio, Speech, Lang. Process.* **23**, 982–992 (2015).

89. Goyal, B., Dogra, A., Agrawal, S., Sohi, B. & Sharma, A. Image Denoising Review: From Classical to State-of-the-Art Approaches. *Inf. Fusion* **55**, 220–244 (2020).

90. Girdher, A., Goyal, B., Dogra, A., Dhindsa, A. & Agrawal, S. Image Denoising: Issues and Challenges. *Available at SSRN 3446627* (2019).

91. Fan, L., Zhang, F., Fan, H. & Zhang, C. Brief Review of Image Denoising Techniques. *Vis. Comput. for Ind. Biomed. Art* **2**, 7 (2019).

92. Gedraite, E. S. & Hadad, M. Investigation on the Effect of a Gaussian Blur in Image Filtering and Segmentation. In *Proceedings ELMAR*, 393–396 (IEEE, 2011).

93. Deng, G. & Cahill, L. An Adaptive Gaussian Filter for Noise Reduction and Edge Detection. In *1993 IEEE Conference Record Nuclear Science Symposium and Medical Imaging Conference*, 1615–1619 (IEEE, 1993).

94. Chang, H.-H., Lin, Y.-J. & Zhuang, A. H. An Automatic Parameter Decision System of Bilateral Filtering with GPU-Based Acceleration for Brain MR Images. *J. Digit. Imaging* **32**, 148–161 (2019).

95. Chaudhury, K. N. & Rithwik, K. Image Denoising Using Optimally Weighted Bilateral Filters: A Sure and Fast Approach. In *IEEE International Conference on Image Processing*, 108–112 (IEEE, 2015).

96. Anantrasirichai, N. *et al.* Adaptive-Weighted Bilateral Filtering and Other Pre-Processing Techniques for Optical Coherence Tomography. *Comput. Med. Imaging Graph.* **38**, 526–539 (2014).

97. Tomasi, C. & Manduchi, R. Bilateral Filtering for Gray and Color Images. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, 839–846 (IEEE, 1998).

98. Budhiraja, S., Goyal, B., Dogra, A., Agrawal, S. *et al.* An Efficient Image Denoising Scheme for Higher Noise Levels Using Spatial Domain Filters. *Biomed. Pharmacol. J.* **11**, 625–634 (2018).

99. Nair, R. R., David, E. & Rajagopal, S. A Robust Anisotropic Diffusion Filter with Low Arithmetic Complexity for Images. *EURASIP J. on Image Video Process.* **2019**, 48 (2019).

100. Perona, P. & Malik, J. Scale-Space and Edge Detection Using Anisotropic Diffusion. *IEEE Transactions on Pattern Analysis Mach. Intell.* **12**, 629–639 (1990).

101. Wang, Z. & Zhang, D. Progressive Switching Median Filter for the Removal of Impulse Noise from Highly Corrupted Images. *IEEE Transactions on Circuits Syst. II: Analog. Digit. Signal Process.* **46**, 78–80 (1999).

102. Yang, R., Yin, L., Gabbouj, M., Astola, J. & Neuvo, Y. Optimal Weighted Median Filtering Under Structural Constraints. *IEEE Transactions on Signal Process.* **43**, 591–604 (1995).

103. Kodi Ramanah, D., Lavaux, G. & Wandelt, B. D. Wiener Filter Reloaded: Fast Signal Reconstruction Without Preconditioning. *Mon. Notices Royal Astron. Soc.* **468**, 1782–1793 (2017).

104. Elsner, F. & Wandelt, B. D. Efficient Wiener Filtering Without Preconditioning. *Astron. & Astrophys.* **549**, A111 (2013).

105. Robinson, E. A. & Treitel, S. Principles of Digital Wiener Filtering. *Geophys. Prospect.* **15**, 311–332 (1967).

106. Bayer, F. M., Kozakevicius, A. J. & Cintra, R. J. An Iterative Wavelet Threshold for Signal Denoising. *Signal Process.* **162**, 10–20 (2019).

107. Mohideen, S. K., Perumal, S. A. & Sathik, M. M. Image De-Noising Using Discrete Wavelet Transform. *Int. J. Comput. Sci. Netw. Secur.* **8**, 213–216 (2008).

108. Luisier, F., Blu, T. & Unser, M. A New SURE Approach to Image Denoising: Interscale Orthonormal Wavelet Thresholding. *IEEE Transactions on Image Process.* **16**, 593–606 (2007).

109. Jansen, M. & Bultheel, A. Empirical Bayes Approach to Improve Wavelet Thresholding for Image Noise Reduction. *J. Am. Stat. Assoc.* **96**, 629–639 (2001).

110. Chang, S. G., Yu, B. & Vetterli, M. Adaptive Wavelet Thresholding for Image Denoising and Compression. *IEEE Transactions on Image Process.* **9**, 1532–1546 (2000).

111. Donoho, D. L. & Johnstone, J. M. Ideal Spatial Adaptation by Wavelet Shrinkage. *Biometrika* **81**, 425–455 (1994).

112. Ma, J. & Plonka, G. The Curvelet Transform. *IEEE Signal Process. Mag.* **27**, 118–133 (2010).

113. Starck, J.-L., Candès, E. J. & Donoho, D. L. The Curvelet Transform for Image Denoising. *IEEE Transactions on Image Process.* **11**, 670–684 (2002).

114. Ahmed, S. S. *et al.* Nonparametric Denoising Methods Based on Contourlet Transform with Sharp Frequency Localization: Application to Low Exposure Time Electron Microscopy Images. *Entropy* **17**, 3461–3478 (2015).

115. Do, M. N. & Vetterli, M. The Contourlet Transform: An Efficient Directional Multiresolution Image Representation. *IEEE Transactions on Image Process.* **14**, 2091–2106 (2005).

116. Diwakar, M. & Kumar, P. Wavelet Packet Based CT Image Denoising Using Bilateral Method and Bayes Shrinkage Rule. In *Handbook of Multimedia Information Security: Techniques and Applications*, 501–511 (Springer, 2019).

117. Thakur, K., Damodare, O. & Sapkal, A. Hybrid Method for Medical Image Denoising Using Shearlet Transform and Bilateral Filter. In *2015 International Conference on Information Processing (ICIP)*, 220–224 (IEEE, 2015).

118. Nagu, M. & Shanker, N. V. Image De-Noising by Using Median Filter and Weiner Filter. *Image* **2**, 5641–5649 (2014).

119. Bae, T.-W. Spatial and Temporal Bilateral Filter for Infrared Small Target Enhancement. *Infrared Phys. & Technol.* **63**, 42–53 (2014).

120. Knaus, C. & Zwicker, M. Dual-Domain Image Denoising. In *2013 IEEE International Conference on Image Processing*, 440–444 (IEEE, 2013).

121. Danielyan, A., Katkovnik, V. & Egiazarian, K. BM3D Frames and Variational Image Deblurring. *IEEE Transactions on Image Process.* **21**, 1715–1728 (2011).

122. Dabov, K., Foi, A., Katkovnik, V. & Egiazarian, K. Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering. *IEEE Transactions on Image Process.* **16**, 2080–2095 (2007).

123. Jia, L. *et al.* Image Denoising via Sparse Representation Over Grouped Dictionaries with Adaptive Atom Size. *IEEE Access* **5**, 22514–22529 (2017).

124. Shao, L., Yan, R., Li, X. & Liu, Y. From Heuristic Optimization to Dictionary Learning: A Review and Comprehensive Comparison of Image Denoising Algorithms. *IEEE Transactions on Cybern.* **44**, 1001–1013 (2013).

125. Chatterjee, P. & Milanfar, P. Clustering-Based Denoising with Locally Learned Dictionaries. *IEEE Transactions on Image Process.* **18**, 1438–1451 (2009).

126. Aharon, M., Elad, M. & Bruckstein, A. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE Transactions on Signal Process.* **54**, 4311–4322 (2006).

127. Elad, M. & Aharon, M. Image Denoising via Sparse and Redundant Representations Over Learned Dictionaries. *IEEE Transactions on Image processing* **15**, 3736–3745 (2006).

128. Pairis, S. *et al.* Shot-Noise-Limited Nanomechanical Detection and Radiation Pressure Backaction from an Electron Beam. *Phys. Rev. Lett.* **122**, 083603 (2019).

129. Seki, T., Ikuhara, Y. & Shibata, N. Theoretical Framework of Statistical Noise in Scanning Transmission Electron Microscopy. *Ultramicroscopy* **193**, 118–125 (2018).

130. Lee, Z., Rose, H., Lehtinen, O., Biskupek, J. & Kaiser, U. Electron Dose Dependence of Signal-to-Noise Ratio, Atom Contrast and Resolution in Transmission Electron Microscope Images. *Ultramicroscopy* **145**, 3–12 (2014).

131. Timischl, F., Date, M. & Nemoto, S. A Statistical Model of Signal–Noise in Scanning Electron Microscopy. *Scanning* **34**, 137–144 (2012).

132. Sim, K., Thong, J. & Phang, J. Effect of Shot Noise and Secondary Emission Noise in Scanning Electron Microscope Images. *Scanning: The J. Scanning Microsc.* **26**, 36–40 (2004).

133. Boyat, A. K. & Joshi, B. K. A Review Paper: Noise Models in Digital Image Processing. *arXiv preprint arXiv:1505.03489* (2015).

134. Meyer, R. R. & Kirkland, A. I. Characterisation of the Signal and Noise Transfer of CCD Cameras for Electron Detection *Microsc. Res. Tech.* **49**, 269–280 (2000).

135. Kujawa, S. & Krahl, D. Performance of a Low-Noise CCD Camera Adapted to a Transmission Electron Microscope *Ultramicroscopy* **46**, 395–403 (1992).

136. Rose, H. H. Optics of High-Performance Electron Microscopes. *Sci. Technol. Adv. Mater.* **9**, 014107 (2008).

137. Fujinaka, S., Sato, Y., Teranishi, R. & Kaneko, K. Understanding of Scanning-System Distortions of Atomic-Scale Scanning Transmission Electron Microscopy Images for Accurate Lattice Parameter Measurements. *J. Mater. Sci.* **55**, 8123–8133 (2020).

138. Sang, X. *et al.* Dynamic Scan Control in STEM: Spiral Scans. *Adv. Struct. Chem. Imaging* **2**, 1–8 (2016).

139. Ning, S. *et al.* Scanning Distortion Correction in STEM Images. *Ultramicroscopy* **184**, 274–283 (2018).

140. Ophus, C., Ciston, J. & Nelson, C. T. Correcting Nonlinear Drift Distortion of Scanning Probe and Scanning Transmission Electron Microscopies from Image Pairs with Orthogonal Scan Directions. *Ultramicroscopy* **162**, 1–9 (2016).

141. Jones, L. & Nellist, P. D. Identifying and Correcting Scan Noise and Drift in the Scanning Transmission Electron Microscope. *Microsc. Microanal.* **19**, 1050–1060 (2013).

142. Karthik, C., Kane, J., Butt, D. P., Windes, W. & Ubic, R. In Situ Transmission Electron Microscopy of Electron-Beam Induced Damage Process in Nuclear Grade Graphite. *J. nuclear materials* **412**, 321–326 (2011).

143. Roels, J. *et al.* An Interactive ImageJ Plugin for Semi-Automated Image Denoising in Electron Microscopy. *Nat. Commun.* **11**, 1–13 (2020).

144. Narasimha, R. *et al.* Evaluation of Denoising Algorithms for Biological Electron Tomography. *J. Struct. Biol.* **164**, 7–17 (2008).

145. Mevenkamp, N. *et al.* Poisson Noise Removal from High-Resolution STEM Images based on Periodic Block Matching *Adv. Struct. Chem. Imaging* **1**, 3 (2015).

146. Bajić, B., Lindblad, J. & Sladoje, N. Blind Restoration of Images Degraded with Mixed Poisson-Gaussian Noise with Application in Transmission Electron Microscopy. In *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, 123–127 (IEEE, 2016).

147. Bodduna, K. & Weickert, J. Image Denoising with Less Artefacts: Novel Non-Linear Filtering on Fast Patch Reorderings *arXiv preprint arXiv:2002.00638* (2020).

148. Jonić, S. *et al.* Denoising of High-Resolution Single-Particle Electron-Microscopy Density Maps by Their Approximation Using Three-Dimensional Gaussian Functions. *J. Struct. Biol.* **194**, 423–433 (2016).

149. Chung, S.-C. *et al.* Two-Stage Dimension Reduction for Noisy High-Dimensional Images and Application to Cryogenic Electron Microscopy. *arXiv* arXiv–1911 (2020).

150. Wang, J. & Yin, C. A Zernike-Moment-Based Non-Local Denoising Filter for Cryo-EM Images. *Sci. China Life Sci.* **56**, 384–390 (2013).

151. Furnival, T., Leary, R. K. & Midgley, P. A. Denoising Time-Resolved Microscopy Image Sequences with Singular Value Thresholding. *Ultramicroscopy* **178**, 112–124 (2017).

152. Sorzano, C. O. S., Ortiz, E., López, M. & Rodrigo, J. Improved Bayesian Image Denoising Based on Wavelets with Applications to Electron Microscopy. *Pattern Recognit.* **39**, 1205–1213 (2006).

153. Ouyang, J. *et al.* Cryo-Electron Microscope Image Denoising Based on the Geodesic Distance. *BMC Struct. Biol.* **18**, 18 (2018).

154. Du, H. A Nonlinear Filtering Algorithm for Denoising HR (S)TEM Micrographs. *Ultramicroscopy* **151**, 62–67 (2015).

155. Kushwaha, H. S., Tanwar, S., Rathore, K. & Srivastava, S. De-noising Filters for TEM (Transmission Electron Microscopy) Image of Nanomaterials. In *2012 Second International Conference on Advanced Computing & Communication Technologies*, 276–281 (IEEE, 2012).

156. Hanai, T., Morinaga, T., Suzuki, H. & Hibino, M. Maximum Entropy Restoration of Electron Microscope Images with a Random-Spatial-Distribution Constraint. *Scanning Microsc.* **11**, 379–390 (1997).

157. Pennycook, S. J. The Impact of STEM Aberration Correction on Materials Science. *Ultramicroscopy* **180**, 22–33 (2017).

158. Ramasse, Q. M. Twenty Years After: How "Aberration Correction in the STEM" Truly Placed a "A Synchrotron in a Microscope". *Ultramicroscopy* **180**, 41–51 (2017).

159. Hawkes, P. Aberration Correction Past and Present. *Philos. Transactions Royal Soc. A: Math. Phys. Eng. Sci.* **367**, 3637–3664 (2009).

160. Goodge, B. H., Bianco, E. & Kourkoutis, H. W. Atomic-Resolution Cryo-STEM Across Continuously Variable Temperature. *arXiv preprint arXiv:2001.11581* (2020).

161. Egerton, R. F. Radiation Damage to Organic and Inorganic Specimens in the TEM. *Micron* **119**, 72–87 (2019).

162. Egerton, R. F. Control of Radiation Damage in the TEM. *Ultramicroscopy* **127**, 100–108 (2013).

163. Egerton, R. Mechanisms of Radiation Damage in Beam-Sensitive Specimens, for TEM Accelerating Voltages Between 10 and 300 kV. *Microsc. Res. Tech.* **75**, 1550–1556 (2012).

164. Mankos, M. *et al.* Electron Optics for a Multi-Pass Transmission Electron Microscope. *Adv. Imaging Electron Phys.* **212**, 71–86 (2019).

165. Koppell, S. A. *et al.* Design for a 10 keV Multi-Pass Transmission Electron Microscope. *Ultramicroscopy* **207**, 112834 (2019).

166. Juffmann, T. *et al.* Multi-Pass Transmission Electron Microscopy. *Sci. Reports* **7**, 1–7 (2017).

167. Jones, L. *et al.* Managing Dose-, Damage- and Data-Rates in Multi-Frame Spectrum-Imaging. *Microscopy* **67**, i98–i113 (2018).

168. Krull, A., Buchholz, T.-O. & Jug, F. Noise2Void - Learning Denoising from Single Noisy Images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2129–2137 (2019).

169. Guo, S., Yan, Z., Zhang, K., Zuo, W. & Zhang, L. Toward Convolutional Blind Denoising of Real Photographs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1712–1722 (2019).

170. Lefkimmiatis, S. Universal Denoising Networks: A Novel CNN Architecture for Image Denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3204–3213 (2018).

171. Weigert, M. *et al.* Content-Aware Image Restoration: Pushing the Limits of Fluorescence Microscopy. *Nat. Methods* **15**, 1090–1097 (2018).

172. Zhang, K., Zuo, W. & Zhang, L. FFDNet: Toward a Fast and Flexible Solution for CNN-Based Image Denoising. *IEEE Transactions on Image Process.* **27**, 4608–4622 (2018).

173. Weigert, M., Royer, L., Jug, F. & Myers, G. Isotropic Reconstruction of 3D Fluorescence Microscopy Images Using Convolutional Neural Networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 126–134 (Springer, 2017).

174. Zhang, K., Zuo, W., Chen, Y., Meng, D. & Zhang, L. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *IEEE Transactions on Image Process.* **26**, 3142–3155 (2017).

175. Tai, Y., Yang, J., Liu, X. & Xu, C. MemNet: A Persistent Memory Network for Image Restoration. In *Proceedings of the IEEE International Conference on Computer Vision*, 4539–4547 (2017).

176. Mao, X., Shen, C. & Yang, Y.-B. Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections. In *Advances in Neural Information Processing Systems*, 2802–2810 (2016).

177. Buchholz, T.-O., Jordan, M., Pigino, G. & Jug, F. Cryo-CARE: Content-Aware Image Restoration for Cryo-Transmission Electron Microscopy Data. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, 502–506 (IEEE, 2019).

178. Fang, L. *et al.* Deep Learning-Based Point-Scanning Super-Resolution Imaging. *bioRxiv* 740548 (2019).

179. Mohan, S. *et al.* Deep Denoising For Scientific Discovery: A Case Study In Electron Microscopy. *arXiv preprint arXiv:2010.12970* (2020).

180. Giannatou, E., Papavieros, G., Constantoudis, V., Papageorgiou, H. & Gogolides, E. Deep Learning Denoising of SEM Images Towards Noise-Reduced LER Measurements. *Microelectron. Eng.* **216**, 111051 (2019).

181. Chaudhary, N., Savari, S. A. & Yeddulapalli, S. S. Line Roughness Estimation and Poisson Denoising in Scanning Electron Microscope Images Using Deep Learning. *J. Micro/Nanolithography, MEMS, MOEMS* **18**, 024001 (2019).

182. Vasudevan, R. K. & Jesse, S. Deep Learning as a Tool for Image Denoising and Drift Correction. *Microsc. Microanal.* **25**, 190–191 (2019).

183. Wang, F., Henninen, T. R., Keller, D. & Erni, R. Noise2Atom: Unsupervised Denoising for Scanning Transmission Electron Microscopy Images. *Res. Sq.* DOI: 10.21203/rs.3.rs-54657/v1 (2020).

184. Bepler, T., Noble, A. J. & Berger, B. Topaz-Denoise: General Deep Denoising Models for CryoEM. *bioRxiv* 838920 (2019).

185. Lehtinen, J. *et al.* Noise2Noise: Learning Image Restoration without Clean Data. In *International Conference on Machine Learning*, 2965–2974 (2018).

186. Tegunov, D. & Cramer, P. Real-Time Cryo-Electron Microscopy Data Preprocessing with Warp. *Nat. Methods* **16**, 1146–1152 (2019).

187. Zhang, C., Berkels, B., Wirth, B. & Voyles, P. M. Joint Denoising and Distortion Correction for Atomic Column Detection in Scanning Transmission Electron Microscopy Images. *Microsc. Microanal.* **23**, 164–165 (2017).

188. Jin, P. & Li, X. Correction of Image Drift and Distortion in a Scanning Electron Microscopy. *J. Microsc.* **260**, 268–280 (2015).

189. Tong, X. *et al.* Image Registration with Fourier-Based Image Correlation: A Comprehensive Review of Developments and Applications. *IEEE J. Sel. Top. Appl. Earth Obs. Remote. Sens.* **12**, 4062–4081 (2019).

190. Krishnan, A. P. *et al.* Optical aberration correction via phase diversity and deep learning. *bioRxiv* (2020).

191. Cumming, B. P. & Gu, M. Direct Determination of Aberration Functions in Microscopy by an Artificial Neural Network. *Opt. Express* **28**, 14511–14521 (2020).

192. Wang, W., Wu, B., Zhang, B., Li, X. & Tan, J. Correction of Refractive Index Mismatch-Induced Aberrations Under Radially Polarized Illumination by Deep Learning. *Opt. Express* **28**, 26028–26040 (2020).

193. Tian, Q. *et al.* DNN-Based Aberration Correction in a Wavefront Sensorless Adaptive Optics System. *Opt. Express* **27**, 10765–10776 (2019).

194. Rivenson, Y. *et al.* Deep Learning Enhanced Mobile-Phone Microscopy. *Acs Photonics* **5**, 2354–2364 (2018).

195. Nguyen, T. *et al.* Automatic Phase Aberration Compensation for Digital Holographic Microscopy Based on Deep Learning Background Detection. *Opt. Express* **25**, 15043–15057 (2017).

196. Jeon, S. & Kim, C. Deep Learning-Based Speed of Sound Aberration Correction in Photoacoustic Images. In *Photons Plus Ultrasound: Imaging and Sensing 2020*, vol. 11240, 112400J (International Society for Optics and Photonics, 2020).

197. Gui, J., Sun, Z., Wen, Y., Tao, D. & Ye, J. A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications. *arXiv preprint arXiv:2001.06937* (2020).

198. Saxena, D. & Cao, J. Generative Adversarial Networks (GANs): Challenges, Solutions, and Future Directions. *arXiv preprint arXiv:2005.00065* (2020).

199. Pan, Z. *et al.* Recent Progress on Generative Adversarial Networks (GANs): A Survey. *IEEE Access* **7**, 36322–36333 (2019).

200. Wang, Z., She, Q. & Ward, T. E. Generative Adversarial Networks: A Survey and Taxonomy. *arXiv preprint arXiv:1906.01529* (2019).

201. Ede, J. M. & Beanland, R. Partial Scanning Transmission Electron Microscopy with Deep Learning. *Sci. Reports* **10**, 1–10 (2020).

202. Ede, J. M. Deep Learning Supersampled Scanning Transmission Electron Microscopy. *arXiv preprint arXiv:1910.10467* (2019).

203. Atta, R. E., Kasem, H. M. & Attia, M. A Comparison Study for Image Compression Based on Compressive Sensing. In *Eleventh International Conference on Graphics and Image Processing (ICGIP 2019)*, vol. 11373, 1137315 (International Society for Optics and Photonics, 2020).

204. Vidyasagar, M. *An Introduction to Compressed Sensing* (SIAM, 2019).

205. Rani, M., Dhok, S. B. & Deshmukh, R. A Systematic Review of Compressive Sensing: Concepts, Implementations and Applications. *IEEE Access* **6**, 4875–4894 (2018).

206. Eldar, Y. C. & Kutyniok, G. *Compressed Sensing: Theory and Applications* (Cambridge University Press, 2012).

207. Donoho, D. L. Compressed Sensing. *IEEE Transactions on Information Theory* **52**, 1289–1306 (2006).

208. Johnson, P. M., Recht, M. P. & Knoll, F. Improving the Speed of MRI with Artificial Intelligence. In *Seminars in Musculoskeletal Radiology*, vol. 24, 12 (NIH Public Access, 2020).

209. Ye, J. C. Compressed Sensing MRI: A Review from Signal Processing Perspective. *BMC Biomed. Eng.* **1**, 1–17 (2019).

210. Lustig, M., Donoho, D. & Pauly, J. M. Sparse MRI: The Application of Compressed Sensing for Rapid MR Imaging. *Magn. Reson. Medicine: An Off. J. Int. Soc. for Magn. Reson. Medicine* **58**, 1182–1195 (2007).

211. Yuan, X. & Haimi-Cohen, R. Image Compression Based on Compressive Sensing: End-to-end Comparison with JPEG. *IEEE Transactions on Multimed.* **22**, 2889–2904 (2020).

212. Gunasheela, S. & Prasantha, H. Compressed Sensing for Image Compression: Survey of Algorithms. In *Emerging Research in Computing, Information, Communication and Applications*, 507–517 (Springer, 2019).

213. Wang, Z., Chen, J. & Hoi, S. C. H. Deep Learning for Image Super-Resolution: A Survey. *IEEE Transactions on Pattern Analysis Mach. Intell.* (2020).

214. Yang, W. *et al.* Deep Learning for Single Image Super-Resolution: A Brief Review. *IEEE Transactions on Multimed.* **21**, 3106–3121 (2019).

215. Shin, Y. J. *et al.* Low-Dose Abdominal CT Using a Deep Learning-Based Denoising Algorithm: A Comparison with CT Reconstructed with Filtered Back Projection or Iterative Reconstruction Algorithm. *Korean J. Radiol.* **21**, 356–364 (2020).

216. Cong, W. *et al.* Deep-Learning-Based Breast CT for Radiation Dose Reduction. In *Developments in X-Ray Tomography XII*, vol. 11113, 111131L (International Society for Optics and Photonics, 2019).

217. Barkan, O., Weill, J., Averbuch, A. & Dekel, S. Adaptive Compressed Tomography Sensing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2195–2202 (2013).

218. Almasri, F. & Debeir, O. Robust Perceptual Night Vision in Thermal Colorization. *arXiv preprint arXiv:2003.02204* (2020).

219. Chen, C., Chen, Q., Xu, J. & Koltun, V. Learning to See in the Dark. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3291–3300 (2018).

220. Peet, M. J., Henderson, R. & Russo, C. J. The Energy Dependence of Contrast and Damage in Electron Cryomicroscopy of Biological Molecules. *Ultramicroscopy* **203**, 125–131 (2019).

221. Zhang, X. *et al.* Radiation Damage in Nanostructured Materials. *Prog. Mater. Sci.* **96**, 217–321 (2018).

222. Lehnert, T., Lehtinen, O., Algara-Siller, G. & Kaiser, U. Electron Radiation Damage Mechanisms in 2D $MoSe_2$. *Appl. Phys. Lett.* **110**, 033106 (2017).

223. Hermannsdörfer, J., Tinnemann, V., Peckys, D. B. & de Jonge, N. The Effect of Electron Beam Irradiation in Environmental Scanning Transmission Electron Microscopy of Whole Cells in Liquid. *Microsc. Microanal.* **22**, 656–665 (2016).

224. Johnston-Peck, A. C., DuChene, J. S., Roberts, A. D., Wei, W. D. & Herzing, A. A. Dose-Rate-Dependent Damage of Cerium Dioxide in the Scanning Transmission Electron Microscope. *Ultramicroscopy* **170**, 1–9 (2016).

225. Jenkins, M. L. & Kirk, M. A. *Characterisation of Radiation Damage by Transmission Electron Microscopy* (CRC Press, 2000).

226. Egerton, R. F., Li, P. & Malac, M. Radiation Damage in the TEM and SEM. *Micron* **35**, 399–409 (2004).

227. S'ari, M., Cattle, J., Hondow, N., Brydson, R. & Brown, A. Low Dose Scanning Transmission Electron Microscopy of Organic Crystals by Scanning Moiré Fringes. *Micron* **120**, 1–9 (2019).

228. Mayoral, A., Mahugo, R., Sánchez-Sánchez, M. & Díaz, I. Cs-Corrected STEM Imaging of Both Pure and Silver-Supported Metal-Organic Framework MIL-100 (Fe). *ChemCatChem* **9**, 3497–3502 (2017).

229. Gnanasekaran, K., de With, G. & Friedrich, H. Quantification and Optimization of ADF-STEM Image Contrast for Beam-Sensitive Materials. *Royal Soc. Open Sci.* **5**, 171838 (2018).

230. Ilett, M., Brydson, R., Brown, A. & Hondow, N. Cryo-Analytical STEM of Frozen, Aqueous Dispersions of Nanoparticles. *Micron* **120**, 35–42 (2019).

231. Ede, J. M. Warwick Electron Microscopy Datasets. *Mach. Learn. Sci. Technol.* **1**, 045003 (2020).

232. Landau, H. J. Sampling, Data Transmission, and the Nyquist Rate. *Proc. IEEE* **55**, 1701–1706 (1967).

233. Amidror, I. Sub-Nyquist Artefacts and Sampling Moiré effects. *Royal Soc. Open Sci.* **2**, 140550 (2015).

234. Fadnavis, S. Image Interpolation Techniques in Digital Image Processing: An Overview. *Int. J. Eng. Res. Appl.* **4**, 70–73 (2014).

235. Getreuer, P. Linear Methods for Image Interpolation. *Image Process. On Line* **1**, 238–259 (2011).

236. Turkowski, K. Filters for Common Resampling Tasks. In *Graphics Gems*, 147–165 (Morgan Kaufmann, 1990).

237. Beretta, L. & Santaniello, A. Nearest Neighbor Imputation Algorithms: A Critical Evaluation. *BMC Med. Informatics Decis. Mak.* **16**, 74 (2016).

238. Alfeld, P. A Trivariate Clough—Tocher Scheme for Tetrahedral Data. *Comput. Aided Geom. Des.* **1**, 169–181 (1984).

239. Cruz, C., Mehta, R., Katkovnik, V. & Egiazarian, K. O. Single Image Super-Resolution Based on Wiener Filter in Similarity Domain. *IEEE Transactions on Image Process.* **27**, 1376–1389 (2017).

240. Zulkifli, N., Karim, S., Shafie, A. & Sarfraz, M. Rational Bicubic Ball for Image Interpolation. In *Journal of Physics: Conference Series*, vol. 1366, 012097 (IOP Publishing, 2019).

241. Costella, J. The Magic Kernel. Towards Data Science, Online: https://web.archive.org/web/20170707165835/http://johncostella.webs.com/magic (2017).

242. Olivier, R. & Hanqiang, C. Nearest Neighbor Value Interpolation. *Int. J. Adv. Comput. Sci. Appl.* **3**, 25–30 (2012).

243. Jones, L. *et al.* Managing Dose-, Damage- and Data-Rates in Multi-Frame Spectrum-Imaging. *Microscopy* **67**, i98–i113 (2018).

244. Trampert, P. *et al.* How Should a Fixed Budget of Dwell Time be Spent in Scanning Electron Microscopy to Optimize Image Quality? *Ultramicroscopy* **191**, 11–17 (2018).

245. Stevens, A. *et al.* A Sub-Sampled Approach to Extremely Low-Dose STEM. *Appl. Phys. Lett.* **112**, 043104 (2018).

246. Hwang, S., Han, C. W., Venkatakrishnan, S. V., Bouman, C. A. & Ortalan, V. Towards the Low-Dose Characterization of Beam Sensitive Nanostructures via Implementation of Sparse Image Acquisition in Scanning Transmission Electron Microscopy. *Meas. Sci. Technol.* **28**, 045402 (2017).

247. Hujsak, K., Myers, B. D., Roth, E., Li, Y. & Dravid, V. P. Suppressing Electron Exposure Artifacts: An Electron Scanning Paradigm with Bayesian Machine Learning. *Microsc. Microanal.* **22**, 778–788 (2016).

248. Anderson, H. S., Ilic-Helms, J., Rohrer, B., Wheeler, J. & Larson, K. Sparse Imaging for Fast Electron Microscopy. In *Computational Imaging XI*, vol. 8657, 86570C (International Society for Optics and Photonics, 2013).

249. Stevens, A., Yang, H., Carin, L., Arslan, I. & Browning, N. D. The Potential for Bayesian Compressive Sensing to Significantly Reduce Electron Dose in High-Resolution STEM Images. *Microscopy* **63**, 41–51 (2013).

250. Candes, E. & Romberg, J. Sparsity and Incoherence in Compressive Sampling. *Inverse Probl.* **23**, 969 (2007).

251. Kovarik, L., Stevens, A., Liyu, A. & Browning, N. D. Implementing an Accurate and Rapid Sparse Sampling Approach for Low-Dose Atomic Resolution STEM Imaging. *Appl. Phys. Lett.* **109**, 164102 (2016).

252. Béché, A., Goris, B., Freitag, B. & Verbeeck, J. Development of a Fast Electromagnetic Beam Blanker for Compressed Sensing in Scanning Transmission Electron Microscopy. *Appl. Phys. Lett.* **108**, 093103 (2016).

253. Li, X., Dyck, O., Kalinin, S. V. & Jesse, S. Compressed Sensing of Scanning Transmission Electron Microscopy (STEM) with Nonrectangular Scans. *Microsc. Microanal.* **24**, 623–633 (2018).

254. Sang, X. *et al.* Precision Controlled Atomic Resolution Scanning Transmission Electron Microscopy Using Spiral Scan Pathways. *Sci. Reports* **7**, 43585 (2017).

255. Gandhare, S. & Karthikeyan, B. Survey on FPGA Architecture and Recent Applications. In *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, 1–4 (IEEE, 2019).

256. Qiao, M., Meng, Z., Ma, J. & Yuan, X. Deep Learning for Video Compressive Sensing. *APL Photonics* **5**, 030801 (2020).

257. Wu, Y., Rosca, M. & Lillicrap, T. Deep Compressed Sensing. *arXiv preprint arXiv:1905.06723* (2019).

258. Adler, A., Boublil, D. & Zibulevsky, M. Block-Based Compressed Sensing of Images via Deep Learning. In *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*, 1–6 (IEEE, 2017).

259. de Haan, K., Ballard, Z. S., Rivenson, Y., Wu, Y. & Ozcan, A. Resolution Enhancement in Scanning Electron Microscopy Using Deep Learning. *Sci. Reports* **9**, 1–7 (2019).

260. Gao, Z., Ma, W., Huang, S., Hua, P. & Lan, C. Deep Learning for Super-Resolution in a Field Emission Scanning Electron Microscope. *AI* **1**, 1–10 (2020).

261. Ede, J. M. & Beanland, R. Adaptive Learning Rate Clipping Stabilizes Learning. *Mach. Learn. Sci. Technol.* **1**, 015011 (2020).

262. Suveer, A., Gupta, A., Kylberg, G. & Sintorn, I.-M. Super-Resolution Reconstruction of Transmission Electron Microscopy Images Using Deep Learning. In *2019 IEEE 16th International Symposium on Biomedical Imaging*, 548–551 (IEEE, 2019).

263. Ahmed, M. W. & Abdulla, A. A. Quality Improvement for Exemplar-based Image Inpainting Using a Modified Searching Mechanism. *UHD J. Sci. Technol.* **4**, 1–8 (2020).

264. Pinjarkar, A. V. & Tuptewar, D. Robust Exemplar-Based Image and Video Inpainting for Object Removal and Region Filling. In *Computing, Communication and Signal Processing*, 817–825 (Springer, 2019).

265. Zhang, N., Ji, H., Liu, L. & Wang, G. Exemplar-Based Image Inpainting Using Angle-Aware Patch Matching. *EURASIP J. on Image Video Process.* **2019**, 70 (2019).

266. Criminisi, A., Pérez, P. & Toyama, K. Region Filling and Object Removal by Exemplar-Based Image Inpainting. *IEEE Transactions on Image Process.* **13**, 1200–1212 (2004).

267. Lu, M. & Niu, S. A Detection Approach Using LSTM-CNN for Object Removal Caused by Exemplar-Based Image Inpainting. *Electronics* **9**, 858 (2020).

268. Telea, A. An Image Inpainting Technique Based on the Fast Marching Method. *J. Graph. Tools* **9**, 23–34 (2004).

269. Bertalmio, M., Bertozzi, A. L. & Sapiro, G. Navier-Stokes, Fluid Dynamics, and Image and Video Inpainting. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, I–I (IEEE, 2001).

270. He, T. *et al.* Bag of Tricks for Image Classification with Convolutional Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 558–567 (2019).

271. Sun, Y., Xue, B., Zhang, M. & Yen, G. G. Evolving Deep Convolutional Neural Networks for Image Classification. *IEEE Transactions on Evol. Comput.* **24**, 394–407 (2019).

272. Rawat, W. & Wang, Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Comput.* **29**, 2352–2449 (2017).

273. Druzhkov, P. N. & Kustikova, V. D. A Survey of Deep Learning Methods and Software Tools for Image Classification and Object Detection. *Pattern Recognit. Image Analysis* **26**, 9–15 (2016).

274. Yokoyama, Y. *et al.* Development of a Deep Learning-Based Method to Identify “Good” Regions of a Cryo-Electron Microscopy Grid. *Biophys. Rev.* **12**, 349–354 (2020).

275. Sanchez-Garcia, R., Segura, J., Maluenda, D., Sorzano, C. & Carazo, J. MicrographCleaner: A Python Package for Cryo-EM Micrograph Cleaning Using Deep Learning. *J. Struct. Biol.* 107498 (2020).

276. Aguiar, J., Gong, M., Unocic, R., Tasdizen, T. & Miller, B. Decoding Crystallography from High-Resolution Electron Imaging and Diffraction Datasets with Deep Learning. *Sci. Adv.* **5**, eaaw1949 (2019).

277. Vasudevan, R. K. *et al.* Mapping Mesoscopic Phase Evolution During E-Beam Induced Transformations via Deep Learning of Atomically Resolved Images. *npj Comput. Mater.* **4** (2018).

278. Avramov, T. K. *et al.* Deep Learning for Validating and Estimating Resolution of Cryo-Electron Microscopy Density Maps. *Molecules* **24**, 1181 (2019).

279. Koch, G., Zemel, R. & Salakhutdinov, R. Siamese Neural Networks for One-Shot Image Recognition. In *ICML Deep Learning Workshop*, vol. 2 (Lille, 2015).

280. Chopra, S., Hadsell, R. & LeCun, Y. Learning a Similarity Metric Discriminatively, with Application to Face Verification In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 539–546 (IEEE, 2005).

281. Bromley, J., Guyon, I., LeCun, Y., Säckinger, E. & Shah, R. Signature Verification Using a "Siamese" Time Delay Neural Network. In *Advances in Neural Information Processing Systems*, 737–744 (1994).

282. Cai, Q., Pan, Y., Yao, T., Yan, C. & Mei, T. Memory Matching Networks for One-Shot Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4080–4088 (2018).

283. Li, X. *et al.* Predicting the Effective Mechanical Property of Heterogeneous Materials by Image Based Modeling and Deep Learning. *Comput. Methods Appl. Mech. Eng.* **347**, 735–753 (2019).

284. Sanchez-Garcia, R., Segura, J., Maluenda, D., Carazo, J. M. & Sorzano, C. O. S. Deep Consensus, A Deep Learning-Based Approach for Particle Pruning in Cryo-Electron Microscopy. *IUCrJ* **5**, 854–865 (2018).

285. Wang, F. *et al.* DeepPicker: A Deep Learning Approach for Fully Automated Particle Picking in Cryo-EM. *J. Struct Biol.* **195**, 325–336 (2016).

286. George, B. *et al.* CASSPER: A Semantic Segmentation Based Particle Picking Algorithm for Single Particle Cryo-Electron Microscopy. *bioRxiv* (2020).

287. Roberts, G. *et al.* Deep Learning for Semantic Segmentation of Defects in Advanced STEM Images of Steels. *Sci Reports* **9**, 1–12 (2019).

288. Madsen, J. *et al.* A Deep Learning Approach to Identify Local Structures in Atomic-Resolution Transmission Electron Microscopy Images. *Adv. Theory Simulations* **1**, 1800037 (2018).

289. Ziatdinov, M. *et al.* Deep Learning of Atomically Resolved Scanning Transmission Electron Microscopy Images: Chemical Identification and Tracking Local Transformations. *ACS Nano* **11**, 12742–12752 (2017).

290. Ziatdinov, M. *et al.* Building and Exploring Libraries of Atomic Defects in Graphene: Scanning Transmission Electron and Scanning Tunneling Microscopy Study. *Sci. Adv.* **5**, eaaw8989 (2019).

291. Meyer, J. C. *et al.* Direct Imaging of Lattice Atoms and Topological Defects in Graphene Membranes. *Nano Lett.* **8**, 3582–3586 (2008).

292. Meyer, J. C. *et al.* Experimental Analysis of Charge Redistribution Due to Chemical Bonding by High-Resolution Transmission Electron Microscopy. *Nat. Mater.* **10**, 209–215 (2011).

293. He, X. *et al.* In Situ Atom Scale Visualization of Domain Wall Dynamics in VO_2 Insulator-Metal Phase Transition. *Sci Reports* **4**, 6544 (2014).

294. Nagao, K., Inuzuka, T., Nishimoto, K. & Edagawa, K. Experimental Observation of Quasicrystal Growth. *Phys. Rev Lett.* **115**, 075501 (2015).

295. Li, X. *et al.* Direct Observation of the Layer-by-Layer Growth of ZnO Nanopillar by In Situ High Resolution Transmission Electron Microscopy. *Sci. Reports* **7**, 40911 (2017).

296. Schneider, S., Surrey, A., Pohl, D., Schultz, L. & Rellinghaus, B. Atomic Surface Diffusion on Pt Nanoparticles Quantified by High-Resolution Transmission Electron Microscopy. *Micron* **63**, 52–56 (2014).

297. Hussaini, Z., Lin, P. A., Natarajan, B., Zhu, W. & Sharma, R. Determination of Atomic Positions from Time Resolved High Resolution Transmission Electron Microscopy Images. *Ultramicroscopy* **186**, 139–145 (2018).

298. Pham, D. L., Xu, C. & Prince, J. L. Current Methods in Medical Image Segmentation. *Annu. Rev. Biomed. Eng.* **2**, 315–337 (2000).

299. Mesejo, P., Valsecchi, A., Marrakchi-Kacem, L., Cagnoni, S. & Damas, S. Biomedical Image Segmentation Using Geometric Deformable Models and Metaheuristics. *Comput. Med. Imaging Graph.* **43**, 167–178 (2015).

300. Zheng, Y., Jeon, B., Xu, D., Wu, Q. M. & Zhang, H. Image Segmentation by Generalized Hierarchical Fuzzy C-Means Algorithm. *J. Intell. & Fuzzy Syst.* **28**, 961–973 (2015).

301. Hao, S., Zhou, Y. & Guo, Y. A Brief Survey on Semantic Segmentation with Deep Learning. *Neurocomputing* **406**, 302–321 (2020).

302. Sultana, F., Sufian, A. & Dutta, P. Evolution of Image Segmentation Using Deep Convolutional Neural Network: A Survey. *Knowledge-Based Syst.* **201–202**, 106062 (2020).

303. Minaee, S. *et al.* Image segmentation Using deep learning: A survey. *arXiv preprint arXiv:2001.05566* (2020).

304. Guo, Y., Liu, Y., Georgiou, T. & Lew, M. S. A Review of Semantic Segmentation using Deep Neural Networks. *Int. J. Multimed. Inf. Retr.* **7**, 87–93 (2018).

305. Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F. & Adam, H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 801–818 (2018).

306. Chen, L.-C., Papandreou, G., Schroff, F. & Adam, H. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv preprint arXiv:1706.05587* (2017).

307. Badrinarayanan, V., Kendall, A. & Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis Mach. Intell.* **39**, 2481–2495 (2017).

308. Ronneberger, O., Fischer, P. & Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 234–241 (Springer, 2015).

309. Yi, J., Yuan, Z. & Peng, J. Adversarial-Prediction Guided Multi-Task Adaptation for Semantic Segmentation of Electron Microscopy Images. In *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, 1205–1208 (IEEE, 2020).

310. Khadangi, A., Boudier, T. & Rajagopal, V. EM-net: Deep Learning for Electron Microscopy Image Segmentation. *bioRxiv* (2020).

311. Roels, J. & Saeys, Y. Cost-Efficient Segmentation of Electron Microscopy Images Using Active Learning. *arXiv preprint arXiv:1911.05548* (2019).

312. Yu, Z. X. *et al.* High-Throughput, Algorithmic Determination of Pore Parameters from Electron Microscopy. *Comput. Mater. Sci.* **171**, 109216 (2020).

313. Fakhry, A., Zeng, T. & Ji, S. Residual Deconvolutional Networks for Brain Electron Microscopy Image Segmentation. *IEEE Transactions on Med. Imaging* **36**, 447–456 (2016).

314. Urakubo, H., Bullmann, T., Kubota, Y., Oba, S. & Ishii, S. UNI-EM: An Environment for Deep Neural Network-Based Automated Segmentation of Neuronal Electron Microscopic Images. *Sci. Reports* **9**, 1–9 (2019).

315. Roberts, G. *et al.* DefectNet – A Deep Convolutional Neural Network for Semantic Segmentation of Crystallographic Defects in Advanced Microscopy Images. *Microsc. Microanal.* **25**, 164–165 (2019).

316. Ibtehaz, N. & Rahman, M. S. MultiResUNet: Rethinking the U-Net Architecture for Multimodal Biomedical Image Segmentation. *Neural Networks* **121**, 74–87 (2020).

317. Groschner, C. K., Choi, C. & Scott, M. Methodologies for Successful Segmentation of HRTEM Images via Neural Network. *arXiv preprint arXiv:2001.05022* (2020).

318. Horwath, J. P., Zakharov, D. N., Megret, R. & Stach, E. A. Understanding Important Features of Deep Learning Models for Transmission Electron Microscopy Image Segmentation. *arXiv preprint arXiv:1912.06077* (2019).

319. Chen, M. *et al.* Convolutional Neural Networks for Automated Annotation of Cellular Cryo-Electron Tomograms. *Nat. Methods* **14**, 983 (2017).

320. Feng, D. *et al.* Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges. *IEEE Transactions on Intell. Transp. Syst.* (2020).

321. Yang, K., Bi, S. & Dong, M. Lightningnet: Fast and Accurate Semantic Segmentation for Autonomous Driving Based on 3D LIDAR Point Cloud. In *2020 IEEE International Conference on Multimedia and Expo*, 1–6 (IEEE, 2020).

322. Hofmarcher, M. *et al.* Visual Scene Understanding for Autonomous Driving Using Semantic Segmentation. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, 285–296 (Springer, 2019).

323. Blum, H., Sarlin, P.-E., Nieto, J., Siegwart, R. & Cadena, C. Fishyscapes: A Benchmark for Safe Semantic Segmentation in Autonomous Driving. In *Proceedings of the IEEE International Conference on Computer Vision Workshops* (2019).

324. Zhou, W., Berrio, J. S., Worrall, S. & Nebot, E. Automated Evaluation of Semantic Segmentation Robustness for Autonomous Driving. *IEEE Transactions on Intell. Transp. Syst.* **21**, 1951–1963 (2019).

325. Pfisterer, K. J. *et al.* Fully-Automatic Semantic Segmentation for Food Intake Tracking in Long-Term Care Homes. *arXiv preprint arXiv:1910.11250* (2019).

326. Aslan, S., Ciocca, G. & Schettini, R. Semantic Food Segmentation for Automatic Dietary Monitoring. In *2018 IEEE 8th International Conference on Consumer Electronics-Berlin*, 1–6 (IEEE, 2018).

327. Ghosh, S., Ray, N., Boulanger, P., Punithakumar, K. & Noga, M. Automated Left Atrial Segmentation from Magnetic Resonance Image Sequences Using Deep Convolutional Neural Network with Autoencoder. In *2020 IEEE 17th International Symposium on Biomedical Imaging*, 1756–1760 (IEEE, 2020).

328. Memis, A., Varli, S. & Bilgili, F. Semantic Segmentation of the Multiform Proximal Femur and Femoral Head Bones with the Deep Convolutional Neural Networks in Low Quality MRI Sections Acquired in Different MRI Protocols. *Comput. Med. Imaging Graph.* **81**, 101715 (2020).

329. Duran, A., Jodoin, P.-M. & Lartizien, C. Prostate Cancer Semantic Segmentation by Gleason Score Group in mp-MRI with Self Attention Model on the Peripheral Zone. In *Medical Imaging with Deep Learning* (2020).

330. Bevilacqua, V. *et al.* A Comparison Between Two Semantic Deep Learning Frameworks for the Autosomal Dominant Polycystic Kidney Disease Segmentation Based on Magnetic Resonance Images. *BMC Med. Informatics Decis. Mak.* **19**, 1–12 (2019).

331. Liu, F. *et al.* Deep Convolutional Neural Network and 3D Deformable Approach for Tissue Segmentation in Musculoskeletal Magnetic Resonance Imaging. *Magn. Reson. Medicine* **79**, 2379–2391 (2018).

332. Taghanaki, S. A., Abhishek, K., Cohen, J. P., Cohen-Adad, J. & Hamarneh, G. Deep Semantic Segmentation of Natural and Medical Images: A Review. *Artif. Intell. Rev.* (2020).

333. Tajbakhsh, N. *et al.* Embracing Imperfect Datasets: A Review of Deep Learning Solutions for Medical Image Segmentation. *Med. Image Analysis* **63**, 101693 (2020).

334. Du, G., Cao, X., Liang, J., Chen, X. & Zhan, Y. Medical Image Segmentation Based on U-Net: A Review. *J. Imaging Sci. Technol.* **64**, 20508–1 (2020).

335. Yang, X. *et al.* Hybrid Attention for Automatic Segmentation of Whole Fetal Head in Prenatal Ultrasound Volumes. *Comput. Methods Programs Biomed.* **194**, 105519 (2020).

336. Wang, X. *et al.* Joint Segmentation and Landmark Localization of Fetal Femur in Ultrasound Volumes. In *2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, 1–5 (IEEE, 2019).

337. Venturini, L., Papageorghiou, A. T., Noble, J. A. & Namburete, A. I. Multi-task CNN for Structural Semantic Segmentation in 3D Fetal Brain Ultrasound. In *Annual Conference on Medical Image Understanding and Analysis*, 164–173 (Springer, 2019).

338. Yang, X. *et al.* Towards Automated Semantic Segmentation in Prenatal Volumetric Ultrasound. *IEEE Transactions on Med. Imaging* **38**, 180–193 (2018).

339. Tasar, O., Tarabalka, Y., Giros, A., Alliez, P. & Clerc, S. StandardGAN: Multi-source Domain Adaptation for Semantic Segmentation of Very High Resolution Satellite Images by Data Standardization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 192–193 (2020).

340. Barthakur, M. & Sarma, K. K. Deep Learning Based Semantic Segmentation Applied to Satellite Image. In *Data Visualization and Knowledge Engineering*, 79–107 (Springer, 2020).

341. Wu, M., Zhang, C., Liu, J., Zhou, L. & Li, X. Towards Accurate High Resolution Satellite Image Semantic Segmentation. *IEEE Access* **7**, 55609–55619 (2019).

342. Wurm, M., Stark, T., Zhu, X. X., Weigand, M. & Taubenböck, H. Semantic Segmentation of Slums in Satellite Images Using Transfer Learning on Fully Convolutional Neural Networks. *ISPRS J. Photogramm. Remote. Sens.* **150**, 59–69 (2019).

343. Zhou, L., Zhang, C. & Wu, M. D-LinkNet: LinkNet With Pretrained Encoder and Dilated Convolution for High Resolution Satellite Imagery Road Extraction. In *CVPR Workshops*, 182–186 (2018).

344. Joyce, T., Chartsias, A. & Tsaftaris, S. A. Deep Multi-Class Segmentation Without Ground-Truth Labels. In *1st Conference on Medical Imaging with Deep Learning* (2018).

345. Araslanov, N. & Roth, S. Single-Stage Semantic Segmentation from Image Labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4253–4262 (2020).

346. Chen, Z., Tian, Z., Li, X., Zhang, Y. & Dormer, J. D. Exploiting Confident Information for Weakly Supervised Prostate Segmentation Based on Image-Level Labels. In *Medical Imaging 2020: Image-Guided Procedures, Robotic Interventions, and Modeling*, vol. 11315, 1131523 (International Society for Optics and Photonics, 2020).

347. Jing, L., Chen, Y. & Tian, Y. Coarse-to-Fine Semantic Segmentation from Image-Level Labels. *IEEE Transactions on Image Process.* **29**, 225–236 (2019).

348. Oh, S. J. *et al.* Exploiting Saliency for Object Segmentation from Image Level Labels. In *IEEE Conference on Computer Vision and Pattern Recognition*, 5038–5047 (IEEE, 2017).

349. Ede, J. M., Peters, J. J. P., Sloan, J. & Beanland, R. Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning. *arXiv preprint arXiv:2001.10938* (2020).

350. Frabboni, S., Gazzadi, G. C. & Pozzi, G. Young's Double-Slit Interference Experiment with Electrons. *Am. J. Phys.* **75**, 1053–1055 (2007).

351. Matteucci, G. & Beeli, C. An Experiment on Electron Wave-Particle Duality Including a Planck Constant Measurement. *Am. J. Phys.* **66**, 1055–1059 (1998).

352. Lehmann, M. & Lichte, H. Tutorial on Off-Axis Electron Holography. *Microsc. Microanal.* **8**, 447–466 (2002).

353. Tonomura, A. Applications of Electron Holography. *Rev. Mod. Phys.* **59**, 639 (1987).

354. Lentzen, M. & Urban, K. Reconstruction of the Projected Crystal Potential in Transmission Electron Microscopy by Means of a Maximum-Likelihood Refinement Algorithm. *Acta Crystallogr. Sect. A: Foundations Crystallogr.* **56**, 235–247 (2000).

355. Auslender, A., Halabi, M., Levi, G., Diéguez, O. & Kohn, A. Measuring the Mean Inner Potential of Al_2O_3 Sapphire Using Off-Axis Electron Holography. *Ultramicroscopy* **198**, 18–25 (2019).

356. Fu, Q., Lichte, H. & Völkl, E. Correction of Aberrations of an Electron Microscope by Means of Electron Holography. *Phys. Rev. Lett.* **67**, 2319 (1991).

357. McCartney, M. R. & Gajdardziska-Josifovska, M. Absolute Measurement of Normalized Thickness, t/λ_i, from Off-Axis Electron Holography. *Ultramicroscopy* **53**, 283–289 (1994).

358. Park, H. S. *et al.* Observation of the Magnetic Flux and Three-Dimensional Structure of Skyrmion Lattices by Electron Holography. *Nat. Nanotechnol.* **9**, 337–342 (2014).

359. Dunin-Borkowski, R. E. *et al.* Off-Axis Electron Holography of Magnetic Nanowires and Chains, Rings, and Planar Arrays of Magnetic Nanoparticles. *Microsc. Res. Tech.* **64**, 390–402 (2004).

360. Lubk, A. *et al.* Fundamentals of Focal Series Inline Electron Holography. In *Advances in Imaging and Electron Physics*, vol. 197, 105–147 (Elsevier, 2016).

361. Koch, C. T. Towards Full-Resolution Inline Electron Holography. *Micron* **63**, 69–75 (2014).

362. Haigh, S. J., Jiang, B., Alloyeau, D., Kisielowski, C. & Kirkland, A. I. Recording Low and High Spatial Frequencies in Exit Wave Reconstructions. *Ultramicroscopy* **133**, 26–34 (2013).

363. Koch, C. T. & Lubk, A. Off-Axis and Inline Electron Holography: A Quantitative Comparison. *Ultramicroscopy* **110**, 460–471 (2010).

364. Van Dyck, D., de Beeck, M. O. & Coene, W. Object Wavefunction Reconstruction in High Resolution Electron Microscopy. In *Proceedings of 1st International Conference on Image Processing*, vol. 3, 295–298 (IEEE, 1994).

365. Ozsoy-Keskinbora, C., Boothroyd, C., Dunin-Borkowski, R., Van Aken, P. & Koch, C. Hybridization Approach to In-Line and Off-Axis (Electron) Holography for Superior Resolution and Phase Sensitivity. *Sci. Reports* **4**, 1–10 (2014).

366. Rivenson, Y., Zhang, Y., Günaydın, H., Teng, D. & Ozcan, A. Phase Recovery and Holographic Image Reconstruction Using Deep Learning in Neural Networks. *Light. Sci. & Appl.* **7**, 17141–17141 (2018).

367. Wu, Y. *et al.* Extended Depth-of-Field in Holographic Imaging Using Deep-Learning-Based AutofocUsing and Phase Recovery. *Optica* **5**, 704–710 (2018).

368. Sinha, A., Lee, J., Li, S. & Barbastathis, G. Lensless Computational Imaging Through Deep Learning. *Optica* **4**, 1117–1125 (2017).

369. Beach, M. J. *et al.* QuCumber: Wavefunction Reconstruction with Neural Networks. *arXiv preprint arXiv:1812.09329* (2018).

370. Dral, P. O. Quantum Chemistry in the Age of Machine Learning. *The J. Phys. Chem. Lett.* **11**, 2336–2347 (2020).

371. Liu, X. *et al.* Deep Learning for Feynman's Path Integral in Strong-Field Time-Dependent Dynamics. *Phys. Rev. Lett.* **124**, 113202 (2020).

372. Bharti, K., Haug, T., Vedral, V. & Kwek, L.-C. Machine Learning Meets Quantum Foundations: A Brief Survey. *arXiv preprint arXiv:2003.11224* (2020).

373. Carleo, G. *et al.* NetKet: A Machine Learning Toolkit for Many-Body Quantum Systems. *arXiv preprint arXiv:1904.00031* (2019).

374. Schütt, K., Gastegger, M., Tkatchenko, A., Müller, K.-R. & Maurer, R. J. Unifying Machine Learning and Quantum Chemistry with a Deep Neural Network for Molecular Wavefunctions. *Nat. Commun.* **10**, 1–10 (2019).

375. Laanait, N., He, Q. & Borisevich, A. Y. Reconstruction of 3-D Atomic Distortions from Electron Microscopy with Deep Learning. *arXiv preprint arXiv:1902.06876* (2019).

376. Morgan, A. J., Martin, A. V., D'Alfonso, A. J., Putkunz, C. T. & Allen, L. J. Direct Exit-Wave Reconstruction From a Single Defocused Image. *Ultramicroscopy* **111**, 1455–1460 (2011).

377. Martin, A. & Allen, L. Direct Retrieval of a Complex Wave From its Diffraction Pattern. *Opt. communications* **281**, 5114–5121 (2008).

378. Schlitz, M. Science Without Publication Paywalls a Preamble to: cOAlition S for the Realisation of Full and Immediate Open Access. *Sci. Eur.* (2018).

379. Coalition of European Funders Announces "Plan S" to Require Full OA, Cap APCs, & Disallow Publication in Hybrid Journals. SPARC, Online: https://sparcopen.org/news/2018/coalition-european-funders-announces-plan-s (2018).

380. cOAlition S. Plan S: Making Full and Immediate Open Access a Reality. Online: https://www.coalition-s.org (2020).

381. Banks, G. C. *et al.* Answers to 18 Questions About Open Science Practices. *J. Bus. Psychol.* **34**, 257–270 (2019).

382. Shi, R. *et al.* FTDL: An FPGA-Tailored Architecture for Deep Learning Systems. In *FPGA*, 320 (2020).

383. Kaarmukilan, S., Poddar, S. *et al.* FPGA Based Deep Learning Models for Object Detection and Recognition Comparison of Object Detection Comparison of Object Detection Models Using FPGA. In *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, 471–474 (IEEE, 2020).

384. Wang, T., Wang, C., Zhou, X. & Chen, H. An Overview of FPGA Based Deep Learning Accelerators: Challenges and Opportunities. In *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 1674–1681 (IEEE, 2019).

385. Guo, K., Zeng, S., Yu, J., Wang, Y. & Yang, H. [DL] A Survey of FPGA-Based Neural Network Inference Accelerators. *ACM Transactions on Reconfigurable Technol. Syst. (TRETS)* **12**, 1–26 (2019).

386. Cano, A. A Survey on Graphic Processing Unit Computing for Large-Scale Data Mining. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **8**, e1232 (2018).

387. Nvidia. Tesla V100 GPU Architecture Whitepaper. Online: https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf (2017).

388. Gaster, B. R. *Heterogeneous Computing with OpenCL, 2nd Edition* (Elsevier/Morgan Kaufmann, 2013).

389. Gordienko, Y. *et al.* Scaling Analysis of Specialized Tensor Processing Architectures for Deep Learning Models. In *Deep Learning: Concepts and Architectures*, 65–99 (Springer, 2020).

390. Jouppi, N., Young, C., Patil, N. & Patterson, D. Motivation for and Evaluation of the First Tensor Processing Unit. *IEEE Micro* **38**, 10–19 (2018).

391. Jouppi, N. P. *et al.* In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 1–12 (2017).

392. Mattson, P. *et al.* MLPerf Training Benchmark. *arXiv preprint arXiv:1910.01500* (2020).

393. MLPerf: Fair and Useful Benchmarks for Measuring Training and Inference Performance of ML Hardware, Software, and Services. Online: https://mlperf.org (2020).

394. Wang, Y. E., Wei, G.-Y. & Brooks, D. Benchmarking TPU, GPU, and CPU Platforms for Deep Learning. *arXiv preprint arXiv:1907.10701* (2019).

395. Wang, Y. *et al.* Performance and Power Evaluation of AI Accelerators for Training Deep Learning Models. *arXiv preprint arXiv:1909.06842* (2019).

396. Li, F., Ye, Y., Tian, Z. & Zhang, X. Cpu versus gpu: Which can perform matrix computation faster – performance comparison for basic linear algebra subprograms. *Neural Comput. Appl.* **31**, 4353–4365 (2019).

397. Awan, A. A., Subramoni, H. & Panda, D. K. An In-Depth Performance Characterization of CPU-and GPU-Based DNN Training on Modern Architectures. In *Proceedings of the Machine Learning on HPC Environments*, 1–8 (2017).

398. Nurvitadhi, E. *et al.* Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks? In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 5–14 (2017).

399. GPU vs FPGA Performance Comparison. Berten Digital Signal Processing, Online: http://www.bertendsp.com/pdf/whitepaper/BWP001_GPU_vs_FPGA_Performance_Comparison_v1.0.pdf (2016).

400. Nangia, R. & Shukla, N. K. Resource Utilization Optimization with Design Alternatives in FPGA Based Arithmetic Logic Unit Architectures. *Procedia Comput. Sci.* **132**, 843–848 (2018).

401. Grover, N. & Soni, M. Design of fpga based 32-bit floating point arithmetic unit and verification of its vhdl code using matlab. *Int. J. Inf. Eng. Electron. Bus.* **6**, 1 (2014).

402. Dolbeau, R. Theoretical Peak FLOPS Per Instruction Set: A Tutorial. *The J. Supercomput.* **74**, 1341–1377 (2018).

403. Strubell, E., Ganesh, A. & McCallum, A. Energy and Policy Considerations for Deep Learning in NLP. *arXiv preprint arXiv:1906.02243* (2019).

404. Nelson, M. J. & Hoover, A. K. Notes on Using Google Colaboratory in AI Education. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, 533–534 (2020).

405. Bisong, E. Google Colaboratory. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, 59–64 (Springer, 2019).

406. Tutorialspoint. Colab Tutorial. Online: https://www.tutorialspoint.com/google_colab/google_colab_tutorial.pdf (2019).

407. Carneiro, T. *et al.* Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. *IEEE Access* **6**, 61677–61685 (2018).

408. Kaggle Documentation. Online: https://www.kaggle.com/docs (2020).

409. Kalinin, S. V., Vasudevan, R. K. & Ziatdinov, M. Decoding the Relationship Between Domain Structure and Functionality in Ferroelectrics via Hidden Latent Variables. *arXiv preprint arXiv:2006.01374* (2020).

410. Green, O. How to Install a New Graphics Card – From Hardware to Drivers. Help Desk Geek, Online: https://helpdeskgeek.com/how-to/how-to-install-a-new-graphics-card-from-hardware-to-drivers (2019).

411. Ryan, T. How to Install a Graphics Card. PC World, Online: https://www.pcworld.com/article/2913370/how-to-install-a-graphics-card.html (2017).

412. Radecic, D. An Utterly Simple Guide on Installing Tensorflow-GPU 2.0 on Windows 10. Towards Data Science, Online: https://towardsdatascience.com/an-utterly-simple-guide-on-installing-tensorflow-gpu-2-0-on-windows-10-198368dc07a1 (2020).

413. Varile, M. Train Neural Networks Using AMD GPU and Keras. Towards Data Science, Online: https://towardsdatascience.com/train-neural-networks-Using-amd-gpus-and-keras-37189c453878 (2019).

414. Tim Dettmers. A Full Hardware Guide to Deep Learning. Online: https://timdettmers.com/2018/12/16/deep-learning-hardware-guide (2018).

415. Chetlur, S. *et al.* cuDNN: Efficient Primitives for Deep Learning. *arXiv preprint arXiv:1410.0759* (2014).

416. List of Cloud Services for Deep Learning. Online: https://github.com/zszazi/Deep-learning-in-cloud (2020).

417. Marozzo, F. Infrastructures for High-Performance Computing: Cloud Infrastructures. *Encycl. Bioinforma. Comput. Biol.* 240–246 (2019).

418. Joshi, N. & Shah, S. A Comprehensive Survey of Services Provided by Prevalent Cloud Computing Environments. In *Smart Intelligent Computing and Applications*, 413–424 (Springer, 2019).

419. Gupta, A., Goswami, P., Chaudhary, N. & Bansal, R. Deploying an Application Using Google Cloud Platform. In *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, 236–239 (IEEE, 2020).

420. Ooi, B. C. *et al.* SINGA: A Distributed Deep Learning Platform. In *Proceedings of the 23rd ACM international Conference on Multimedia*, 685–688 (2015).

421. Apache SINGA License. Online: https://github.com/apache/singa/blob/master/LICENSE (2020).

422. Dai, J. J. *et al.* BigDL: A Distributed Deep Learning Framework for Big Data. In *Proceedings of the ACM Symposium on Cloud Computing*, 50–60 (2019).

423. BigDL License. Online: https://github.com/intel-analytics/BigDL/blob/master/LICENSE (2020).

424. Jia, Y. *et al.* Caffe: Convolutional Architecture for Fast Feature Embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia*, 675–678 (2014).

425. Synced. Caffe2 Merges with PyTorch. Medium, Online: https://medium.com/@Synced/caffe2-merges-with-pytorch-a89c70ad9eb7 (2004).

426. Caffe License. Online: https://github.com/BVLC/caffe/blob/master/LICENSE (2017).

427. Tokui, S., Oono, K., Hido, S. & Clayton, J. Chainer: A Next-Generation Open Source Framework for Deep Learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in the Twenty-Ninth Annual Conference on Neural Information Processing Systems (NIPS)*, vol. 5, 1–6 (2015).

428. Chainer License. Online: https://docs.chainer.org/en/stable/license.html (2020).

429. Gibson, A. *et al.* Deeplearning4j: Distributed, Open-Source Deep Learning for Java and Scala on Hadoop and Spark. Towards Data Science, Online: https://deeplearning4j.org (2016).

430. Deeplearning4j License. Online: https://github.com/eclipse/deeplearning4j/blob/master/LICENSE (2020).

431. King, D. E. Dlib-ml: A Machine Learning Toolkit. *The J. Mach. Learn. Res.* **10**, 1755–1758 (2009).

432. Dlib C++ Library. Online: http://dlib.net (2020).

433. Dlib License. Online: https://github.com/davisking/dlib/blob/master/dlib/LICENSE.txt (2020).

434. Innes, M. Flux: Elegant Machine Learning with Julia. *J. Open Source Softw.* **3**, 602 (2018).

435. Flux License. Online: https://github.com/FluxML/Flux.jl/blob/master/LICENSE.md (2020).

436. Beale, M., Hagan, M. & Demuth, H. PDF Documentation: MATLAB Deep Learning Toolbox User's Guide. Online: https://uk.mathworks.com/help/deeplearning (2020).

437. MATLAB License. Online: https://mathworks.com/pricing-licensing.html (2020).

438. Seide, F. Keynote: The Computer Science Behind the Microsoft Cognitive Toolkit: An Open Source Large-Scale Deep Learning Toolkit for Windows and Linux. In *2017 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, xi–xi (IEEE, 2017).

439. CNTK License. Online: https://github.com/microsoft/CNTK/blob/master/LICENSE.md (2020).

440. Chen, T. *et al.* MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. *arXiv preprint arXiv:1512.01274* (2015).

441. MXNet License. Online: https://github.com/apache/incubator-mxnet/blob/master/LICENSE (2020).

442. OpenNN. Online: https://www.opennn.net (2020).

443. OpenNN License. Online: https://github.com/Artelnics/OpenNN/blob/master/LICENSE.txt (2020).

444. Ma, Y., Yu, D., Wu, T. & Wang, H. PaddlePaddle: An Open-Source Deep Learning Platform from Industrial Practice. *Front. Data Comput.* **1**, 105–115 (2019).

445. PaddlePaddle License. Online: https://github.com/PaddlePaddle/Paddle/blob/develop/LICENSE (2020).

446. Paszke, A. *et al.* PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, 8024–8035 (2019).

447. PyTorch License. Online: https://github.com/pytorch/pytorch/blob/master/LICENSE (2020).

448. Abadi, M. *et al.* TensorFlow: A System for Large-Scale Machine Learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 265–283 (2016).

449. Abadi, M. *et al.* TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv preprint arXiv:1603.04467* (2016).

450. TensorFlow License. Online: https://github.com/tensorflow/tensorflow/blob/master/LICENSE (2020).

451. Team, T. T. D. *et al.* Theano: A Python Framework for Fast Computation of Mathematical Expressions. *arXiv preprint arXiv:1605.02688* (2016).

452. Ketkar, N. Introduction to Theano. In *Deep Learning with Python*, 35–61 (Springer, 2017).

453. Theano License. Online: https://github.com/Theano/Theano/blob/master/doc/LICENSE.txt (2020).

454. Collobert, R., Bengio, S. & Mariéthoz, J. Torch: A Modular Machine Learning Software Library. Tech. Rep., Idiap (2002).

455. Torch License. Online: https://github.com/torch/torch7/blob/master/COPYRIGHT.txt (2020).

456. Mathematica Neural Networks Documentation. Online: https://reference.wolfram.com/language/guide/NeuralNetworks.html (2020).

457. Mathematica Licenses. Online: https://www.wolfram.com/legal (2020).

458. Li, M. *et al.* The Deep Learning Compiler: A Comprehensive Survey. *arXiv preprint arXiv:2002.03794* (2020).

459. Nguyen, G. *et al.* Machine Learning and Deep Learning Frameworks and Libraries for Large-Scale Data Mining: A Survey. *Artif. Intell. Rev.* **52**, 77–124 (2019).

460. Dai, W. & Berleant, D. Benchmarking Contemporary Deep Learning Hardware and Frameworks: A Survey of Qualitative Metrics. In *2019 IEEE First International Conference on Cognitive Machine Intelligence (CogMI)*, 148–155, DOI: 10.1109/CogMI48466.2019.00029 (IEEE, 2019).

461. Kharkovyna, O. Top 10 Best Deep Learning Frameworks in 2019. Towards Data Science, Online: https://towardsdatascience.com/top-10-best-deep-learning-frameworks-in-2019-5ccb90ea6de (2019).

462. Zacharias, J., Barz, M. & Sonntag, D. A Survey on Deep Learning Toolkits and Libraries for Intelligent User Interfaces. *arXiv preprint arXiv:1803.04818* (2018).

463. Parvat, A., Chavan, J., Kadam, S., Dev, S. & Pathak, V. A Survey of Deep-Learning Frameworks. In *2017 International Conference on Inventive Systems and Control (ICISC)*, 1–7 (IEEE, 2017).

464. Erickson, B. J., Korfiatis, P., Akkus, Z., Kline, T. & Philbrick, K. Toolkits and Libraries for Deep Learning. *J. Digit. Imaging* **30**, 400–405 (2017).

465. Baydin, A. G., Pearlmutter, B. A., Radul, A. A. & Siskind, J. M. Automatic Differentiation in Machine Learning: A Survey. *The J. Mach. Learn. Res.* **18**, 5595–5637 (2017).

466. Barham, P. & Isard, M. Machine Learning Systems are Stuck in a Rut. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, 177–183 (2019).

467. Afif, M., Said, Y. & Atri, M. Computer Vision Algorithms Acceleration Using Graphic Processors NVIDIA CUDA. *Clust. Comput.* 1–13 (2020).

468. Cook, S. *CUDA Programming: A Developer's Guide to Parallel Computing with GPUs* (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2012), 1st edn.

469. Nickolls, J., Buck, I., Garland, M. & Skadron, K. Scalable Parallel Programming with CUDA. *Queue* **6**, 40–53 (2008).

470. Jordà, M., Valero-Lara, P. & Peña, A. J. Performance Evaluation of cuDNN Convolution Algorithms on NVIDIA Volta GPUs. *IEEE Access* **7**, 70461–70473 (2019).

471. de Supinski, B. R. *et al.* The Ongoing Evolution of OpenMP. *Proc. IEEE* **106**, 2004–2019 (2018).

472. Dagum, L. & Menon, R. OpenMP: An Industry Standard API for Shared-Memory Programming. *IEEE Comput. Sci. Eng.* **5**, 46–55 (1998).

473. He, H. The State of Machine Learning Frameworks in 2019. The Gradient, Online: https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry (2019).

474. Papers With Code: Trends. https://paperswithcode.com/trends (2020).

475. TensorFlow Libraries and Extensions. Online: https://www.tensorflow.org/resources/libraries-extensions (2020).

476. Chollet, F. *et al.* Keras. Online: https://keras.io (2020).

477. Sonnet repository. Online: https://github.com/deepmind/sonnet (2020).

478. Vaswani, A. *et al.* Tensor2tensor for Neural Machine Translation. *arXiv preprint arXiv:1803.07416* (2018).

479. Tang, Y. TF.Learn: TensorFlow's High-Level Module for Distributed Machine Learning. *arXiv preprint arXiv:1612.04251* (2016).

480. Damien, A. *et al.* TFLearn Repository. Online: https://github.com/tflearn/tflearn (2019).

481. TensorFlow Addons. Online: https://github.com/tensorflow/addons (2020).

482. Sergio Guadarrama, Anoop Korattikara, Oscar Ramirez, Pablo Castro, Ethan Holly, Sam Fishman, Ke Wang, Ekaterina Gonina, Neal Wu, Efi Kokiopoulou, Luciano Sbaiz, Jamie Smith, Gábor Bartók, Jesse Berent, Chris Harris, Vincent Vanhoucke, Eugene Brevdo. TF-Agents: A Library for Reinforcement Learning in TensorFlow. Online: https://github.com/tensorflow/agents (2018).

483. Castro, P. S., Moitra, S., Gelada, C., Kumar, S. & Bellemare, M. G. Dopamine: A Research Framework for Deep Reinforcement Learning. *arXiv preprint arXiv:1812.06110* (2018).

484. McMahan, B. & Ramage, D. Federated Learning: Collaborative Machine Learning Without Centralized Training Data. *Google Res. Blog* **4** (2017).

485. TensorFlow Federated. Online: https://github.com/tensorflow/federated (2018).

486. Caldas, S. *et al.* LEAF: A Benchmark for Federated Settings. *arXiv preprint arXiv:1812.01097* (2018).

487. Dillon, J. V. *et al.* TensorFlow Distributions. *arXiv preprint arXiv:1711.10604* (2017).

488. Hessel, M., Martic, M., de Las Casas, D. & Barth-Maron, G. Open Sourcing TRFL: A Library of Reinforcement Learning Building Blocks. DeepMind Blog, Online: https://blog.paperspace.com/geometric-deep-learning-framework-comparison (2018).

489. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).

490. ANNdotNET. Online: https://github.com/bhrnjica/anndotnet (2020).

491. Create ML Documentation. Online: https://developer.apple.com/documentation/createml (2020).

492. Deep Cognition. Online: https://deepcognition.ai (2020).

493. MathWorks Deep Network Designer. Online: https://uk.mathworks.com/help/deeplearning/ref/deepnetworkdesigner-app.html (2020).

494. DIGITS. Online: https://developer.nvidia.com/digits (2020).

495. ENNUI. Online: https://math.mit.edu/ennui (2020).

496. Expresso. Online: http://val.serc.iisc.ernet.in/expresso (2020).

497. Neural Designer: Data Science and Machine Learning Platform. Online: https://www.neuraldesigner.com (2020).

498. Witten, I. H., Frank, E., Hall, M. A. & Pal, C. J. *Data Mining: Practical Machine Learning Tools and Techniques* (Morgan Kaufmann, 2016).

499. Hall, M. *et al.* The WEKA Data Mining Software: An Update. *ACM SIGKDD Explor. Newsl.* **11**, 10–18 (2009).

500. Holmes, G., Donkin, A. & Witten, I. H. WEKA: A Machine Learning Workbench. In *Proceedings of ANZIIS'94-Australian New Zealnd Intelligent Information Systems Conference*, 357–361 (IEEE, 1994).

501. Von Chamier, L. *et al.* ZeroCostDL4Mic: An Open Platform to Simplify Access and Use of Deep-Learning in Microscopy. *BioRxiv* (2020).

502. Ye, J. C. & Sung, W. K. Understanding Geometry of Encoder-Decoder CNNs. *arXiv preprint arXiv:1901.07647* (2019).

503. Ye, J. C., Han, Y. & Cha, E. Deep Convolutional Framelets: A General Deep Learning Framework for Inverse Problems. *SIAM J. on Imaging Sci.* **11**, 991–1048 (2018).

504. Sutskever, I., Vinyals, O. & Le, Q. V. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems*, 3104–3112 (2014).

505. List of Collections of Pretrained Models. Online: https://awesomeopensource.com/projects/pretrained-models (2020).

506. Model Zoo. Online: https://modelzoo.co (2020).

507. Open Neural Network Exchange. Online: https://onnx.ai (2020).

508. Bai, J., Lu, F., Zhang, K. *et al.* ONNX: Open Neural Network Exchange. Online: https://github.com/onnx/onnx (2020).

509. Shah, S. Microsoft and Facebook's Open AI Ecosystem Gains More Support. Engadget, Online: https://www.engadget.com/2017/10/11/microsoft-facebooks-ai-onxx-partners (2017).

510. Boyd, E. Microsoft and Facebook Create Open Ecosystem for AI Model Interoperability. Microsoft Azure Blog, Online: https://azure.microsoft.com/en-us/blog/microsoft-and-facebook-create-open-ecosystem-for-ai-model-interoperability (2017).

511. ONNX Model Zoo. Online: https://github.com/onnx/models (2020).

512. Gordon, J. Introducing TensorFlow Hub: A Library for Reusable Machine Learning Modules in TensorFlow. Medium, Online: https://tfhub.dev (2018).

513. TensorFlow Hub. Online: https://tfhub.dev (2020).

514. TensorFlow Model Garden. Online: https://github.com/tensorflow/models (2020).

515. Liang, H., Fu, W. & Yi, F. A Survey of Recent Advances in Transfer Learning. In *2019 IEEE 19th International Conference on Communication Technology (ICCT)*, 1516–1523 (IEEE, 2019).

516. Zhuang, F. *et al.* A Comprehensive Survey on Transfer Learning. *arXiv preprint arXiv:1911.02685* (2019).

517. Tan, C. *et al.* A Survey on Deep Transfer Learning. In *International Conference on Artificial Neural Networks*, 270–279 (Springer, 2018).

518. Marcelino, P. Transfer Learning From Pre-Trained Models. Towards Data Science, Online: https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751 (2018).

519. Weiss, K., Khoshgoftaar, T. M. & Wang, D. A Survey of Transfer Learning. *J. Big data* **3**, 9 (2016).

520. Yosinski, J., Clune, J., Bengio, Y. & Lipson, H. How Transferable are Features in Deep Neural Networks? In *Advances in Neural Information Processing Systems*, 3320–3328 (2014).

521. Da Silva, F. L., Warnell, G., Costa, A. H. R. & Stone, P. Agents Teaching Agents: A Survey on Inter-Agent Transfer Learning. *Auton. Agents Multi-Agent Syst.* **34**, 9 (2020).

522. Shermin, T. *et al.* Enhanced Transfer Learning with ImageNet Trained Classification Layer. In *Pacific-Rim Symposium on Image and Video Technology*, 142–155 (Springer, 2019).

523. Ada, S. E., Ugur, E. & Akin, H. L. Generalization in Transfer Learning. *arXiv preprint arXiv:1909.01331* (2019).

524. The Khronos NNEF Working Group. Neural Network Exchange Format. Online: https://www.khronos.org/registry/NNEF (2020).

525. The HDF Group. Hierarchical Data Format, Version 5. Online: http://www.hdfgroup.org/HDF5 (2020).

526. HDF5 for Python. Online: http://www.h5py.org (2020).

527. Somnath, S., Smith, C. R., Laanait, N., Vasudevan, R. K. & Jesse, S. USID and Pycroscopy – Open Source Frameworks for Storing and Analyzing Imaging and Spectroscopy Data. *Microsc. Microanal.* **25**, 220–221 (2019).

528. Pycroscopy Repository. Online: https://github.com/pycroscopy/pycroscopy (2020).

529. HyperSpy. Online: https://hyperspy.org (2020).

530. de la Peña, F. *et al.* Electron Microscopy (Big and Small) Data Analysis with the Open Source Software Package HyperSpy. *Microsc. Microanal.* **23**, 214–215 (2017).

531. Rezk, N. M., Purnaprajna, M., Nordström, T. & Ul-Abdin, Z. Recurrent Neural Networks: An Embedded Computing Perspective. *IEEE Access* **8**, 57967–57996 (2020).

532. Du, K.-L. & Swamy, M. Recurrent Neural Networks. In *Neural Networks and Statistical Learning*, 351–371 (Springer, 2019).

533. Yu, Y., Si, X., Hu, C. & Zhang, J. A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Comput.* **31**, 1235–1270 (2019).

534. Choe, Y. J., Shin, J. & Spencer, N. Probabilistic Interpretations of Recurrent Neural Networks. *Probabilistic Graph. Model.* (2017).

535. Choi, M., Kim, T. & Kim, J. Awesome Recurrent Neural Networks. Online: https://github.com/kjw0612/awesome-rnn (2017).

536. Lipton, Z. C., Berkowitz, J. & Elkan, C. A Critical Review of Recurrent Neural Networks for Sequence Learning. *arXiv preprint arXiv:1506.00019* (2015).

537. Hanin, B. & Rolnick, D. How to Start Training: The Effect of Initialization and Architecture. In *Advances in Neural Information Processing Systems*, 571–581 (2018).

538. Raschka, S. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. *arXiv preprint arXiv:1811.12808* (2018).

539. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1251–1258 (2017).

540. Everingham, M. *et al.* The PASCAL Visual Object Classes Challenge: A Retrospective. *Int. J. Comput. Vis.* **111**, 98–136 (2015).

541. Goyal, P. *et al.* Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *arXiv preprint arXiv:1706.02677* (2017).

542. Laanait, N. *et al.* Exascale Deep Learning for Scientific Inverse Problems. *arXiv preprint arXiv:1909.11150* (2019).

543. Castelvecchi, D. Google Unveils Search Engine for Open Data. *Nature* **561**, 161–163 (2018).

544. Noy, N. Discovering Millions of Datasets on the Web. The Keyword, Online: https://blog.google/products/search/discovering-millions-datasets-web (2020).

545. Plesa, N. Machine Learning Datasets: A List of the Biggest Machine Learning Datasets From Across the Web. Online: https://www.datasetlist.com (2020).

546. Dua, D. & Graff, C. UCI Machine Learning Repository. Online: http://archive.ics.uci.edu/ml (2020).

547. Kaggle Datasets. Online: https://www.kaggle.com/datasets (2020).

548. VisualData. Online: https://www.visualdata.io/discovery (2020).

549. Vanschoren, J., Van Rijn, J. N., Bischl, B. & Torgo, L. OpenML: Networked Science in Machine Learning. *ACM SIGKDD Explor. Newsl.* **15**, 49–60 (2014).

550. Stanford, S. The Best Public Datasets for Machine Learning and Data Science. Towards AI, Online: https://towardsai.net/datasets (2020).

551. Datasets for Data Science and Machine Learning. Elite Data Science, Online: https://elitedatascience.com/datasets (2020).

552. Iderhoff, N. Natural Language Processing Datasets. Online: https://github.com/niderhoff/nlp-datasets (2020).

553. Deep Learning Datasets. Online: http://deeplearning.net/datasets (2017).

554. Hughes, I. & Hase, T. *Measurements and Their Uncertainties: A Practical Guide to Modern Error Analysis* (Oxford University Press, 2010).

555. Working Group 1 of the Joint Committee for Guides in Metrology. JCGM 100: 2008 Evaluation of Measurement Data – Guide to the Expression of Uncertainty in Measurement. International Bureau of Weights and Measures, Online: https://www.bipm.org/utils/common/documents/jcgm/JCGM_100_2008_E.pdf (2008).

556. Vaux, D. L., Fidler, F. & Cumming, G. Replicates and Repeats - What is the Difference and is it Significant? A Brief Discussion of Statistics and Experimental Design. *EMBO Reports* **13**, 291–296 (2012).

557. Urbach, P. On the Utility of Repeating the 'Same' Experiment. *Australas. J. Philos.* **59**, 151–162 (1981).

558. Musgrave, A. Popper and 'Diminishing Returns From Repeated Tests'. *Australas. J. Philos.* **53**, 248–253 (1975).

559. Senior, A. W. *et al.* Improved Protein Structure Prediction Using Potentials From Deep Learning. *Nature* **577**, 706–710 (2020).

560. Voß, H., Heck, C. A., Schallmey, M. & Schallmey, A. Database Mining for Novel Bacterial β-Etherases, Glutathione-Dependent Lignin-Degrading Enzymes. *Appl. Environ. Microbiol.* **86** (2020).

561. Papers With Code State-of-the-Art Leaderboards. Online: https://paperswithcode.com/sota (2020).

562. Krizhevsky, A., Nair, V. & Hinton, G. The CIFAR-10 Dataset. Online: http://www.cs.toronto.edu/~kriz/cifar.html (2014).

563. Krizhevsky, A. & Hinton, G. Learning Multiple Layers of Features from Tiny Images. Tech. Rep., Citeseer (2009).

564. LeCun, Y., Cortes, C. & Burges, C. MNIST Handwritten Digit Database. AT&T Labs, Online: http://yann.lecun.com/exdb/mnist (2010).

565. Russakovsky, O. *et al.* ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **115**, 211–252 (2015).

566. Open Access Directory Data Repositories. Online: http://oad.simmons.edu/oadwiki/Data_repositories (2020).

567. Nature Scientific Data Rrecommended Data Repositories. Online: https://www.nature.com/sdata/policies/repositories (2020).

568. Zenodo. Online: https://about.zenodo.org (2020).

569. Zenodo Frequently Asked Questions. Online: https://help.zenodo.org (2020).

570. Ortega, D. R. *et al.* ETDB-Caltech: A Blockchain-Based Distributed Public Database for Electron Tomography. *PLOS ONE* **14**, e0215531 (2019).

571. EMDataResource: Unified Data Resource for 3DEM. Online: https://www.emdataresource.org/index.html (2020).

572. Lawson, C. L. *et al.* EMDataBank Unified Data Resource for 3DEM. *Nucleic Acids Res.* **44**, D396–D403 (2016).

573. Esquivel-Rodríguez, J. *et al.* Navigating 3D Electron microscopy Maps with EM-SURFER. *BMC Bioinforma.* **16**, 181 (2015).

574. Lawson, C. L. *et al.* EMDataBank.org: Unified Data Resource for CryoEM. *Nucleic Acids Res.* **39**, D456–D464 (2010).

575. Henrick, K., Newman, R., Tagari, M. & Chagoyen, M. EMDep: A Web-Based System for the Deposition and Validation of High-Resolution Electron Microscopy Macromolecular Structural Information. *J. Struct. Biol.* **144**, 228–237 (2003).

576. Tagari, M., Newman, R., Chagoyen, M., Carazo, J.-M. & Henrick, K. New Electron Microscopy Database and Deposition System. *Trends Biochem. Sci.* **27**, 589 (2002).

577. Iudin, A., Korir, P. K., Salavert-Torres, J., Kleywegt, G. J. & Patwardhan, A. EMPIAR: A Public Archive for Raw Electron Microscopy Image Data. *Nat. Methods* **13**, 387 (2016).

578. Aversa, R., Modarres, M. H., Cozzini, S., Ciancio, R. & Chiusole, A. The First Annotated Set of Scanning Electron Microscopy Images for Nanoscience. *Sci. Data* **5**, 180172 (2018).

579. Levin, B. D. *et al.* Nanomaterial Datasets to Advance Tomography in Scanning Transmission Electron Microscopy. *Sci. Data* **3**, 1–11 (2016).

580. Cerius^2 Modeling Environment: File Formats. Online: http://www.chem.cmu.edu/courses/09-560/docs/msi/modenv/D_Files.html (2020).

581. CrystalMaker: File Formats Supported. Online: http://www.crystalmaker.com/support/advice/index.html?topic=cm-file-formats (2020).

582. Bernstein, H. J. *et al.* Specification of the Crystallographic Information File format, Version 2.0. *J. Appl. Crystallogr.* **49**, 277–284 (2016).

583. Hall, S. R. & McMahon, B. The Implementation and Evolution of STAR/CIF Ontologies: Interoperability and Preservation of Structured Data. *Data Sci. J.* **15**, 3 (2016).

584. Brown, I. D. & McMahon, B. CIF: The Computer Language of Crystallography. *Acta Crystallogr. Sect. B: Struct. Sci.* **58**, 317–324 (2002).

585. Hall, S. R., Allen, F. H. & Brown, I. D. The Crystallographic Information File (CIF): A New Standard Archive File for Crystallography. *Acta Crystallogr. Sect. A: Foundations Crystallogr.* **47**, 655–685 (1991).

586. Bruno, I. *et al.* Crystallography and Databases. *Data Sci. J.* **16** (2017).

587. Crystallographic Databases and Related Resources. Online: https://www.iucr.org/resources/data/databases (2020).

588. Crystal Structure Databases. Online: https://serc.carleton.edu/research_education/crystallography/xldatabases.html (2020).

589. Quirós, M., Gražulis, S., Girdzijauskaitė, S., Merkys, A. & Vaitkus, A. Using SMILES Strings for the Description of Chemical Connectivity in the Crystallography Open Database. *J. Cheminformatics* **10**, DOI: 10.1186/s13321-018-0279-6 (2018).

590. Merkys, A. *et al.* COD::CIF::Parser: An Error-Correcting CIF Parser for the Perl Language. *J. Appl. Crystallogr.* **49**, 292–301, DOI: 10.1107/S1600576715022396 (2016).

591. Gražulis, S., Merkys, A., Vaitkus, A. & Okulič-Kazarinas, M. Computing Stoichiometric Molecular Composition From Crystal Structures. *J. Appl. Crystallogr.* **48**, 85–91, DOI: 10.1107/S1600576714025904 (2015).

592. Gražulis, S. *et al.* Crystallography Open Database (COD): An Open-Access Collection of Crystal Structures and Platform for World-Wide Collaboration. *Nucleic Acids Res.* **40**, D420–D427, DOI: 10.1093/nar/gkr900 (2012). http://nar.oxfordjournals.org/content/40/D1/D420.full.pdf+html.

593. Gražulis, S. *et al.* Crystallography Open Database – An Open-Access Collection of Crystal Structures. *J. Appl. Crystallogr.* **42**, 726–729, DOI: 10.1107/S0021889809016690 (2009).

594. Downs, R. T. & Hall-Wallace, M. The American Mineralogist Crystal Structure Database. *Am. Mineral.* **88**, 247–250 (2003).

595. Zagorac, D., Müller, H., Ruehl, S., Zagorac, J. & Rehme, S. Recent Developments in the Inorganic Crystal Structure Database: Theoretical Crystal Structure Data and Related Features. *J. Appl. Crystallogr.* **52**, 918–925 (2019).

596. Allmann, R. & Hinek, R. The Introduction of Structure Types into the Inorganic Crystal Structure Database ICSD. *Acta Crystallogr. Sect. A: Foundations Crystallogr.* **63**, 412–417 (2007).

597. Hellenbrandt, M. The Inorganic Crystal Structure Database (ICSD) - Present and Future. *Crystallogr. Rev.* **10**, 17–22 (2004).

598. Belsky, A., Hellenbrandt, M., Karen, V. L. & Luksch, P. New Developments in the Inorganic Crystal Structure Database (ICSD): Accessibility in Support of Materials Research and Design. *Acta Crystallogr. Sect. B: Struct. Sci.* **58**, 364–369 (2002).

599. Bergerhoff, G., Brown, I., Allen, F. *et al.* Crystallographic Databases. *Int. Union Crystallogr. Chester* **360**, 77–95 (1987).

600. Mighell, A. D. & Karen, V. L. NIST Crystallographic Databases for Research and Analysis. *J. Res. Natl. Inst. Standards Technol.* **101**, 273 (1996).

601. NIST Standard Reference Database 3. Online: https://www.nist.gov/srd/nist-standard-reference-database-3 (2020).

602. Kay, W. *et al.* The Kinetics Human Action Video Dataset. *arXiv preprint arXiv:1705.06950* (2017).

603. Abu-El-Haija, S. *et al.* YouTube-8M: A Large-Scale Video Classification Benchmark. *arXiv preprint arXiv:1609.08675* (2016).

604. Rehm, G. *et al.* QURATOR: Innovative Technologies for Content and Data Curation. *arXiv preprint arXiv:2004.12195* (2020).

605. van der Voort, S. R., Smits, M. & Klein, S. DeepDicomSort: An Automatic Sorting Algorithm for Brain Magnetic Resonance Imaging Data. *Neuroinformatics* (2020).

606. Pezoulas, V. C. *et al.* Medical Data Quality Assessment: On the Development of an Automated Framework for Medical Data Curation. *Comput. Biol. Medicine* **107**, 270–283 (2019).

607. Bhat, M. *et al.* ADeX: A Tool for Automatic Curation of Design Decision Knowledge for Architectural Decision recommendations. In *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*, 158–161 (IEEE, 2019).

608. Thirumuruganathan, S., Tang, N., Ouzzani, M. & Doan, A. Data curation with deep learning [vision]. *arXiv preprint arXiv:1803.01384* (2018).

609. Lee, K. *et al.* Scaling up Data Curation Using Deep Learning: An application to Literature Triage in Genomic Variation Resources. *PLoS Comput. Biol.* **14**, e1006390 (2018).

610. Freitas, A. & Curry, E. Big Data Curation. In *New Horizons for a Data-Driven Economy*, 87–118 (Springer, 2016).

611. European Microcredit Whitepaper. Online: https://www.european-microfinance.org/sites/default/files/document/file/paris_europlace_whitepaper_on_microfinance_july_2019.pdf (2019).

612. Di Cosmo, R. & Zacchiroli, S. Software Heritage: Why and How to Preserve Software Source Code. In *Proceedings of 14th International Conference on Digital Preservation (iPRES2017)* (2017).

613. Apache Allura. Online: https://allura.apache.org (2020).

614. AWS CodeCommit. Online: https://aws.amazon.com/codecommit (2020).

615. Beanstalk. Online: https://beanstalkapp.com (2020).

616. BitBucket. Online: https://bitbucket.org/product (2020).

617. GitHub. Online: https://github.com (2020).

618. GitLab. Online: https://about.gitlab.com (2020).

619. Gogs. Online: https://gogs.io (2020).

620. Google Cloud Source Repositories. Online: https://cloud.google.com/source-repositories (2020).

621. Launchpad. Online: https://launchpad.net (2020).

622. Phabricator. Online: https://www.phacility.com/phabricator (2020).

623. Savannah. Online: https://savannah.gnu.org (2020).

624. SourceForge. Online: https://sourceforge.net (2020).

625. Sheoran, J., Blincoe, K., Kalliamvakou, E., Damian, D. & Ell, J. Understanding Watchers on GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, 336–339 (2014).

626. Vale, G., Schmid, A., Santos, A. R., De Almeida, E. S. & Apel, S. On the Relation Between GitHub Communication Activity and Merge Conflicts. *Empir. Softw. Eng.* **25**, 402–433 (2020).

627. Bao, L., Xia, X., Lo, D. & Murphy, G. C. A Large Scale Study of Long-Time Contributor Prediction for GitHub Projects. *IEEE Transactions on Softw. Eng.* (2019).

628. Elazhary, O., Storey, M.-A., Ernst, N. & Zaidman, A. Do as I Do, Not as I Say: Do Contribution Guidelines Match the GitHub Contribution Process? In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 286–290 (IEEE, 2019).

629. Pinto, G., Steinmacher, I. & Gerosa, M. A. More Common than Tou Think: An In-Depth Study of Casual Contributors. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 1, 112–123 (IEEE, 2016).

630. Kobayakawa, N. & Yoshida, K. How GitHub Contributing.md Contributes to Contributors. In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, 694–696 (IEEE, 2017).

631. Lu, Y. *et al.* Studying in the 'Bazaar': An Exploratory Study of Crowdsourced Learning in GitHub. *IEEE Access* **7**, 58930–58944 (2019).

632. Qiu, H. S., Li, Y. L., Padala, S., Sarma, A. & Vasilescu, B. The Signals that Potential Contributors Look for When Choosing Open-source Projects. *Proc. ACM on Human-Computer Interact.* **3**, 1–29 (2019).

633. Alamer, G. & Alyahya, S. Open Source Software Hosting Platforms: A Collaborative Perspective's Review. *J. Softw.* **12**, 274–291 (2017).

634. Wikipedia Contributors. Comparison of source-code-hosting facilities — Wikipedia, the free encyclopedia. Online: https://en.wikipedia.org/w/index.php?title=Comparison_of_source-code-hosting_facilities&oldid=964020832 (2020). [Accessed 25-June-2020].

635. Apache Allura Feature Comparison. Online: https://forge-allura.apache.org/p/allura/wiki/Feature%20Comparison (2020).

636. Alexa Top Sites. Online: https://www.alexa.com/topsites (2020).

637. How are Alexa's Traffic Rankings Determined. Online: https://support.alexa.com/hc/en-us/articles/200449744-How-are-Alexa-s-traffic-rankings-determined- (2020).

638. Haider, J. & Sundin, O. *Invisible Search and Online Search Engines: The Ubiquity of Search in Everyday Life* (Routledge, 2019).

639. Vincent, N., Johnson, I., Sheehan, P. & Hecht, B. Measuring the Importance of User-Generated Content to Search Engines. In *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 13, 505–516 (2019).

640. Jain, A. The Role and Importance of Search Engine and Search Engine Optimization. *Int. J. Emerg. Trends & technology Comput. Sci.* **2**, 99–102 (2013).

641. Brin, S. & Page, L. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Comput. Networks* **30**, 107–117 (1998).

642. Fröbe, M., Bittner, J. P., Potthast, M. & Hagen, M. The effect of content-equivalent near-duplicates on the evaluation of search engines. In *European Conference on Information Retrieval*, 12–19 (Springer, 2020).

643. Kostagiolas, P., Strzelecki, A., Banou, C. & Lavranos, C. The Impact of Google on Discovering Scholarly Information: Managing STM publishers' Visibility in Google. *Collect. Curation* (2020).

644. Gul, S., Ali, S. & Hussain, A. Retrieval Performance of Google, Yahoo and Bing for Navigational Queries in the Field of "Life Science and Biomedicine". *Data Technol. Appl.* **54**, 133–150 (2020).

645. Shafi, S. & Ali, S. Retrieval Performance of Select Search Engines in the Field of Physical Sciences. *NISCAIR-CSIR* 117–122 (2019).

646. Steiner, M., Magin, M., Stark, B. & Geiß, S. Seek and You Shall Find? A Content Analysis on the Diversity of Five Search Engines' Results on Political Queries. *Information, Commun. & Soc.* 1–25 (2020).

647. Wu, S., Zhang, Z. & Xu, C. Evaluating the Effectiveness of Web Search Engines on Results Diversification. *Inf. Res. An Int. Electron. J.* **24**, n1 (2019).

648. Rahim, I., Mushtaq, H., Ahmad, S. *et al.* Evaluation of Search Engines Using Advanced Search: Comparative Analysis of Yahoo and Bing. *Libr. Philos. Pract.* (2019).

649. Tazehkandi, M. Z. & Nowkarizi, M. Evaluating the Effectiveness of Google, Parsijoo, Rismoon, and Yooz to Retrieve Persian Documents. *Libr. Hi Tech* (2020).

650. Gusenbauer, M. Google Scholar to Overshadow Them All? Comparing the Sizes of 12 Academic Search Engines and Bibliographic Databases. *Scientometrics* **118**, 177–214 (2019).

651. Hook, D. W., Porter, S. J. & Herzog, C. Dimensions: Building Context for Search and Evaluation. *Front. Res. Metrics Anal.* **3**, 23 (2018).

652. Bates, J., Best, P., McQuilkin, J. & Taylor, B. Will Web Search Engines Replace Bibliographic Databases in the Systematic Identification of Research? *The J. Acad. Librariansh.* **43**, 8–17 (2017).

653. Verheggen, K. *et al.* Anatomy and Evolution of Database Search Engines – A Central Component of Mass Spectrometry Based Proteomic Workflows. *Mass Spectrom. Rev.* **39**, 292–306 (2020).

654. Li, S. *et al.* Deep Job Understanding at LinkedIn. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2145–2148 (2020).

655. Agazzi, A. E. Study of the Usability of LinkedIn: A Social Media Platform Meant to Connect Employers and Employees. *arXiv preprint arXiv:2006.03931* (2020).

656. Forrester, A., Björk, B.-C. & Tenopir, C. New Web Services that Help Authors Choose Journals. *Learn. Publ.* **30**, 281–287 (2017).

657. Kang, D. M., Lee, C. C., Lee, S. & Lee, W. Patent Prior Art Search Using Deep Learning Language Model. In *Proceedings of the 24th Symposium on International Database Engineering & Applications*, 1–5 (2020).

658. Kang, M., Lee, S. & Lee, W. Prior Art Search Using Multi-modal Embedding of Patent Documents. In *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 548–550 (IEEE, 2020).

659. Shalaby, W. & Zadrozny, W. Patent Retrieval: A Literature Review. *Knowl. Inf. Syst.* **61**, 631–660 (2019).

660. Khode, A. & Jambhorkar, S. A Literature Review on Patent Information Retrieval Techniques. *Indian J. Sci. Technol.* **10**, 1–13 (2017).

661. Kong, X., Shi, Y., Yu, S., Liu, J. & Xia, F. Academic Social Networks: Modeling, Analysis, Mining and Applications. *J. Netw. Comput. Appl.* **132**, 86–103 (2019).

662. Makri, K., Papadas, K. & Schlegelmilch, B. B. Global Social Networking Sites and Global Identity: A Three-Country Study. *J. Bus. Res.* (2019).

663. Acquisti, A. & Fong, C. An Experiment in Hiring Discrimination via Online Social Networks. *Manag. Sci.* **66**, 1005–1024 (2020).

664. Mustafaraj, E., Lurie, E. & Devine, C. The Case for Voter-Centered Audits of Search Engines During Political Elections. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 559–569 (2020).

665. Kulshrestha, J. *et al.* Search Bias Quantification: Investigating Political Bias in Social Media and Web Search. *Inf. Retr. J.* **22**, 188–227 (2019).

666. Puschmann, C. Beyond the Bubble: Assessing the Diversity of Political Search Results. *Digit. Journalism* **7**, 824–843 (2019).

667. Ray, L. 2020 Google Search Survey: How Much Do Users Trust Their Search Results? MOZ, Online: https://moz.com/blog/2020-google-search-survey (2020).

668. Johnson, D. M. Lectures, Textbooks, Academic Calendar, and Administration: An Agenda for Change. In *The Uncertain Future of American Public Higher Education*, 75–89 (Springer, 2019).

669. Lin, H. Teaching and Learning Without a Textbook: Undergraduate Student Perceptions of Open Educational Resources. *Int. Rev. Res. Open Distributed Learn.* **20**, 1–18 (2019).

670. Stack Overflow. Online: https://stackoverflow.com/tour (2020).

671. Wu, Y., Wang, S., Bezemer, C.-P. & Inoue, K. How do Developers Utilize Source Code from Stack Overflow? *Empir. Softw. Eng.* **24**, 637–673 (2019).

672. Zhang, H., Wang, S., Chen, T.-H. & Hassan, A. E. Reading Answers on Stack Overflow: Not Enough! *IEEE Transactions on Softw. Eng.* (2019).

673. Zhang, T., Gao, C., Ma, L., Lyu, M. & Kim, M. An Empirical Study of Common Challenges in Developing Deep Learning Applications. In *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, 104–115 (IEEE, 2019).

674. Ragkhitwetsagul, C., Krinke, J., Paixao, M., Bianco, G. & Oliveto, R. Toxic Code Snippets on Stack Overflow. *IEEE Transactions on Softw. Eng.* (2019).

675. Zhang, T., Upadhyaya, G., Reinhardt, A., Rajan, H. & Kim, M. Are Code Examples on an Online Q&A Forum Reliable?: A Study of API Misuse on Stack Overflow. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, 886–896 (IEEE, 2018).

676. Medium. Online: https://medium.com (2020).

677. Machine Learning Subreddit. Reddit, Online: https://www.reddit.com/r/MachineLearning (2020).

678. Learn Machine Learning Subreddit. Reddit, Online: https://www.reddit.com/r/learnmachinelearning (2020).

679. Mitchell, D. R. G. & Schaffer, B. Scripting-Customised Microscopy Tools for Digital Micrograph. *Ultramicroscopy* **103**, 319–332 (2005).

680. DigitalMicrograph Scripts. Online: http://www.dmscripting.com/scripts.html (2020).

681. Internet Archive. Online: ttps://archive.org (2020).

682. Kanhabua, N. *et al.* How to Search the Internet Archive Without Indexing It. In *International Conference on Theory and Practice of Digital Libraries*, 147–160 (Springer, 2016).

683. Internet Archive Wayback Machine. Online: https://archive.org/web (2020).

684. Bowyer, S. The Wayback Machine: Notes on a Re-Enchantment. *Arch. Sci.* (2020).

685. Grotke, A. Web Archiving at the Library of Congress. *Comput. Libr.* **31**, 15–19 (2011).

686. About Distill. Online: https://distill.pub/about (2020).

687. Lewinson, E. My 10 Favorite Resources for Learning Data Science Online. Towards Data Science, Online: https://towardsdatascience.com/my-10-favorite-resources-for-learning-data-science-online-c645aa3d0afb (2020).

688. Chadha, H. S. Handpicked Resources for Learning Deep Learning in 2020. Towards Data Science, Online: https://towardsdatascience.com/handpicked-resources-for-learning-deep-learning-in-2020-e50c6768ab6e (2020).

689. Besbes, A. Here Are My Top Resources to Learn Deep Learning. Towards Data Science, Online: https://medium.com/datadriveninvestor/my-top-resources-to-learn-deep-learning-a14d1fc8e95a (2020).

690. Hutson, M. Artificial Intelligence Faces Reproducibility Crisis (2018).

691. Baker, M. Reproducibility Crisis? *Nature* **533**, 353–66 (2016).

692. Sethi, A., Sankaran, A., Panwar, N., Khare, S. & Mani, S. DLPaper2Code: Auto-Generation of Code from Deep Learning Research Papers. *arXiv preprint arXiv:1711.03543* (2017).

693. 2018 Global State of Peer Review. Publons, Online: https://publons.com/static/Publons-Global-State-Of-Peer-Review-2018.pdf (2018).

694. Tennant, J. P. The State of the Art in Peer Review. *FEMS Microbiol. Lett.* **365** (2018).

695. Walker, R. & Rocha da Silva, P. Emerging Trends in Peer Review – A Survey. *Front. Neurosci.* **9**, 169 (2015).

696. Vesper, I. Peer Reviewers Unmasked: Largest Global Survey Reveals Trends. *Nature* (2018).

697. Tan, Z.-Y., Cai, N., Zhou, J. & Zhang, S.-G. On Performance of Peer Review for Academic Journals: Analysis Based on Distributed Parallel System. *IEEE Access* **7**, 19024–19032 (2019).

698. Kim, L., Portenoy, J. H., West, J. D. & Stovel, K. W. Scientific Journals Still Matter in the Era of Academic Search Engines and Preprint Archives. *J. Assoc. for Inf. Sci. Technol.* **71** (2019).

699. Rallison, S. What are Journals For? *The Annals The Royal Coll. Surg. Engl.* **97**, 89–91 (2015).

700. Bornmann, L. & Mutz, R. Growth Rates of Modern Science: A Bibliometric Analysis Based on the Number of Publications and Cited References. *J. Assoc. for Inf. Sci. Technol.* **66**, 2215–2222 (2015).

701. Kaldas, M., Michael, S., Hanna, J. & Yousef, G. M. Journal Impact Factor: A Bumpy Ride in an Open Space. *J. Investig. Medicine* **68**, 83–87 (2020).

702. Orbay, K., Miranda, R. & Orbay, M. Building Journal Impact Factor Quartile into the Assessment of Academic Performance: A Case Study. *Particip. Educ. Res.* **7**, 1–13, DOI: https://doi.org/10.17275/per.20.26.7.2 (2020).

703. Lei, L. & Sun, Y. Should Highly Cited Items be Excluded in Impact Factor Calculation? The Effect of Review Articles on Journal Impact Factor. *Scientometrics* **122**, 1697–1706 (2020).

704. Top Most Research Tools For Selecting The Best Journal For Your Research Article. Pubrica, https://pubrica.com/academy/2019/11/14/topmost-research-tools-for-selecting-the-best-journal-for-your-research-article (2019).

705. Hoy, M. B. Rise of the Rxivs: How Preprint Servers are Changing the Publishing Process. *Med. Ref. Serv. Q.* **39**, 84–89 (2020).

706. Fry, N. K., Marshall, H. & Mellins-Cohen, T. In Praise of Preprints. *Microb. Genomics* **5** (2019).

707. Rodríguez, E. G. Preprints and Preprint Servers as Academic Communication Tools. *Revista Cuba. de Información en Ciencias de la Salud* **30**, 7 (2019).

708. About arXiv. Online: https://arxiv.org/about (2020).

709. Ginsparg, P. ArXiv at 20. *Nature* **476**, 145–147 (2011).

710. Fraser, N., Momeni, F., Mayr, P. & Peters, I. The Relationship Between bioRxiv Preprints, Citations and Altmetrics. *Quant. Sci. Stud.* **1**, 618–638 (2020).

711. Wang, Z., Glänzel, W. & Chen, Y. The Impact of Preprints in Library and Information Science: An Analysis of Citations, Usage and Social Attention Indicators. *Scientometrics* **125**, 1403–1423 (2020).

712. Furnival, A. C. & Hubbard, B. Open Access to Scholarly Communications: Advantages, Policy and Advocacy. *Acceso Abierto a la información en las Bibliotecas Académicas de América Latina y el Caribe* 101–120 (2020).

713. Fu, D. Y. & Hughey, J. J. Meta-Research: Releasing a Preprint is Associated with More Attention and Citations for the Peer-Reviewed Article. *eLife* **8**, e52646 (2019).

714. Niyazov, Y. *et al.* Open Access Meets Discoverability: Citations to Articles Posted to Academia.edu. *PLOS ONE* **11**, e0148257 (2016).

715. Robinson-Garcia, N., Costas, R. & van Leeuwen, T. N. State of Open Access Penetration in Universities Worldwide. *arXiv preprint arXiv:2003.12273* (2020).

716. Siler, K. & Frenken, K. The Pricing of Open Access Journals: Diverse Niches and Sources of Value in Academic Publishing. *Quant. Sci. Stud.* **1**, 28–59 (2020).

717. Green, T. Is Open Access Affordable? Why Current Models Do Not Work and Why We Need Internet-Era Transformation of Scholarly Communications. *Learn. Publ.* **32**, 13–25 (2019).

718. Gadd, E., Fry, J. & Creaser, C. The Influence of Journal Publisher Characteristics on Open Access Policy Trends. *Scientometrics* **115**, 1371–1393 (2018).

719. Why Should You Publish in Machine Learning: Science and Technology? IOP Science, Online: https://iopscience.iop.org/journal/2632-2153/page/about-the-journal (2020).

720. Gibney, E. Open Journals that Piggyback on arXiv Gather Momentum. *Nat. News* **530**, 117 (2016).

721. Martínez-López, J. I., Barrón-González, S. & Martínez López, A. Which Are the Tools Available for Scholars? A Review of Assisting Software for Authors During Peer Reviewing Process. *Publications* **7**, 59 (2019).

722. Microsoft Word. Online: https://www.microsoft.com/en-gb/microsoft-365/word (2020).

723. 10 Free MS Word Alternatives You Can Use Today. Investintech, https://www.investintech.com/resources/articles/tenwordalternatives (2020).

724. Pignalberi, G. & Dominici, M. Introduction to LATEX and to Some of its Tools. *ArsTEXnica* **28**, 8–46 (2019).

725. Bransen, M. & Schulpen, G. Pimp Your Thesis: A Minimal Introduction to LATEX. IC/TC, U.S.S. Proton, Online: https://ussproton.nl/files/careerweeks/20180320-pimpyourthesis.pdf (2018).

726. Lamport, L. *LATEX: A document Preparation System: User's Guide and Reference Manual* (Addison-Wesley, 1994).

727. Matthews, D. Craft Beautiful Equations in Word with LaTeX (2019).

728. Knauff, M. & Nejasmic, J. An Efficiency Comparison of Document Preparation Systems Used in Academic Research and Development. *PloS one* **9**, e115069 (2014).

729. Why I Write with LaTeX (and Why You Should Too). Medium, Online: https://medium.com/@marko_kovic/why-i-write-with-latex-and-why-you-should-too-ba6a764fadf9 (2017).

730. Allington, D. The LaTeX Fetish (Or: Don't Write in LaTeX! It's Just for Typesetting). Online: http://www.danielallington.net/2016/09/the-latex-fetish (2016).

731. Overleaf Documentation. Online: https://www.overleaf.com/learn (2020).

732. Venkateshaiah, A. *et al.* Microscopic Techniques for the Analysis of Micro and Nanostructures of Biopolymers and Their Derivatives. *Polymers* **12**, 512 (2020).

733. Alqaheem, Y. & Alomair, A. A. Microscopy and Spectroscopy Techniques for Characterization of Polymeric Membranes. *Membranes* **10**, 33 (2020).

734. Morrison, K. *Characterisation Methods in Solid State and Materials Science* (IOP Publishing, 2019).

735. Maghsoudy-Louyeh, S., Kropf, M. & Tittmann, B. Review of Progress in Atomic Force Microscopy. *The Open Neuroimaging J.* **12**, 86–104 (2018).

736. Rugar, D. & Hansma, P. Atomic Force Microscopy. *Phys. Today* **43**, 23–30 (1990).

737. Krull, A., Hirsch, P., Rother, C., Schiffrin, A. & Krull, C. Artificial-Intelligence-Driven Scanning Probe Microscopy. *Commun. Phys.* **3**, 1–8 (2020).

738. Dutta, A. Fourier Transform Infrared Spectroscopy. In *Spectroscopic Methods for Nanomaterials Characterization*, 73–93 (Elsevier, 2017).

739. Griffiths, P. R. & De Haseth, J. A. *Fourier Transform Infrared Spectrometry*, vol. 171 (John Wiley & Sons, 2007).

740. Chien, P.-H., Griffith, K. J., Liu, H., Gan, Z. & Hu, Y.-Y. Recent Advances in Solid-State Nuclear Magnetic Resonance Techniques for Materials Research. *Annu. Rev. Mater. Res.* **50**, 493–520 (2020).

741. Lambert, J. B., Mazzola, E. P. & Ridge, C. D. *Nuclear Magnetic Resonance Spectroscopy: An Introduction to Principles, Applications, and Experimental Methods* (John Wiley & Sons, 2019).

742. Mlynárik, V. Introduction to Nuclear Magnetic Resonance. *Anal. Biochem.* **529**, 4–9 (2017).

743. Rabi, I. I., Zacharias, J. R., Millman, S. & Kusch, P. A New Method of Measuring Nuclear Magnetic Moment. *Phys. Rev.* **53**, 318 (1938).

744. Smith, E. & Dent, G. *Modern Raman Spectroscopy: A Practical Approach* (John Wiley & Sons, 2019).

745. Jones, R. R., Hooper, D. C., Zhang, L., Wolverson, D. & Valev, V. K. Raman techniques: Fundamentals and frontiers. *Nanoscale Res. Lett.* **14**, 1–34 (2019).

746. Ameh, E. A Review of Basic Crystallography and X-Ray Diffraction Applications. *The Int. J. Adv. Manuf. Technol.* **105**, 3289–3302 (2019).

747. Rostron, P., Gaber, S. & Gaber, D. Raman Spectroscopy, Review. *Int. J. Eng. Tech. Res.* **6**, 2454–4698 (2016).

748. Zhang, X., Tan, Q.-H., Wu, J.-B., Shi, W. & Tan, P.-H. Review on the Raman Spectroscopy of Different Types of Layered Materials. *Nanoscale* **8**, 6435–6450 (2016).

749. Epp, J. X-Ray Diffraction (XRD) Techniques for Materials Characterization. In *Materials Characterization Using Nondestructive Evaluation (NDE) Methods*, 81–124 (Elsevier, 2016).

750. Keren, S. *et al.* Noninvasive Molecular Imaging of Small Living Subjects using Raman Spectroscopy. *Proc. Natl. Acad. Sci.* **105**, 5844–5849 (2008).

751. Khan, H. *et al.* Experimental Methods in Chemical Engineering: X-Ray Diffraction Spectroscopy – XRD. *The Can. J. Chem. Eng.* **98**, 1255–1266 (2020).

752. Scarborough, N. M. *et al.* Dynamic X-Ray Diffraction Sampling for Protein Crystal Positioning. *J. Synchrotron Radiat.* **24**, 188–195 (2017).

753. Leani, J. J., Robledo, J. I. & Sánchez, H. J. Energy Dispersive Inelastic X-Ray Scattering Spectroscopy – A Review. *Spectrochimica Acta Part B: At. Spectrosc.* **154**, 10–24 (2019).

754. Vanhoof, C., Bacon, J. R., Fittschen, U. E. & Vincze, L. 2020 Atomic Spectrometry Update – A Review of Advances in X-Ray Fluorescence Spectrometry and its Special Applications. *J. Anal. At. Spectrom.* **35**, 1704–1719 (2020).

755. Shackley, M. S. X-Ray Fluorescence Spectrometry (XRF). *The Encycl. Archaeol. Sci.* 1–5 (2018).

756. Greczynski, G. & Hultman, L. X-Ray Photoelectron Spectroscopy: Towards Reliable Binding Energy Referencing. *Prog. Mater. Sci.* **107**, 100591 (2020).

757. Baer, D. R. *et al.* Practical Guides for X-Ray Photoelectron Spectroscopy: First Steps in Planning, Conducting, and Reporting XPS Measurements. *J. Vac. Sci. & Technol. A: Vacuum, Surfaces, Films* **37**, 031401 (2019).

758. Du, M. & Jacobsen, C. Relative Merits and Limiting Factors for X-Ray and Electron Microscopy of Thick, Hydrated Organic Materials (Revised) (2020).

759. Hsu, T. Technique of Reflection Electron Microscopy. *Microsc. Res. Tech.* **20**, 318–332 (1992).

760. Yagi, K. Reflection Electron Microscopy. *J. Appl. Crystallogr.* **20**, 147–160 (1987).

761. Mohammed, A. & Abdullah, A. Scanning Electron Microscopy (SEM): A Review. In *Proceedings of the 2018 International Conference on Hydraulics and Pneumatics, Băile Govora, Romania*, 7–9 (2018).

762. Goldstein, J. I. *et al.* *Scanning Electron Microscopy and X-Ray Microanalysis* (Springer, 2017).

763. Keyse, R. *Introduction to Scanning Transmission Electron Microscopy* (Routledge, 2018).

764. Pennycook, S. J. & Nellist, P. D. *Scanning Transmission Electron Microscopy: Imaging and Analysis* (Springer Science & Business Media, 2011).

765. Sutter, P. Scanning Tunneling Microscopy in Surface Science. In *Springer Handbook of Microscopy*, 2–2 (Springer, 2019).

766. Voigtländer, B. *et al.* Invited Review Article: Multi-Tip Scanning Tunneling Microscopy: Experimental Techniques and Data Analysis. *Rev. Sci. Instruments* **89**, 101101 (2018).

767. Carter, C. B. & Williams, D. B. *Transmission Electron Microscopy: Diffraction, Imaging, and Spectrometry* (Springer, 2016).

768. Tang, C. & Yang, Z. Transmission Electron Microscopy (TEM). In *Membrane Characterization*, 145–159 (Elsevier, 2017).

769. Harris, J. R. Transmission Electron Microscopy in Molecular Structural Biology: A Historical Survey. *Arch. Biochem. Biophys.* **581**, 3–18 (2015).

770. Herzog, C., Hook, D. & Konkiel, S. Dimensions: Bringing Down Barriers Between Scientometricians and Data. *Quant. Sci. Stud.* **1**, 387–395 (2020).

771. Bode, C., Herzog, C., Hook, D. & McGrath, R. A Guide to the Dimensions Data Approach. *Digit. Sci.* (2018).

772. Adams, J. *et al.* Dimensions-A Collaborative Approach to Enhancing Research Discovery. *Digit. Sci.* (2018).

773. Gleichmann, N. SEM vs TEM. Technology Networks: Analysis & Separations, Online: https://www.technologynetworks.com/analysis/articles/sem-vs-tem-331262 (2020).

774. Owen, G. Purchasing an Electron Microscope? – Considerations and Scientific Strategies to Help in the Decision Making Process. *Microscopy* (2018).

775. Electron Microscopy Suite: Price List. The Open University, Online: http://www9.open.ac.uk/emsuite/services/price-list (2020).

776. Electron Microscopy Research Services: Prices. Newcastle University, Online: https://www.ncl.ac.uk/emrs/prices (2020).

777. Sahlgrenska Academy: Prices for Electron Microscopy. University of Gothenburg, Online: https://cf.gu.se/english/centre_for_cellular_imaging/User_Information/Prices/electron-microscopy (2020).

778. Electron Microscopy: Pricelist. Harvard Medical School, Online: https://electron-microscopy.hms.harvard.edu/pricelist (2020).

779. Cambridge Advanced Imaging Centre: Services and Charges. University of Cambridge, Online: https://caic.bio.cam.ac.uk/booking/services (2020).

780. Ichimiya, A., Cohen, P. I. & Cohen, P. I. *Reflection High-Energy Electron Diffraction* (Cambridge University Press, 2004).

781. Braun, W. *Applied RHEED: Reflection High-Energy Electron Diffraction During Crystal Growth*, vol. 154 (Springer Science & Business Media, 1999).

782. Xiang, Y., Guo, F., Lu, T. & Wang, G. Reflection High-Energy Electron Diffraction Measurements of Reciprocal Space Structure of 2D Materials. *Nanotechnology* **27**, 485703 (2016).

783. Mašek, K., Moroz, V. & Matolín, V. Reflection High-Energy Electron Loss Spectroscopy (RHEELS): A New Approach in the Investigation of Epitaxial Thin Film Growth by Reflection High-Energy Electron Diffraction (RHEED). *Vacuum* **71**, 59–64 (2003).

784. Atwater, H. A. & Ahn, C. C. Reflection Electron Energy Loss Spectroscopy During Initial Stages of Ge Growth on Si by Molecular Beam Epitaxy. *Appl. Phys. Lett.* **58**, 269–271 (1991).

785. Yu, L. *et al.* Aberration Corrected Spin Polarized Low Energy Electron Microscope. *Ultramicroscopy* **216**, 113017 (2020).

786. Bauer, E. LEEM, SPLEEM and SPELEEM. In *Springer Handbook of Microscopy*, 2–2 (Springer, 2019).

787. Li, Q. *et al.* A Study of Chiral Magnetic Stripe Domains Within an In-Plane Virtual Magnetic Field Using SPLEEM *APS* **2017**, L50–006 (2017).

788. Matsui, F. Auger Electron Spectroscopy. In *Compendium of Surface and Interface Analysis*, 39–44 (Springer, 2018).

789. MacDonald, N. & Waldrop, J. Auger Electron Spectroscopy in the Scanning Electron Microscope: Auger Electron Images. *Appl. Phys. Lett.* **19**, 315–318 (1971).

790. Scimeca, M., Bischetti, S., Lamsira, H. K., Bonfiglio, R. & Bonanno, E. Energy Dispersive X-Ray (EDX) Microanalysis: A Powerful Tool in Biomedical Research and Diagnosis. *Eur. J. Histochem.* **62** (2018).

791. Chen, Z. *et al.* Quantitative Atomic Resolution Elemental Mapping via Absolute-Scale Energy Dispersive X-Ray Spectroscopy. *Ultramicroscopy* **168**, 7–16 (2016).

792. Eggert, F., Camus, P., Schleifer, M. & Reinauer, F. Benefits from Bremsstrahlung Distribution Evaluation to get Unknown Information from Specimen in SEM and TEM. *IOP Conf. Series: Mater. Sci. Eng.* **304**, 012005 (2018).

793. Mohr, P. J., Newell, D. B. & Taylor, B. N. CODATA Recommended Values of the Fundamental Physical Constants: 2014. *J. Phys. Chem. Ref. Data* **45**, 043102 (2016).

794. Romano, A. & Marasco, A. An Introduction to Special Relativity. In *Classical Mechanics with Mathematica®*, 569–597 (Springer, 2018).

795. French, A. P. *Special Relativity* (CRC Press, 2017).

796. Rayleigh, L. XXXI. Investigations in Optics, with Special Reference to the Spectroscope. *The London, Edinburgh, Dublin Philos. Mag. J. Sci.* **8**, 261–274 (1879).

797. Ram, S., Ward, E. S. & Ober, R. J. Beyond Rayleigh's Criterion: A Resolution Measure with Application to Single-Molecule Microscopy. *Proc. Natl. Acad. Sci.* **103**, 4457–4462 (2006).

798. The Rayleigh Criterion. HyperPhysics, Online: http://hyperphysics.phy-astr.gsu.edu/hbase/phyopt/Raylei.html (2020)

799. Güémez, J., Fiolhais, M. & Fernández, L. A. The Principle of Relativity and the de Broglie Relation. *Am. J. Phys.* **84**, 443–447 (2016).

800. MacKinnon, E. De Broglie's Thesis: A Critical Retrospective. *Am. J. Phys.* **44**, 1047–1055 (1976).

801. DeBroglie Wavelength. HyperPhysics, Online: http://hyperphysics.phy-astr.gsu.edu/hbase/quantum/debrog2.html#c5 (2020).

802. Glossary of TEM Terms: Wavelength of Electron. JEOL, Online: https://www.jeol.co.jp/en/words/emterms/search_result.html?keyword=wavelength%20of%20electron (2020).

803. Mendenhall, M. H. *et al.* High-Precision Measurement of the X-Ray Cu Kα Spectrum. *J. Phys. B: At. Mol. Opt. Phys.* **50**, 115004 (2017).

804. Transmission Electron Microscopy vs Scanning Electron Microscopy. ThermoFisher Scientific, Online: https://www.thermofisher.com/uk/en/home/materials-science/learning-center/applications/sem-tem-difference.html (2020).

805. Latychevskaia, T. Spatial Coherence of Electron Beams from Field Emitters and its Effect on the Resolution of Imaged Objects. *Ultramicroscopy* **175**, 121–129 (2017).

806. Van Dyck, D. Persistent Misconceptions about Incoherence in Electron Microscopy. *Ultramicroscopy* **111**, 894–900 (2011).

807. Krumeich, F. Properties of Electrons, their Interactions with Matter and Applications in Electron Microscopy. *Lab. Inorg. Chem.* (2011).

808. Greffet, J.-J. & Nieto-Vesperinas, M. Field Theory for Generalized Bidirectional Reflectivity: Derivation of Helmholtz's Reciprocity Principle and Kirchhoff's Law. *JOSA A* **15**, 2735–2744 (1998).

809. Clarke, F. & Parry, D. Helmholtz Reciprocity: Its Validity and Application to Reflectometry. *Light. Res. & Technol.* **17**, 1–11 (1985).

810. Rose, H. & Kisielowski, C. F. On the Reciprocity of TEM and STEM. *Microsc. Microanal.* **11**, 2114 (2005).

811. Peters, J. J. P. *Structure and Ferroelectricity at the Atomic Level in Perovskite Oxides*. Ph.D. thesis, University of Warwick (2017).

812. Yakovlev, S., Downing, K., Wang, X. & Balsara, N. Advantages of HAADF vs. Conventional TEM Imaging for Study of PSS-PMB Diblock Copolymer Systems. *Microsc. Microanal.* **16**, 1698–1699 (2010).

813. Voelkl, E., Hoyle, D., Howe, J., Inada, H. & Yotsuji, T. STEM and TEM: Disparate Magnification Definitions and a Way Out. *Microsc. Microanal.* **23**, 56–57 (2017).

814. Bendersky, L. A. & Gayle, F. W. Electron Diffraction Using Transmission Electron Microscopy. *J. Res. Natl. Inst. Standards Technol.* **106**, 997 (2001).

815. Hubert, A., Römer, R. & Beanland, R. Structure Refinement from 'Digital' Large Angle Convergent Beam Electron Diffraction Patterns. *Ultramicroscopy* **198**, 1–9 (2019).

816. Beanland, R., Thomas, P. J., Woodward, D. I., Thomas, P. A. & Roemer, R. A. Digital Electron Diffraction – Seeing the Whole Picture. *Acta Crystallogr. Sect. A: Foundations Crystallogr.* **69**, 427–434 (2013).

817. Tanaka, M. Convergent-Beam Electron Diffraction. *Acta Crystallogr. Sect. A: Foundations Crystallogr.* **50**, 261–286 (1994).

818. Hovden, R. & Muller, D. A. Electron Tomography for Functional Nanomaterials. *arXiv preprint arXiv:2006.01652* (2020).

819. Koneti, S. *et al.* Fast Electron Tomography: Applications to Beam Sensitive Samples and in situ TEM or Operando Environmental TEM Studies. *Mater. Charact.* **151**, 480–495 (2019).

820. Song, H. *et al.* Electron Tomography: A Unique Tool Solving Intricate Hollow Nanostructures. *Adv. Mater.* **31**, 1801564 (2019).

821. Chen, M. *et al.* A Complete Data Processing Workflow for Cryo-ET and Subtomogram Averaging. *Nat. Methods* **16**, 1161–1168 (2019).

822. Ercius, P., Alaidi, O., Rames, M. J. & Ren, G. Electron Tomography: A Three-Dimensional Analytic Tool for Hard and Soft Materials Research. *Adv. Mater.* **27**, 5638–5663 (2015).

823. Weyland, M. & Midgley, P. A. Electron Tomography. *Mater. Today* **7**, 32–40 (2004).

824. Wang, Z. *et al.* A Consensus Framework of Distributed Multiple-Tilt Reconstruction in Electron Tomography. *J. Comput. Biol.* **27**, 212–222 (2020).

825. Doerr, A. Cryo-Electron Tomography. *Nat. Methods* **14**, 34–34 (2017).

826. Öktem, O. Mathematics of Electron Tomography. *Handb. Math. Methods Imaging* **1** (2015).

827. Tichelaar, W., Hagen, W. J., Gorelik, T. E., Xue, L. & Mahamid, J. TEM Bright Field Imaging of Thick Specimens: Nodes in Thon Ring Patterns. *Ultramicroscopy* **216**, 113023 (2020).

828. Fujii, T. *et al.* Toward Quantitative Bright Field TEM Imaging of Ultra Thin Samples. *Microsc. Microanal.* **24**, 1612–1613 (2018).

829. Vander Wal, R. L. Soot Precursor Carbonization: Visualization Using LIF and LII and Comparison Using Bright and Dark Field TEM. *Combust. Flame* **112**, 607–616 (1998).

830. Bals, S., Kabius, B., Haider, M., Radmilovic, V. & Kisielowski, C. Annular Dark Field Imaging in a TEM. *Solid State Commun.* **130**, 675–680 (2004).

831. Yücelen, E., Lazić, I. & Bosch, E. G. Phase Contrast Scanning Transmission Electron Microscopy Imaging of Light and Heavy Atoms at the Limit of Contrast and Resolution. *Sci. Reports* **8**, 1–10 (2018).

832. Krajnak, M., McGrouther, D., Maneuski, D., O'Shea, V. & McVitie, S. Pixelated Detectors and Improved Efficiency for Magnetic Imaging in STEM Differential Phase Contrast. *Ultramicroscopy* **165**, 42–50 (2016).

833. Lazić, I., Bosch, E. G. & Lazar, S. Phase Contrast STEM for Thin Samples: Integrated Differential Phase Contrast. *Ultramicroscopy* **160**, 265–280 (2016).

834. Müller-Caspary, K. *et al.* Comparison of First Moment STEM with Conventional Differential Phase contrast and the Dependence on Electron Dose. *Ultramicroscopy* **203**, 95–104 (2019).

835. Zhou, D. *et al.* Sample Tilt Effects on Atom Column Position Determination in ABF-STEM Imaging. *Ultramicroscopy* **160**, 110–117 (2016).

836. Okunishi, E. *et al.* Visualization of Light Elements at Ultrahigh Resolution by STEM Annular Bright Field Microscopy. *Microsc. Microanal.* **15**, 164–165 (2009).

837. Van den Bos, K. H. *et al.* Unscrambling Mixed Elements Using High Angle Annular Dark Field Scanning Transmission Electron Microscopy. *Phys. Rev. Lett.* **116**, 246101 (2016).

838. McMullan, G., Faruqi, A. R. & Henderson, R. Direct Electron Detectors. In *Methods in Enzymology*, vol. 579, 1–17 (Elsevier, 2016).

839. McMullan, G., Chen, S., Henderson, R. & Faruqi, A. Detective Quantum Efficiency of Electron Area Detectors in Electron Microscopy. *Ultramicroscopy* **109**, 1126–1143 (2009).

840. Torruella, P. *et al.* Clustering Analysis Strategies for Electron Energy Loss Spectroscopy (EELS). *Ultramicroscopy* **185**, 42–48 (2018).

841. Pomarico, E. *et al.* Ultrafast Electron Energy-Loss Spectroscopy in Transmission Electron Microscopy. *Mrs Bull.* **43**, 497–503 (2018).

842. Koguchi, M., Tsuneta, R., Anan, Y. & Nakamae, K. Analytical Electron Microscope Based on Scanning Transmission Electron Microscope with Wavelength Dispersive X-Ray Spectroscopy to Realize Highly Sensitive Elemental Imaging Especially for Light Elements. *Meas. Sci. Technol.* **28**, 015904 (2016).

843. Tanaka, M., Takeguchi, M. & Furuya, K. X-Ray Analysis and Mapping by Wavelength Dispersive X-Ray Spectroscopy in an Electron Microscope. *Ultramicroscopy* **108**, 1427–1431 (2008).

844. Schwartz, A. J., Kumar, M., Adams, B. L. & Field, D. P. *Electron Backscatter Diffraction in Materials Science*, vol. 2 (Springer, 2009).

845. Humphreys, F. Review Grain and Subgrain Characterisation by Electron Backscatter Diffraction. *J. Mater. Sci.* **36**, 3833–3854 (2001).

846. Winkelmann, A., Nolze, G., Vos, M., Salvat-Pujol, F. & Werner, W. Physics-Based Simulation Models for EBSD: Advances and Challenges. *Nanoscale* **12**, 15 (2016).

847. Wright, S. I., Nowell, M. M. & Field, D. P. A Review of Strain Analysis Using Electron Backscatter Diffraction. *Microsc. Microanal.* **17**, 316 (2011).

848. Wilkinson, A. J., Meaden, G. & Dingley, D. J. Mapping Strains at the Nanoscale Using Electron Back Scatter Diffraction. *Superlattices Microstruct.* **45**, 285–294 (2009).

849. Wilkinson, A. J., Meaden, G. & Dingley, D. J. High-Resolution Elastic Strain Measurement from Electron Backscatter Diffraction Patterns: New Levels of Sensitivity. *Ultramicroscopy* **106**, 307–313 (2006).

850. Wisniewski, W., Švančárek, P., Prnová, A., Parchovianskỳ, M. & Galusek, D. Y_2O_3–Al_2O_3 Microsphere Crystallization Analyzed by Electron Backscatter Diffraction (EBSD). *Sci. Reports* **10**, 1–21 (2020).

851. Basu, I., Chen, M., Loeck, M., Al-Samman, T. & Molodov, D. Determination of Grain Boundary Mobility During Recrystallization by Statistical Evaluation of Electron Backscatter Diffraction Measurements. *Mater. Charact.* **117**, 99–112 (2016).

852. Zou, Y. *et al.* Dynamic Recrystallization in the Particle/Particle Interfacial Region of Cold-Sprayed Nickel Coating: Electron Backscatter Diffraction Characterization. *Scripta Materialia* **61**, 899–902 (2009).

853. Kirkland, E. J. Image Simulation in Transmission Electron Microscopy. Cornell University, Online: http://muller.research.engineering.cornell.edu/sites/WEELS/summer06/mtutor.pdf (2006).

854. Kirkland, E. J. Computation in Electron Microscopy. *Acta Crystallogr. Sect. A: Foundations Adv.* **72**, 1–27 (2016).

855. Kirkland, E. J. *Advanced Computing in Electron Microscopy* (Springer Science & Business Media, 2010).

856. computem Repository. Online: https://sourceforge.net/projects/computem (2017).

857. Dyson, M. A. *Advances in Computational Methods for Transmission Electron Microscopy Simulation and Image Processing*. Ph.D. thesis, University of Warwick (2014).

858. Peters, J. J. P. & Dyson, M. A. clTEM. Online: https://github.com/JJPPeters/clTEM (2019).

859. cudaEM Repository. Online: https://github.com/ningustc/cudaEM (2018).

860. Barthel, J. Dr. Probe: A Software for High-Resolution STEM Image Simulation. *Ultramicroscopy* **193**, 1–11 (2018).

861. Barthel, J. Dr. Probe - STEM Image Simulation Software. Online: https://er-c.org/barthel/drprobe (2020).

862. Singh, S., Ram, F. & De Graef, M. EMsoft: Open Source Software for Electron Diffraction/Image Simulations. *Microsc. Microanal.* **23**, 212–213 (2017).

863. EMsoft Github Repository. Online: https://github.com/EMsoft-org/EMsoft (2020).

864. Stadelmann, P. JEMS. Online: https://web.archive.org/web/20151201081003/http://cimewww.epfl.ch/people/stadelmann/jemsWebSite/jems.html (2015).

865. Zuo, J. & Spence, J. *Electron Microdiffraction* (Springer Science & Business Media, 2013).

866. Lobato, I., Van Aert, S. & Verbeeck, J. Accurate and Fast Electron Microscopy Simulations Using the Open Source MULTEM Program. In *European Microscopy Congress 2016: Proceedings*, 531–532 (Wiley Online Library, 2016).

867. Lobato, I., Van Aert, S. & Verbeeck, J. Progress and New Advances in Simulating Electron Microscopy Datasets Using MULTEM. *Ultramicroscopy* **168**, 17–27 (2016).

868. Lobato, I. & Van Dyck, D. MULTEM: A New Multislice Program to Perform Accurate and Fast Electron Diffraction and Imaging Simulations Using Graphics Processing Units with CUDA. *Ultramicroscopy* **156**, 9–17 (2015).

869. O'Keefe, M. A. & Kilaas, R. Advances in High-Resolution Image Simulation. *Pfefferkorn Conf. Proceeding* (1988).

870. Electron Direct Methods. Online: http://www.numis.northwestern.edu/edm (2020).

871. Northwestern University Multislice and Imaging System. Online: http://www.numis.northwestern.edu/Software (2020).

872. Pryor, A., Ophus, C. & Miao, J. A Streaming Multi-GPU Implementation of Image Simulation Algorithms for Scanning Transmission Electron Eicroscopy. *Adv. Struct. Chem. Imaging* **3**, 15 (2017).

873. Ophus, C. A Fast Image Simulation Algorithm for Scanning Transmission Electron Microscopy. *Adv. Struct. Chem. Imaging* **3**, 13 (2017).

874. Prismatic Repository. Online: https://github.com/prism-em/prismatic (2020).

875. QSTEM. Online: https://www.physics.hu-berlin.de/en/sem/software/software_qstem (2020).

876. Gómez-Rodríguez, A., Beltrán-del Río, L. & Herrera-Becerra, R. SimulaTEM: Multislice Simulations for General Objects. *Ultramicroscopy* **110**, 95–104 (2010).

877. STEM-CELL. Online: http://tem-s3.nano.cnr.it/?page_id=2 (2020).

878. Tempas. Online: https://www.totalresolution.com/ (2020).

879. Ishizuka, K. A Practical Approach for STEM Image Simulation Based on the FFT Multislice Method. *Ultramicroscopy* **90**, 71–83 (2002).

880. Ishizuka, K. Prospects of Atomic Resolution Imaging with an Aberration-Corrected STEM. *Microscopy* **50**, 291–305 (2001).

881. Ishizuka, K. Multislice Formula for Inclined Illumination. *Acta Crystallogr. Sect. A: Cryst. Physics, Diffraction, Theor. Gen. Crystallogr.* **38**, 773–779 (1982).

882. Ishizuka, K. Contrast Transfer of Crystal Images in TEM. *Ultramicroscopy* **5**, 55–65 (1980).

883. Ishizuka, K. & Uyeda, N. A new theoretical and practical approach to the multislice method. *Acta Crystallogr. Sect. A: Cryst. Physics, Diffraction, Theor. Gen. Crystallogr.* **33**, 740–749 (1977).

884. HREM Simulation Suite. HREM Research, Online: https://www.hremresearch.com/Eng/simulation.html (2020).

885. Gianola, S., Jesus, T. S., Bargeri, S. & Castellini, G. Publish or Perish: Reporting Characteristics of Peer-Reviewed Publications, Pre-Prints and Registered Studies on the COVID-19 Pandemic. *medRxiv* (2020).

886. Nielsen, P. & Davison, R. M. Predatory Journals: A Sign of an Unhealthy Publish or Perish Game? *Inf. Syst. J.* **30**, 635–638 (2020).

887. Génova, G. & de la Vara, J. L. The Problem is not Professional Publishing, but the Publish-or-Perish Culture. *Sci. Eng. Ethics* **25**, 617–619 (2019).

888. Zuo, J.-M. & Weickenmeier, A. On the Beam Selection and Convergence in the Bloch-Wave Method. *Ultramicroscopy* **57**, 375–383 (1995).

889. Yang, Y., Yang, Q., Huang, J., Cai, C. & Lin, J. Quantitative Comparison Between Real Space and Bloch Wave Methods in Image Simulation. *Micron* **100**, 73–78 (2017).

890. Peng, Y., Nellist, P. D. & Pennycook, S. J. HAADF-STEM Imaging with Sub-Angstrom Probes: A Full Bloch Wave Analysis. *J. Electron Microsc.* **53**, 257–266 (2004).

891. Cheng, L., Ming, Y. & Ding, Z. Bohmian Trajectory-Bloch Wave Approach to Dynamical Simulation of Electron Diffraction in Crystal. *New J. Phys.* **20**, 113004 (2018).

892. Beanland, R., Evans, K., Roemer, R. A. *et al.* Felix. Online: https://github.com/RudoRoemer/Felix (2020).

893. Morimura, T. & Hasaka, M. Bloch-Wave-Based STEM Image Simulation With Layer-by-Layer Representation. *Ultramicroscopy* **109**, 1203–1209 (2009).

894. Gatan Microscopy Suite Software. Online: www.gatan.com/products/tem-analysis/gatan-microscopy-suite-software (2020).

895. FELMI/ZFE Script Database. Online: https://www.felmi-zfe.at/dm-script (2020).

896. Gatan Scripts Library. Online: https://www.gatan.com/resources/scripts-library (2020).

897. Potapov, P. temDM: Software for TEM in DigitalMicrograph. Online: http://temdm.com/web (2020).

898. Koch, C. Electron Microscopy Software. Online: https://www.physics.hu-berlin.de/en/sem/software (2016).

899. Schaffer, B. "How to script..." - Digital Micrograph Scripting Handbook. Online: http://digitalmicrograph-scripting.tavernmaker.de/HowToScript_index.htm (2015).

900. Mitchell, D. A Guide to Compiling C++ Code to Create Plugins for DigitalMicrograph (GMS 2.x). Dave Mitchell's DigitalMicrograph Scripting Website, Online: http://www.dmscripting.com/tutorial_compiling_plugins_for_GMS2.pdf (2014).

901. Miller, B. & Mick, S. Real-Time Data Processing Using Python in DigitalMicrograph. *Microsc. Microanal.* **25**, 234–235 (2019).

902. Hoffman, C. RAM Disks Explained: What They Are and Why You Probably Shouldn't Use One. How-To Geek, Online: https://www.howtogeek.com/171432/ram-disks-explained-what-they-are-and-why-you-probably-shouldnt-use-one (2019).

903. Coughlin, T., Hoyt, R. & Handy, J. Digital Storage and Memory Technology (Part 1). IEEE Technology Trend Paper, https://www.ieee.org/content/dam/ieee-org/ieee/web/org/about/corporate/ieee-industry-advisory-board/digital-storage-memory-technology.pdf (2017).

904. A dedicated Site for Quantitative Electron Microscopy. HREM Research, Online: https://www.hremresearch.com/index.html (2020).

905. Rene de Cotret, L. P. TCP Socket Plug-In for Gatan Microscopy Suite 3.x. Online: https://github.com/LaurentRDC/gms-socket-plugin (2019).

906. Schorb, M., Haberbosch, I., Hagen, W. J., Schwab, Y. & Mastronarde, D. N. Software Tools for Automated Transmission Electron Microscopy. *Nat. Methods* **16**, 471–477 (2019).

907. Peters, J. J. P. DM Stack Builder. Online: https://github.com/JJPPeters/DM-Stack-Builder (2018).

908. Wolf, D., Lubk, A. & Lichte, H. Weighted Simultaneous Iterative Reconstruction Technique for Single-Axis Tomography. *Ultramicroscopy* **136**, 15–25 (2014).

909. Wolf, D. Tomography Menu. Online: http://wwwpub.zih.tu-dresden.de/~dwolf/ (2013).

910. Schindelin, J., Rueden, C. T., Hiner, M. C. & Eliceiri, K. W. The ImageJ Ecosystem: An Open Platform for Biomedical Image Analysis. *Mol. reproduction development* **82**, 518–529 (2015).

911. EM Software. EMDataResource, Online: https://www.emdataresource.org/emsoftware.html (2020).

912. Software Tools For Molecular Microscopy. WikiBooks, Online: https://en.wikibooks.org/wiki/Software_Tools_For_Molecular_Microscopy (2020).

913. Centre for Microscopy and Microanalysis: Online Tools: Scientific Freeware. University of Queensland, Online: https://cmm.centre.uq.edu.au/online-tools (2020).

914. Ben-Nun, T. & Hoefler, T. Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis. *ACM Comput. Surv. (CSUR)* **52**, 1–43 (2019).

915. Dryden, N. *et al.* Channel and Filter Parallelism for Large-Scale CNN Training. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–20 (2019).

916. Nwankpa, C., Ijomah, W., Gachagan, A. & Marshall, S. Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. *arXiv preprint arXiv:1811.03378* (2018).

917. Hayou, S., Doucet, A. & Rousseau, J. On the Impact of the Activation Function on Deep Neural Networks Training. *arXiv preprint arXiv:1902.06853* (2019).

918. Roos, M. Deep Learning Neurons versus Biological Neurons. Towards Data Science, Online: https://towardsdatascience.com/deep-learning-versus-biological-neurons-floating-point-numbers-spikes-and-neurotransmitters-6eebfa3390e9 (2019).

919. Eldan, R. & Shamir, O. The Power of Depth for Feedforward Neural Networks. In *Conference on learning theory*, 907–940 (2016).

920. Telgarsky, M. Benefits of Depth in Neural Networks. *arXiv preprint arXiv:1602.04485* (2016).

921. Ba, J. & Caruana, R. Do Deep Nets Really Need to be Deep? In *Advances in neural information processing systems*, 2654–2662 (2014).

922. Lee, J. *et al.* Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent. In *Advances in Neural Information Processing Systems*, 8572–8583 (2019).

923. Yun, C., Sra, S. & Jadbabaie, A. Small Nonlinearities in Activation Functions Create Bad Local Minima in Neural Networks. *arXiv preprint arXiv:1802.03487* (2018).

924. Nair, V. & Hinton, G. E. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 807–814 (2010).

925. Glorot, X., Bordes, A. & Bengio, Y. Deep Sparse Rectifier Neural Networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 315–323 (2011).

926. Maas, A. L., Hannun, A. Y. & Ng, A. Y. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *Proceedings of the International Conference on Machine Learning*, vol. 30, 3 (2013).

927. Chen, Y. *et al.* Dynamic ReLU. *arXiv preprint arXiv:2003.10027* (2020).

928. Xu, B., Wang, N., Chen, T. & Li, M. Empirical Evaluation of Rectified Activations in Convolutional Network. *arXiv preprint arXiv:1505.00853* (2015).

929. Pedamonti, D. Comparison of Non-Linear Activation Functions for Deep Neural Networks on MNIST Classification Task. *arXiv preprint arXiv:1804.02763* (2018).

930. Chris. Leaky ReLU: Improving Traditional ReLU. MachineCurve, Online: https://www.machinecurve.com/index.php/2019/10/15/leaky-relu-improving-traditional-relu (2019).

931. Arnekvist, I., Carvalho, J. F., Kragic, D. & Stork, J. A. The Effect of Target Normalization and Momentum on Dying ReLU. *arXiv preprint arXiv:2005.06195* (2020).

932. Lu, L., Shin, Y., Su, Y. & Karniadakis, G. E. Dying ReLU and Initialization: Theory and Numerical Examples. *arXiv preprint arXiv:1903.06733* (2019).

933. Douglas, S. C. & Yu, J. Why RELU Units Sometimes Die: Analysis of Single-Unit Error Backpropagation in Neural Networks. In *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, 864–868 (IEEE, 2018).

934. Krizhevsky, A. & Hinton, G. Convolutional Deep Belief Networks on CIFAR-10. *Tech. Rep.* **40**, 1–9 (2010).

935. Shang, W., Sohn, K., Almeida, D. & Lee, H. Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units. In *International Conference on Machine Learning*, 2217–2225 (2016).

936. Gao, H., Cai, L. & Ji, S. Adaptive Convolutional ReLUs. In *AAAI*, 3914–3921 (2020).

937. Eidnes, L. & Nøkland, A. Shifting Mean Activation Towards Zero with Bipolar Activation Functions. *arXiv preprint arXiv:1709.04054* (2017).

938. Jiang, X., Pang, Y., Li, X., Pan, J. & Xie, Y. Deep Neural Networks with Elastic Rectified Linear Units for Object Recognition. *Neurocomputing* **275**, 1132–1139 (2018).

939. Basirat, M. & ROTH, P. L* ReLU: Piece-wise Linear Activation Functions for Deep Fine-grained Visual Categorization. In *The IEEE Winter Conference on Applications of Computer Vision*, 1218–1227 (2020).

940. Clevert, D.-A., Unterthiner, T. & Hochreiter, S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv preprint arXiv:1511.07289* (2015).

941. Klambauer, G., Unterthiner, T., Mayr, A. & Hochreiter, S. Self-Normalizing Neural Networks. In *Advances in Neural Information Processing Systems*, 971–980 (2017).

942. Hryniowski, A. & Wong, A. DeepLABNet: End-to-end Learning of Deep Radial Basis Networks with Fully Learnable Basis Functions. *arXiv preprint arXiv:1911.09257* (2019).

943. Dash, C. S. K., Behera, A. K., Dehuri, S. & Cho, S.-B. Radial Basis Function Neural Networks: A Topical State-of-the-Art Survey. *Open Comput. Sci.* **1**, 33–63 (2016).

944. Orr, M. J. L. Introduction to radial basis function networks. Online: https://www.cc.gatech.edu/~isbell/tutorials/rbf-intro.pdf (1996).

945. Jang, J.-S. & Sun, C.-T. Functional Equivalence Between Radial Basis Function Networks and Fuzzy Inference Systems. *IEEE Transactions on Neural Networks* **4**, 156–159 (1993).

946. Wuraola, A. & Patel, N. Computationally Efficient Radial Basis Function. In *International Conference on Neural Information Processing*, 103–112 (Springer, 2018).

947. Cervantes, J., Garcia-Lamont, F., Rodríguez-Mazahua, L. & Lopez, A. A Comprehensive Survey on Support Vector Machine Classification: Applications, Challenges and Trends. *Neurocomputing* **408**, 189–215 (2020).

948. Scholkopf, B. & Smola, A. J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond* (Adaptive Computation and Machine Learning Series, 2018).

949. Tavara, S. Parallel Computing of Support Vector Machines: A Survey. *ACM Comput. Surv. (CSUR)* **51**, 1–38 (2019).

950. Kundu, A. *et al.* K-TanH: Hardware Efficient Activations For Deep Learning. *arXiv preprint arXiv:1909.07729* (2019).

951. LeCun, Y. A., Bottou, L., Orr, G. B. & Müller, K.-R. Efficient Backprop. In *Neural Networks: Tricks of the Trade*, 9–48 (Springer, 2012).

952. Abdelouahab, K., Pelcat, M. & Berry, F. Why TanH is a Hardware Friendly Activation Function for CNNs. In *Proceedings of the 11th International Conference on Distributed Smart Cameras*, 199–201 (2017).

953. Gulcehre, C., Moczulski, M., Denil, M. & Bengio, Y. Noisy Activation Functions. In *International Conference on Machine Learning*, 3059–3068 (2016).

954. Dunne, R. A. & Campbell, N. A. On the Pairing of the Softmax Activation and Cross-Entropy Penalty Functions and the Derivation of the Softmax Activation Function. In *Proceedings of the 8th Australian Conference on Neural Networks, Melbourne*, vol. 181, 185 (Citeseer, 1997).

955. Dumoulin, V. & Visin, F. A Guide to Convolution Arithmetic for Deep Learning. *arXiv preprint arXiv:1603.07285* (2018).

956. Graham, B. Fractional Max-Pooling. *arXiv preprint arXiv:1412.6071* (2014).

957. Springenberg, J. T., Dosovitskiy, A., Brox, T. & Riedmiller, M. Striving for Simplicity: The All Convolutional Net *arXiv preprint arXiv:1412.6806* (2014).

958. Sabour, S., Frosst, N. & Hinton, G. E. Dynamic Routing Between Capsules. In *Advances in Neural Information Processing Systems*, 3856–3866 (2017).

959. Luo, C. *et al.* Cosine Normalization: Using Cosine Similarity Instead of Dot Product in Neural Networks. In *International Conference on Artificial Neural Networks*, 382–391 (Springer, 2018).

960. Nader, A. & Azar, D. Searching for Activation Functions Using a Self-Adaptive Evolutionary Algorithm. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, 145–146 (2020).

961. Ramachandran, P., Zoph, B. & Le, Q. Searching for Activation Functions. Google Research, Online: https://research.google/pubs/pub46503 (2018).

962. Bingham, G. & Miikkulainen, R. Discovering Parametric Activation Functions. *arXiv preprint arXiv:2006.03179* (2020).

963. Ertuğrul, Ö. F. A Novel Type of Activation Function in Artificial Neural Networks: Trained Activation Function. *Neural Networks* **99**, 148–157 (2018).

964. Lau, M. M. & Lim, K. H. Review of Adaptive Activation Function in Deep Neural Network. In *2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, 686–690 (IEEE, 2018).

965. Chung, H., Lee, S. J. & Park, J. G. Deep Neural Network Using Trainable Activation Functions. In *2016 International Joint Conference on Neural Networks (IJCNN)*, 348–352 (IEEE, 2016).

966. Agostinelli, F., Hoffman, M., Sadowski, P. & Baldi, P. Learning Activation Functions to Improve Deep Neural Networks. *arXiv preprint arXiv:1412.6830* (2014).

967. Wu, Y., Zhao, M. & Ding, X. Beyond Weights Adaptation: A New Neuron Model with Trainable Activation Function and its Supervised Learning. In *Proceedings of International Conference on Neural Networks (ICNN'97)*, vol. 2, 1152–1157 (IEEE, 1997).

968. Lee, J. *et al.* ProbAct: A Probabilistic Activation Function for Deep Neural Networks. *arXiv preprint arXiv:1905.10761* (2019).

969. Kingma, D. P. & Welling, M. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114* (2014).

970. Springenberg, J. T. & Riedmiller, M. Improving Deep Neural Networks with Probabilistic Maxout Units. *arXiv preprint arXiv:1312.6116* (2013).

971. Bawa, V. S. & Kumar, V. Linearized Sigmoidal Activation: A Novel Activation Function with Tractable Non-Linear Characteristics to Boost Representation Capability. *Expert. Syst. with Appl.* **120**, 346–356 (2019).

972. Kurita, K. An Overview of Normalization Methods in Deep Learning. Machine Learning Explained, Online: https://mlexplained.com/2018/11/30/an-overview-of-normalization-methods-in-deep-learning (2018).

973. Ren, M., Liao, R., Urtasun, R., Sinz, F. H. & Zemel, R. S. Normalizing the Normalizers: Comparing and Extending Network Normalization Schemes. *arXiv preprint arXiv:1611.04520* (2016).

974. Liao, Q., Kawaguchi, K. & Poggio, T. Streaming Normalization: Towards Simpler and More Biologically-Plausible Normalizations for Online and Recurrent Learning. *arXiv preprint arXiv:1610.06160* (2016).

975. Santurkar, S., Tsipras, D., Ilyas, A. & Madry, A. How Does Batch Normalization Help Optimization? In *Advances in Neural Information Processing Systems*, 2483–2493 (2018).

976. Ioffe, S. & Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv preprint arXiv:1502.03167* (2015).

977. Bjorck, N., Gomes, C. P., Selman, B. & Weinberger, K. Q. Understanding Batch Normalization. In *Advances in Neural Information Processing Systems*, 7694–7705 (2018).

978. Yang, G., Pennington, J., Rao, V., Sohl-Dickstein, J. & Schoenholz, S. S. A Mean Field Theory of Batch Normalization. *arXiv preprint arXiv:1902.08129* (2019).

979. Ioffe, S. & Cortes, C. Batch Normalization Layers (2019). US Patent 10,417,562.

980. Lian, X. & Liu, J. Revisit Batch Normalization: New Understanding and Refinement via Composition Optimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, 3254–3263 (2019).

981. Gao, P., Yu, L., Wu, Y. & Li, J. Low latency RNN Inference with Cellular Batching. In *Proceedings of the Thirteenth EuroSys Conference*, 1–15 (2018).

982. Fang, Z., Hong, D. & Gupta, R. K. Serving Deep Neural Networks at the Cloud Edge for Vision Applications on Mobile Platforms. In *Proceedings of the 10th ACM Multimedia Systems Conference*, 36–47 (2019).

983. Das, D. *et al.* Distributed Deep Learning Using Synchronous Stochastic Gradient Descent. *arXiv preprint arXiv:1602.06709* (2016).

984. Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M. & Tang, P. T. P. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. *arXiv preprint arXiv:1609.04836* (2016).

985. Masters, D. & Luschi, C. Revisiting Small Batch Training for Deep Neural Networks. *arXiv preprint arXiv:1804.07612* (2018).

986. You, Y., Gitman, I. & Ginsburg, B. Scaling SGD Batch Size to 32k for ImageNet Training. Tech. Rep. UCB/EECS-2017-156, EECS Department, University of California, Berkeley (2017).

987. Devarakonda, A., Naumov, M. & Garland, M. AdaBatch: Adaptive Batch Sizes for Training Deep Neural Networks. *arXiv preprint arXiv:1712.02029* (2017).

988. Hoffer, E. *et al.* Augment Your Batch: Better Training With Larger Batches. *arXiv preprint arXiv:1901.09335* (2019).

989. Hasani, M. & Khotanlou, H. An Empirical Study on Position of the Batch Normalization Layer in Convolutional Neural Networks. In *2019 5th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS)*, 1–4 (IEEE, 2019).

990. Mishkin, D., Sergievskiy, N. & Matas, J. Systematic Evaluation of Convolution Neural Network Advances on the ImageNet. *Comput. Vis. Image Underst.* **161**, 11–19 (2017).

991. Nado, Z. *et al.* Evaluating Prediction-Time Batch Normalization for Robustness Under Covariate Shift. *arXiv preprint arXiv:2006.10963* (2020).

992. Zha, D., Lai, K.-H., Zhou, K. & Hu, X. Experience Replay Optimization. *arXiv preprint arXiv:1906.08387* (2019).

993. Schaul, T., Quan, J., Antonoglou, I. & Silver, D. Prioritized Experience Replay. *arXiv preprint arXiv:1511.05952* (2015).

994. Ioffe, S. Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models. In *Advances in Neural Information Processing Systems*, 1945–1953 (2017).

995. Salimans, T. *et al.* Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems*, 2234–2242 (2016).

996. Chiley, V. *et al.* Online Normalization for Training Neural Networks. In *Advances in Neural Information Processing Systems*, 8433–8443 (2019).

997. Hoffer, E., Banner, R., Golan, I. & Soudry, D. Norm Matters: Efficient and Accurate Normalization Schemes in Deep Networks. In *Advances in Neural Information Processing Systems*, 2160–2170 (2018).

998. Ba, J. L., Kiros, J. R. & Hinton, G. E. Layer Normalization. *arXiv preprint arXiv:1607.06450* (2016).

999. Xu, J., Sun, X., Zhang, Z., Zhao, G. & Lin, J. Understanding and Improving Layer Normalization. In *Advances in Neural Information Processing Systems*, 4381–4391 (2019).

1000. Ulyanov, D., Vedaldi, A. & Lempitsky, V. Instance Normalization: The Missing Ingredient for Fast Stylization. *arXiv preprint arXiv:1607.08022* (2017).

1001. Jing, Y. *et al.* Neural Style Transfer: A Review. *IEEE Transactions on Vis. Comput. Graph.* **26**, 3365–3385 (2019).

1002. Gatys, L. A., Ecker, A. S. & Bethge, M. Image Style Transfer Using Convolutional Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2414–2423 (2016).

1003. Gatys, L. A., Ecker, A. S. & Bethge, M. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576* (2015).

1004. Zhu, J.-Y., Park, T., Isola, P. & Efros, A. A. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2223–2232 (2017).

1005. Li, Y., Wang, N., Liu, J. & Hou, X. Demystifying Neural Style Transfer. *arXiv preprint arXiv:1701.01036* (2017).

1006. Wu, Y. & He, K. Group Normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 3–19 (2018).

1007. Luo, P., Peng, Z., Ren, J. & Zhang, R. Do Normalization Layers in a Deep ConvNet Really Need to be Distinct? *arXiv preprint arXiv:1811.07727* (2018).

1008. Luo, P., Ren, J., Peng, Z., Zhang, R. & Li, J. Differentiable Learning-to-Normalize Via Switchable Normalization. *arXiv preprint arXiv:1806.10779* (2018).

1009. Nam, H. & Kim, H.-E. Batch-Instance Normalization for Adaptively Style-Invariant Neural Networks. In *Advances in Neural Information Processing Systems*, 2558–2567 (2018).

1010. Hao, K. We Analyzed 16,625 Papers to Figure Out Where AI is Headed Next. *MIT Technol. Rev.* (2019).

1011. Cooijmans, T., Ballas, N., Laurent, C., Gülçehre, Ç. & Courville, A. Recurrent Batch Normalization. *arXiv preprint arXiv:1603.09025* (2016).

1012. Liao, Q. & Poggio, T. Bridging the Gaps Between Residual Learning, Recurrent Neural Networks and Visual Cortex. *arXiv preprint arXiv:1604.03640* (2016).

1013. Laurent, C., Pereyra, G., Brakel, P., Zhang, Y. & Bengio, Y. Batch Normalized Recurrent Neural Networks. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2657–2661 (IEEE, 2016).

1014. Salimans, T. & Kingma, D. P. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. In *Advances in Neural Information Processing Systems*, 901–909 (2016).

1015. Qiao, S., Wang, H., Liu, C., Shen, W. & Yuille, A. Weight Standardization. *arXiv preprint arXiv:1903.10520* (2019).

1016. Gitman, I. & Ginsburg, B. Comparison of Batch Normalization and Weight Normalization Algorithms for the Large-Scale Image Classification. *arXiv preprint arXiv:1709.08145* (2017).

1017. Miyato, T., Kataoka, T., Koyama, M. & Yoshida, Y. Spectral Normalization for Generative Adversarial Networks. *arXiv preprint arXiv:1802.05957* (2018).

1018. Wood, G. R. & Zhang, B. P. Estimation of the Lipschitz Constant of a Function. *J. Glob. Optim.* **8**, 91–103 (1996).

1019. Hui, J. Machine Learning — Singular Value Decomposition (SVD) & Principal Component Analysis (PCA). Medium, Online: https://medium.com/@jonathan_hui/machine-learning-singular-value-decomposition-svd-principal-component-analysis-pca-1d45e885e491 (2019).

1020. Afham, M. Singular Value Decomposition and its Applications in Principal Component Analysis. Towards Data Science, Online: https://towardsdatascience.com/singular-value-decomposition-and-its-applications-in-principal-component-analysis-5b7a5f08d0bd (2020).

1021. Wall, M. E., Rechtsteiner, A. & Rocha, L. M. Singular Value Decomposition and Principal Component Analysis. In *A Practical Approach to Microarray Data Analysis*, 91–109 (Springer, 2003).

1022. Klema, V. & Laub, A. The Singular Value Decomposition: Its Computation and Some Applications. *IEEE Transactions on Autom. Control.* **25**, 164–176 (1980).

1023. Yoshida, Y. & Miyato, T. Spectral Norm Regularization for Improving the Generalizability of Deep Learning. *arXiv preprint arXiv:1705.10941* (2017).

1024. Golub, G. H. & Van der Vorst, H. A. Eigenvalue Computation in the 20th Century. *J. Comput. Appl. Math.* **123**, 35–65 (2000).

1025. Nguyen, T. Q. & Salazar, J. Transformers Without Tears: Improving the Normalization of Self-Attention. *arXiv preprint arXiv:1910.05895* (2019).

1026. Nguyen, T. Q. & Chiang, D. Improving Lexical Choice in Neural Machine Translation. *arXiv preprint arXiv:1710.01329* (2017).

1027. Stewart, M. Simple Introduction to Convolutional Neural Networks. Towards Data Science, Online: https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac (2019).

1028. Wu, J. Introduction to Convolutional Neural Networks. *Natl. Key Lab for Nov. Softw. Technol.* **5**, 23 (2017).

1029. McCann, M. T., Jin, K. H. & Unser, M. Convolutional Neural Networks for Inverse Problems in Imaging: A Review. *IEEE Signal Process. Mag.* **34**, 85–95 (2017).

1030. O'Shea, K. & Nash, R. An Introduction to Convolutional Neural Networks. *arXiv preprint arXiv:1511.08458* (2015).

1031. Hubel, D. H. & Wiesel, T. N. Receptive Fields and Functional Architecture of Monkey Striate Cortex. *The J. Physiol.* **195**, 215–243 (1968).

1032. Fukushima, K. A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biol. Cybern.* **36**, 193–202 (1980).

1033. Fukushima, K. & Miyake, S. Neocognitron: A Self-Organizing Neural Network Nodel for a Mechanism of Visual Pattern Recognition. In *Competition and Cooperation in Neural Nets*, 267–285 (Springer, 1982).

1034. Fukushima, K. Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern Recognition. *Neural Networks* **1**, 119–130 (1988).

1035. Fukushima, K. Neocognitron for Handwritten Digit Recognition. *Neurocomputing* **51**, 161–180 (2003).

1036. Atlas, L. E., Homma, T. & Marks II, R. J. An Artificial Neural Network for Spatio-Temporal Bipolar Patterns: Application to Phoneme Classification. In *Neural Information Processing Systems*, 31–40 (1988).

1037. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* **86**, 2278–2324 (1998).

1038. LeCun, Y., Haffner, P., Bottou, L. & Bengio, Y. Object Recognition with Gradient-Based Learning. In *Shape, Contour and Grouping in Computer Vision*, 319–345 (Springer, 1999).

1039. Cireşan, D. C., Meier, U., Gambardella, L. M. & Schmidhuber, J. Deep, Big, Simple Neural Nets for Handwritten Digit Recognition. *Neural Comput.* **22**, 3207–3220 (2010).

1040. Yao, G., Lei, T. & Zhong, J. A Review of Convolutional-Neural-Network-Based Action Recognition. *Pattern Recognit. Lett.* **118**, 14–22 (2019).

1041. Gupta, A. *et al.* Deep Learning in Image Cytometry: A Review. *Cytom. Part A* **95**, 366–380 (2019).

1042. Ma, S. *et al.* Image and Video Compression with Neural Networks: A Review. *IEEE Transactions on Circuits Syst. for Video Technol.* **30**, 1683–1698 (2019).

1043. Liu, D., Li, Y., Lin, J., Li, H. & Wu, F. Deep Learning-Based Video Coding: A Review and a Case Study. *ACM Comput. Surv. (CSUR)* **53**, 1–35 (2020).

1044. Bouwmans, T., Javed, S., Sultana, M. & Jung, S. K. Deep Neural Network Concepts for Background Subtraction: A Systematic Review and Comparative Evaluation. *Neural Networks* **117**, 8–66 (2019).

1045. Anwar, S. M. *et al.* Medical Image Analysis using Convolutional Neural Networks: A Review. *J. Med. Syst.* **42**, 226 (2018).

1046. Soffer, S. *et al.* Convolutional Neural Networks for Radiologic Images: A Radiologist's Guide. *Radiology* **290**, 590–606 (2019).

1047. Yamashita, R., Nishio, M., Do, R. K. G. & Togashi, K. Convolutional Neural Networks: An Overview and Application in Radiology. *Insights into Imaging* **9**, 611–629 (2018).

1048. Bernal, J. *et al.* Deep Convolutional Neural Networks for Brain Image Analysis on Magnetic Resonance Imaging: A Review. *Artif. Intell. Medicine* **95**, 64–81 (2019).

1049. Fu, Y. *et al.* Deep Learning in Medical Image Registration: A Review. *Phys. Medicine & Biol.* **65** (2020).

1050. Badar, M., Haris, M. & Fatima, A. Application of Deep Learning for Retinal Image Analysis: A review. *Comput. Sci. Rev.* **35**, 100203 (2020).

1051. Litjens, G. *et al.* A Survey on Deep Learning in Medical Image Analysis. *Med. Image Analysis* **42**, 60–88 (2017).

1052. Liu, J. *et al.* Applications of Deep Learning to MRI Images: A Survey. *Big Data Min. Anal.* **1**, 1–18 (2018).

1053. Zhao, Z.-Q., Zheng, P., Xu, S.-t. & Wu, X. Object Detection with Deep Learning: A Review. *IEEE Transactions on Neural Networks Learn. Syst.* **30**, 3212–3232 (2019).

1054. Wang, W. *et al.* Salient Object Detection in the Deep Learning Era: An In-Depth Survey. *arXiv preprint arXiv:1904.09146* (2019).

1055. Minaee, S. *et al.* Deep Learning Based Text Classification: A Comprehensive Review. *arXiv preprint arXiv:2004.03705* (2020).

1056. TensorFlow Core v2.2.0 Python Documentation for Convolutional Layer. Online: https://web.archive.org/web/20200520184050/https://www.tensorflow.org/api_docs/python/tf/nn/convolution (2020).

1057. McAndrew, A. *A Computational Introduction to Digital Image Processing* (CRC Press, 2015).

1058. Smoothing Images. OpenCV Documentation, Online: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html (2019).

1059. Vairalkar, M. K. & Nimbhorkar, S. Edge Detection of Images Using Sobel Operator. *Int. J. Emerg. Technol. Adv. Eng.* **2**, 291–293 (2012).

1060. Bogdan, V., Bonchiş, C. & Orhei, C. Custom Extended Sobel Filters. *arXiv preprint arXiv:1910.00138* (2019).

1061. Jähne, B., Scharr, H., Körkel, S. *et al.* Principles of filter design. *Handb. Comput. Vis. Appl.* **2**, 125–151 (1999).

1062. Scharr, H. *Optimal Operators in Digital Image Processing (in German).* Ph.D. thesis, University of Heidelberg (2000)

1063. Kawalec-Latała, E. Edge Detection on Images of Pseudoimpedance Section Supported by Context and Adaptive Transformation Model Images. *Studia Geotech. et Mech.* **36**, 29–36 (2014).

1064. Roberts, L. G. *Machine Perception of Three-Dimensional Solids.* Ph.D. thesis, Massachusetts Institute of Technology (1963).

1065. Prewitt, J. M. Object Enhancement and Extraction. *Pict. Process. Psychopictorics* **10**, 15–19 (1970).

1066. Jin, J., Dundar, A. & Culurciello, E. Flattened Convolutional Neural Networks for Feedforward Acceleration. *arXiv preprint arXiv:1412.5474* (2014).

1067. Chen, J., Lu, Z., Xue, J.-H. & Liao, Q. XSepConv: Extremely Separated Convolution. *arXiv preprint arXiv:2002.12046* (2020).

1068. Jaderberg, M., Vedaldi, A. & Zisserman, A. Speeding up Convolutional Neural Networks with Low Rank Expansions *arXiv preprint arXiv:1405.3866* (2014).

1069. Wu, S., Wang, G., Tang, P., Chen, F. & Shi, L. Convolution With Even-Sized Kernels and Symmetric Padding. In *Advances in Neural Information Processing Systems*, 1194–1205 (2019).

1070. Kossaifi, J., Bulat, A., Panagakis, Y., Pantic, M. & Cambridge, S. A. Efficient *N*-Dimensional Convolutions via Higher-Order Factorization. *arXiv preprint arXiv:1906.06196* (2019).

1071. Chris. Using Constant Padding, Reflection Padding and Replication Padding with Keras. MachineCurve, Online: https://www.machinecurve.com/index.php/2020/02/10/Using-constant-padding-reflection-padding-and-replication-padding-with-keras (2020).

1072. Liu, G. *et al.* Partial Convolution Based Padding. *arXiv preprint arXiv:1811.11718* (2018).

1073. Larsson, G., Maire, M. & Shakhnarovich, G. FractalNet: Ultra-Deep Neural Networks Without Residuals. *arXiv preprint arXiv:1605.07648* (2016).

1074. Szegedy, C., Ioffe, S., Vanhoucke, V. & Alemi, A. A. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *Thirty-First AAAI Conference on Artificial Intelligence* (2017).

1075. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. Rethinking the Inception Architecture for Computer Vision In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2818–2826 (2016).

1076. Szegedy, C. *et al.* Going Deeper with Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–9 (2015).

1077. Zoph, B., Vasudevan, V., Shlens, J. & Le, Q. V. Learning Transferable Architectures for Scalable Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8697–8710 (2018).

1078. Kim, J., Kwon Lee, J. & Mu Lee, K. Deeply-Recursive Convolutional Network for Image Super-Resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1637–1645 (2016).

1079. Tai, Y., Yang, J. & Liu, X. Image Super-Resolution via Deep Recursive Residual Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3147–3155 (2017).

1080. He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778 (2016).

1081. Dwarampudi, M. & Reddy, N. Effects of Padding on LSTMs and CNNs. *arXiv preprint arXiv:1903.07288* (2019).

1082. Liu, G. *et al.* Image Inpainting for Irregular Holes Using Partial Convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 85–100 (2018).

1083. Peng, Z. Multilayer Perceptron Algebra. *arXiv preprint arXiv:1701.04968* (2017).

1084. Pratama, M., Za'in, C., Ashfahani, A., Ong, Y. S. & Ding, W. Automatic Construction of Multi-Layer Perceptron Network from Streaming Examples. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 1171–1180 (2019).

1085. Neyshabur, B. Towards Learning Convolutions from Scratch. *arXiv preprint arXiv:2007.13657* (2020).

1086. Guo, L., Liu, F., Cai, C., Liu, J. & Zhang, G. 3D Deep Encoder-Decoder Network for Fluorescence Molecular Tomography. *Opt. Lett.* **44**, 1892–1895 (2019).

1087. Oseledets, I. V. Tensor-Train Decomposition. *SIAM J. on Sci. Comput.* **33**, 2295–2317 (2011).

1088. Novikov, A., Podoprikhin, D., Osokin, A. & Vetrov, D. P. Tensorizing Neural Networks. In *Advances in Neural Information Processing Systems*, 442–450 (2015).

1089. Kong, C. & Lucey, S. Take it in Your Stride: Do We Need Striding in CNNs? *arXiv preprint arXiv:1712.02502* (2017)

1090. Zaniolo, L. & Marques, O. On The Use of Variable Stride in Convolutional Neural Networks. *Multimed. Tools Appl.* **79**, 13581–13598 (2020).

1091. Shi, W. *et al.* Is the Deconvolution Layer the Same as a Convolutional Layer? *arXiv preprint arXiv:1609.07009* (2016)

1092. Aitken, A. *et al.* Checkerboard Artifact Free Sub-Pixel Convolution: A Note on Sub-Pixel Convolution, Resize Convolution and Convolution Resize. *arXiv preprint arXiv:1707.02937* (2017).

1093. Odena, A., Dumoulin, V. & Olah, C. Deconvolution and Checkerboard Artifacts. *Distill* **1** (2016).

1094. Howard, A. G. *et al.* MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861* (2017).

1095. Guo, J., Li, Y., Lin, W., Chen, Y. & Li, J. Network Decoupling: From Regular to Depthwise Separable Convolutions. *arXiv preprint arXiv:1808.05517* (2018).

1096. Depthwise Separable Convolutional Neural Networks. GeeksforGeeks, Online: https://www.geeksforgeeks.org/depth-wise-separable-convolutional-neural-networks (2020).

1097. Liu, T. Depth-wise Separable Convolutions: Performance Investigations. Online: https://tlkh.dev/depsep-convs-perf-investigations (2020).

1098. Gunther, L. The Eye. In *The Physics of Music and Color*, 325–335 (Springer, 2019).

1099. Lamb, T. D. Why Rods and Cones? *Eye* **30**, 179–185 (2016).

1100. Cohen, A. I. Rods and Cones. In *Physiology of Photoreceptor Organs*, 63–110 (Springer, 1972).

1101. He, K., Zhang, X., Ren, S. & Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis Mach. Intell.* **37**, 1904–1916 (2015).

1102. Zhang, D.-Q. Image Recognition Using Scale Recurrent Neural Networks. *arXiv preprint arXiv:1803.09218* (2018).

1103. Tanaka, N. Introduction to Fourier Transforms for TEM and STEM. In *Electron Nano-Imaging*, 219–226 (Springer, 2017).

1104. Fourier Transform Conventions. Mathematica Documentation, Online: https://reference.wolfram.com/language/tutorial/Calculus.html#26017 (2020).

1105. Frigo, M. & Johnson, S. G. The Design and Implementation of FFTW3. *Proc. IEEE* **93**, 216–231 (2005).

1106. Stokfiszewski, K., Wieloch, K. & Yatsymirskyy, M. The Fast Fourier Transform Partitioning Scheme for GPU's Computation Effectiveness Improvement. In *Conference on Computer Science and Information Technologies*, 511–522 (Springer, 2017).

1107. Chen, Y., Cui, X. & Mei, H. Large-Scale FFT on GPU Clusters. In *Proceedings of the 24th ACM International Conference on Supercomputing*, 315–324 (2010).

1108. Gu, L., Li, X. & Siegel, J. An Empirically Tuned 2D and 3D FFT Library on CUDA GPU. In *Proceedings of the 24th ACM International Conference on Supercomputing*, 305–314 (2010).

1109. Puchała, D., Stokfiszewski, K., Yatsymirskyy, M. & Szczepaniak, B. Effectiveness of Fast Fourier Transform Implementations on GPU and CPU. In *2015 16th International Conference on Computational Problems of Electrical Engineering (CPEE)*, 162–164 (IEEE, 2015).

1110. Ogata, Y., Endo, T., Maruyama, N. & Matsuoka, S. An Efficient, Model-Based CPU-GPU Heterogeneous FFT Library. In *2008 IEEE International Symposium on Parallel and Distributed Processing*, 1–10 (IEEE, 2008).

1111. Cooley, J. W. & Tukey, J. W. An Algorithm for the Machine Calculation of Complex Fourier Series. *Math. computation* **19**, 297–301 (1965).

1112. Duhamel, P. & Vetterli, M. Fast Fourier Transforms: A Tutorial Review and A State of the Art. *Signal Process. (Elsevier)* **19**, 259–299 (1990).

1113. clFFT Repository. Online: https://github.com/clMathLibraries/clFFT (2017).

1114. Highlander, T. & Rodriguez, A. Very Efficient Training of Convolutional Neural Networks Using Fast Fourier Transform and Overlap-and-Add. *arXiv preprint arXiv:1601.06815* (2016).

1115. Weisstein, E. W. Convolution Theorem. Wolfram Mathworld – A Wolfram Web Resource, Online: https://mathworld.wolfram.com/ConvolutionTheorem.html (2020).

1116. Pratt, H., Williams, B., Coenen, F. & Zheng, Y. FCNN: Fourier Convolutional Neural Networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 786–798 (Springer, 2017).

1117. Simonyan, K. & Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556* (2014).

1118. Thomson, A. M. Neocortical Layer 6, A Review. *Front. Neuroanat.* **4**, 13 (2010).

1119. Fitzpatrick, D. The Functional Organization of Local Circuits in Visual Cortex: Insights From the Study of Tree Shrew Striate Cortex. *Cereb. Cortex* **6**, 329–341 (1996).

1120. Zaeemzadeh, A., Rahnavard, N. & Shah, M. Norm-Preservation: Why Residual Networks can Become Extremely Deep? *IEEE Transactions on Pattern Analysis Mach. Intell.* (2020).

1121. Kawaguchi, K. & Bengio, Y. Depth with Nonlinearity Creates No Bad Local Minima in ResNets. *Neural Networks* **118**, 167–174 (2019).

1122. Li, H., Xu, Z., Taylor, G., Studer, C. & Goldstein, T. Visualizing the Loss Landscape of Neural Nets. In *Advances in Neural Information Processing Systems*, 6389–6399 (2018).

1123. Veit, A., Wilber, M. J. & Belongie, S. Residual Networks Behave Like Ensembles of Relatively Shallow Networks. In *Advances in Neural Information Processing Systems*, 550–558 (2016).

1124. Greff, K., Srivastava, R. K. & Schmidhuber, J. Highway and Residual Networks Learn Unrolled Iterative Estimation. *arXiv preprint arXiv:1612.07771* (2016).

1125. Martinez, J., Hossain, R., Romero, J. & Little, J. J. A Simple Yet Effective Baseline for 3D Human Pose Estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, 2640–2649 (2017).

1126. Yue, B., Fu, J. & Liang, J. Residual Recurrent Neural Networks for Learning Sequential Representations. *Information* **9**, 56 (2018).

1127. Kim, J., El-Khamy, M. & Lee, J. Residual LSTM: Design of a Deep Recurrent Architecture for Distant Speech Recognition. In *Proceedings of Interspeech 2017*, 1591–1595 (2017).

1128. Wu, Y. *et al.* Google's Neural Machine Translation System: Bridging the Gap Between Human and Machine Translation. *arXiv preprint arXiv:1609.08144* (2016).

1129. Srivastava, R. K., Greff, K. & Schmidhuber, J. Training Very Deep Networks. In *Advances in Neural Information Processing Systems*, 2377–2385 (2015).

1130. Srivastava, R. K., Greff, K. & Schmidhuber, J. Highway Networks. *arXiv preprint arXiv:1505.00387* (2015).

1131. Huang, G., Liu, Z., Pleiss, G., Van Der Maaten, L. & Weinberger, K. Convolutional Networks with Dense Connectivity. *IEEE Transactions on Pattern Analysis Mach. Intell.* (2019).

1132. Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, K. Q. Densely Connected Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4700–4708 (2017).

1133. Tong, T., Li, G., Liu, X. & Gao, Q. Image Super-Resolution Using Dense Skip Connections. In *Proceedings of the IEEE International Conference on Computer Vision*, 4799–4807 (2017).

1134. Jiang, F. *et al.* An End-to-End Compression Framework Based on Convolutional Neural Networks. *IEEE Transactions on Circuits Syst. for Video Technol.* **28**, 3007–3018 (2017).

1135. Yang, G. & Schoenholz, S. Mean Field Residual Networks: On the Edge of Chaos. In *Advances in Neural Information Processing Systems*, 7103–7114 (2017).

1136. Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S. & Pennington, J. Dynamical Isometry and a Mean Field Theory of CNNs: How to Train 10,000-Layer Vanilla Convolutional Neural Networks. In *International Conference on Machine Learning*, 5393–5402 (2018).

1137. Wu, Q. & Wang, F. Concatenate Convolutional Neural Networks for Non-Intrusive Load Monitoring Across Complex Background. *Energies* **12**, 1572 (2019).

1138. Terwilliger, A. M., Perdue, G. N., Isele, D., Patton, R. M. & Young, S. R. Vertex Reconstruction of Neutrino Interactions Using Deep Learning. In *2017 International Joint Conference on Neural Networks (IJCNN)*, 2275–2281 (IEEE, 2017).

1139. Gers, F. A., Schraudolph, N. N. & Schmidhuber, J. Learning Precise Timing with LSTM Recurrent Networks. *J. Mach. Learn. Res.* **3**, 115–143 (2002).

1140. Gers, F. A. & Schmidhuber, E. LSTM Recurrent Networks Learn Simple Context-Free and Context-Sensitive Languages. *IEEE Transactions on Neural Networks* **12**, 1333–1340 (2001).

1141. Lin, M., Chen, Q. & Yan, S. Network-in-Network. *arXiv preprint arXiv:1312.4400* (2013).

1142. Vaswani, A. *et al.* Attention is All You Need. In *Advances in Neural Information Processing Systems*, 5998–6008 (2017).

1143. Alammar, J. The Illustrated Transformer. GitHub Blog, Online: http://jalammar.github.io/illustrated-transformer (2018).

1144. Mnih, V., Heess, N., Graves, A. & Kavukcuoglu, K. Recurrent Models of Visual Attention. In *Advances in Neural Information Processing Systems*, 2204–2212 (2014).

1145. Ba, J., Mnih, V. & Kavukcuoglu, K. Multiple Object Recognition with Visual Attention. *arXiv preprint arXiv:1412.7755* (2014).

1146. Lillicrap, T. P. *et al.* Continuous Control with Deep Reinforcement Learning. *arXiv preprint arXiv:1509.02971* (2015).

1147. Heess, N., Hunt, J. J., Lillicrap, T. P. & Silver, D. Memory-Based Control with Recurrent Neural Networks. *arXiv preprint arXiv:1512.04455* (2015).

1148. Konda, V. R. & Tsitsiklis, J. N. Actor-Critic Algorithms. In *Advances in Neural Information Processing Systems*, 1008–1014 (2000).

1149. Grabocka, J., Scholz, R. & Schmidt-Thieme, L. Learning Surrogate Losses. *arXiv preprint arXiv:1905.10108* (2019).

1150. Neftci, E. O., Mostafa, H. & Zenke, F. Surrogate Gradient Learning in Spiking Neural Networks. *IEEE Signal Process. Mag.* **36**, 61–63 (2019).

1151. Liang, K. J., Li, C., Wang, G. & Carin, L. Generative Adversarial Network Training is a Continual Learning Problem. *arXiv preprint arXiv:1811.11083* (2018).

1152. Jaderberg, M. *et al.* Decoupled Neural Interfaces Using Synthetic Gradients. In *International Conference on Machine Learning*, 1627–1635 (2017).

1153. Wang, Z., Bovik, A. C., Sheikh, H. R. & Simoncelli, E. P. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE transactions on image processing* **13**, 600–612 (2004).

1154. Pan, Z. *et al.* Loss Functions of Generative Adversarial Networks (GANs): Opportunities and Challenges. *IEEE Transactions on Emerg. Top. Comput. Intell.* **4**, 500–522 (2020).

1155. Dong, H.-W. & Yang, Y.-H. Towards a Deeper Understanding of Adversarial Losses. *arXiv preprint arXiv:1901.08753* (2019).

1156. Mescheder, L., Geiger, A. & Nowozin, S. Which Training Methods for GANs do Actually Converge? *arXiv preprint arXiv:1801.04406* (2018).

1157. Kurach, K., Lučić, M., Zhai, X., Michalski, M. & Gelly, S. A Large-Scale Study on Regularization and Normalization in GANs. In *International Conference on Machine Learning*, 3581–3590 (2019).

1158. Roth, K., Lucchi, A., Nowozin, S. & Hofmann, T. Stabilizing Training of Generative Adversarial Networks Through Regularization. In *Advances in Neural Information Processing Systems*, 2018–2028 (2017).

1159. Goodfellow, I. *et al.* Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, 2672–2680 (2014).

1160. Mao, X. *et al.* On the Effectiveness of Least Squares Generative Adversarial Networks. *IEEE Transactions on Pattern Analysis Mach. Intell.* **41**, 2947–2960 (2018).

1161. Mao, X. *et al.* Least Squares Generative Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2794–2802 (2017).

1162. Wiatrak, M. & Albrecht, S. V. Stabilizing Generative Adversarial Network Training: A Survey. *arXiv preprint arXiv:1910.00927* (2019).

1163. Bang, D. & Shim, H. MGGAN: Solving Mode Collapse Using Manifold Guided Training. *arXiv preprint arXiv:1804.04391* (2018).

1164. Arjovsky, M., Chintala, S. & Bottou, L. Wasserstein Generative Adversarial Networks. In *International Conference on Machine Learning*, 214–223 (2017).

1165. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. & Courville, A. C. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, 5767–5777 (2017).

1166. Hazan, T., Papandreou, G. & Tarlow, D. *Adversarial Perturbations of Deep Neural Networks*, 311–342 (MIT Press, 2017).

1167. Chen, Z., Badrinarayanan, V., Lee, C.-Y. & Rabinovich, A. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. *arXiv preprint arXiv:1711.02257* (2017).

1168. Lee, S. & Son, Y. Multitask Learning with Single Gradient Step Update for Task Balancing. *arXiv preprint arXiv:2005.09910* (2020).

1169. Zhang, H., Goodfellow, I., Metaxas, D. & Odena, A. Self-Attention Generative Adversarial Networks. In *International Conference on Machine Learning*, 7354–7363 (2019).

1170. Brock, A., Donahue, J. & Simonyan, K. Large Scale GAN Training for High Fidelity Natural Image Synthesis. *arXiv preprint arXiv:1809.11096* (2018).

1171. Hindupur, A. The GAN Zoo. Online: https://github.com/hindupuravinash/the-gan-zoo (2018).

1172. Wang, T.-C. *et al.* High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8798–8807 (2018).

1173. Bashkirova, D., Usman, B. & Saenko, K. Unsupervised Video-to-Video Translation. *arXiv preprint arXiv:1806.03698* (2018).

1174. Liu, M.-Y., Breuel, T. & Kautz, J. Unsupervised Image-to-Image Translation Networks. In *Advances in Neural Information Processing Systems*, 700–708 (2017).

1175. Amodio, M. & Krishnaswamy, S. TraVeLGAN: Image-to-Image Translation by Transformation Vector Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8983–8992 (2019).

1176. Tzeng, E., Hoffman, J., Saenko, K. & Darrell, T. Adversarial Discriminative Domain Adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7167–7176 (2017).

1177. Ganin, Y. & Lempitsky, V. Unsupervised Domain Adaptation by Backpropagation. In *International Conference on Machine Learning*, 1180–1189 (2015).

1178. Tzeng, E., Hoffman, J., Darrell, T. & Saenko, K. Simultaneous Deep Transfer Across Domains and Tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, 4068–4076 (2015).

1179. Werbos, P. J. Backpropagation Through Time: What It Does and How To Do It. *Proc. IEEE* **78**, 1550–1560 (1990).

1180. Saldi, N., Yüksel, S. & Linder, T. Asymptotic Optimality of Finite Model Approximations for Partially Observed Markov Decision Processes With Discounted Cost. *IEEE Transactions on Autom. Control.* **65**, 130–142 (2019).

1181. Jaakkola, T., Singh, S. P. & Jordan, M. I. Reinforcement Learning Algorithm for Partially Observable Markov Decision Problems. In *Advances in Neural Information Processing Systems*, 345–352 (1995).

1182. Xu, K. *et al.* Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *International Conference on Machine Learning*, 2048–2057 (2015).

1183. Vinyals, O., Toshev, A., Bengio, S. & Erhan, D. Show and Tell: A Neural Image Caption Generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3156–3164 (2015).

1184. Basmatkar, P., Holani, H. & Kaushal, S. Survey on Neural Machine Translation for Multilingual Translation System. In *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, 443–448 (IEEE, 2019).

1185. Wu, S. *et al.* Deep Learning in Clinical Natural Language Processing: A Methodical Review. *J. Am. Med. Informatics Assoc.* **27**, 457–470 (2020).

1186. Otter, D. W., Medina, J. R. & Kalita, J. K. A Survey of the Usages of Deep Learning for Natural Language Processing. *IEEE Transactions on Neural Networks Learn. Syst.* (2020).

1187. Iyer, S. R., An, U. & Subramanian, L. Forecasting Sparse Traffic Congestion Patterns Using Message-Passing RNNs. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3772–3776 (IEEE, 2020).

1188. Mandal, P. K. & Mahto, R. Deep CNN-LSTM with Word Embeddings for News Headline Sarcasm Detection. In *16th International Conference on Information Technology-New Generations (ITNG 2019)*, 495–498 (Springer, 2019).

1189. Rhanoui, M., Mikram, M., Yousfi, S. & Barzali, S. A CNN-BiLSTM Model for Document-Level Sentiment Analysis. *Mach. Learn. Knowl. Extr.* **1**, 832–847 (2019).

1190. Zhang, X., Chen, F. & Huang, R. A Combination of RNN and CNN for Attention-Based Relation Classification. *Procedia Comput. Sci.* **131**, 911–917 (2018).

1191. Qu, Y., Liu, J., Kang, L., Shi, Q. & Ye, D. Question Answering Over Freebase via Attentive RNN with Similarity Matrix Based CNN. *arXiv preprint arXiv:1804.03317* **38** (2018).

1192. Sieg, A. From Pre-trained Word Embeddings To Pre-trained Language Models – Focus on BERT. Towards Data Science, Online: https://towardsdatascience.com/from-pre-trained-word-embeddings-to-pre-trained-language-models-focus-on-bert-343815627598 (2019).

1193. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805* (2018).

1194. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. Distributed Representations of Words and Phrases and Their Compositionality. In *Advances in Neural Information Processing Systems*, 3111–3119 (2013).

1195. Mnih, A. & Kavukcuoglu, K. Learning Word Embeddings Efficiently with Noise-Contrastive Estimation. In *Advances in Neural Information Processing Systems*, 2265–2273 (2013).

1196. Grave, É., Bojanowski, P., Gupta, P., Joulin, A. & Mikolov, T. Learning Word Vectors for 157 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)* (2018).

1197. Le, Q. & Mikolov, T. Distributed Representations of Sentences and Documents. In *International Conference on Machine Learning*, 1188–1196 (2014).

1198. Lau, J. H. & Baldwin, T. An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. *arXiv preprint arXiv:1607.05368* (2016).

1199. Pennington, J., Socher, R. & Manning, C. D. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543 (2014).

1200. Mikolov, T., Chen, K., Corrado, G. & Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781* (2013).

1201. Sherstinsky, A. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network *Phys. D: Nonlinear Phenom.* **404**, 132306 (2020).

1202. Olah, C. Understanding LSTM Networks. Online: https://colah.github.io/posts/2015-08-Understanding-LSTMs (2015)

1203. Gers, F. A., Schmidhuber, J. & Cummins, F. Learning to Forget: Continual Prediction with LSTM. *Neural Comput.* **12**, 2451–2471 (2000).

1204. Hochreiter, S. & Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **9**, 1735–1780 (1997).

1205. Cho, K. *et al.* Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv preprint arXiv:1406.1078* (2014).

1206. Dey, R. & Salemt, F. M. Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks. In *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 1597–1600 (IEEE, 2017).

1207. Heck, J. C. & Salem, F. M. Simplified Minimal Gated Unit Variations for Recurrent Neural Networks. In *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 1593–1596 (IEEE, 2017).

1208. Pascanu, R., Mikolov, T. & Bengio, Y. On the Difficulty of Training Recurrent Neural Networks. In *International Conference on Machine Learning*, 1310–1318 (2013).

1209. Hanin, B. Which Neural Net Architectures Give Rise to Exploding and Vanishing Gradients? In *Advances in Neural Information Processing Systems*, 582–591 (2018).

1210. Britz, D., Goldie, A., Luong, M.-T. & Le, Q. Massive Exploration of Neural Machine Translation Architectures. *arXiv preprint arXiv:1703.03906* (2017).

1211. Jozefowicz, R., Zaremba, W. & Sutskever, I. An Empirical Exploration of Recurrent Network Architectures. In *International Conference on Machine Learning*, 2342–2350 (2015).

1212. Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In *NIPS 2014 Workshop on Deep Learning* (2014).

1213. Gruber, N. & Jockisch, A. Are GRU Cells More Specific and LSTM Cells More Sensitive in Motive Classification of Text? *Front. Artif. Intell.* **3**, 40 (2020).

1214. Weiss, G., Goldberg, Y. & Yahav, E. On the Practical Computational Power of Finite Precision RNNs for Language Recognition. *arXiv preprint arXiv:1805.04908* (2018).

1215. Bayer, J., Wierstra, D., Togelius, J. & Schmidhuber, J. Evolving Memory Cell Structures for Sequence Learning. In *International Conference on Artificial Neural Networks*, 755–764 (Springer, 2009).

1216. Zhou, G.-B., Wu, J., Zhang, C.-L. & Zhou, Z.-H. Minimal Gated Unit for Recurrent Neural Networks. *Int. J. Autom. Comput.* **13**, 226–234 (2016).

1217. Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R. & Schmidhuber, J. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks Learn. Syst.* **28**, 2222–2232 (2016).

1218. Mozer, M. C., Kazakov, D. & Lindsey, R. V. Discrete Event, Continuous Time RNNs. *arXiv preprint arXiv:1710.04110* (2017).

1219. Funahashi, K.-i. & Nakamura, Y. Approximation of Dynamical Systems by Continuous Time Recurrent Neural Networks. *Neural Networks* **6**, 801–806 (1993).

1220. Quinn, M. Evolving Communication Without Dedicated Communication Channels. In *European Conference on Artificial Life*, 357–366 (Springer, 2001).

1221. Beer, R. D. The Dynamics of Adaptive Behavior: A Research Program. *Robotics Auton. Syst.* **20**, 257–289 (1997).

1222. Harvey, I., Husbands, P. & Cliff, D. Seeing the Light: Artificial Evolution, Real Vision. *From Animals to Animat.* **3**, 392–401 (1994).

1223. Elman, J. L. Finding Structure in Time. *Cogn. Sci.* **14**, 179–211 (1990).

1224. Jordan, M. I. Serial Order: A Parallel Distributed Processing Approach. In *Advances in Psychology*, vol. 121, 471–495 (Elsevier, 1997).

1225. Li, S., Li, W., Cook, C., Zhu, C. & Gao, Y. Independently Recurrent Neural Network (IndRNN): Building a Longer and Deeper RNN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5457–5466 (2018).

1226. Sathasivam, S. & Abdullah, W. A. T. W. Logic Learning in Hopfield Networks. *arXiv preprint arXiv:0804.4075* (2008).

1227. Tutschku, K. Recurrent Multilayer Perceptrons for Identification and Control: The Road to Applications. *Inst. Comput. Sci. Res. Report, Univ. Würzburg Am Hubland* (1995).

1228. Jia, Y., Wu, Z., Xu, Y., Ke, D. & Su, K. Long Short-Term Memory Projection Recurrent Neural Network Architectures for Piano's Continuous Note Recognition. *J. Robotics* **2017** (2017).

1229. Pascanu, R., Gulcehre, C., Cho, K. & Bengio, Y. How to Construct Deep Recurrent Neural Networks. In *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)* (2014).

1230. Schuster, M. & Paliwal, K. K. Bidirectional Recurrent Neural Networks. *IEEE transactions on Signal Process.* **45**, 2673–2681 (1997).

1231. Bahdanau, D., Cho, K. & Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015* (2015).

1232. Graves, A. & Schmidhuber, J. Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures. *Neural Networks* **18**, 602–610 (2005).

1233. Thireou, T. & Reczko, M. Bidirectional Long Short-Term Memory Networks for Predicting the Subcellular Localization of Eukaryotic Proteins. *IEEE/ACM Transactions on Comput. Biol. Bioinforma.* **4**, 441–446 (2007).

1234. Cho, K., Van Merriënboer, B., Bahdanau, D. & Bengio, Y. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *arXiv preprint arXiv:1409.1259* (2014).

1235. Zhang, T., Huang, M. & Zhao, L. Learning Structured Representation for Text Classification via Reinforcement Learning. In *Thirty-Second AAAI Conference on Artificial Intelligence* (2018).

1236. Chung, J., Ahn, S. & Bengio, Y. Hierarchical Multiscale Recurrent Neural Networks. *arXiv preprint arXiv:1609.01704* (2016).

1237. Sordoni, A. *et al.* A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 553–562 (2015).

1238. Paine, R. W. & Tani, J. How Hierarchical Control Self-Organizes in Artificial Adaptive Systems. *Adapt. Behav.* **13**, 211–225 (2005).

1239. Schmidhuber, J. Learning Complex, Extended Sequences Using the Principle of History Compression. *Neural Comput.* **4**, 234–242 (1992).

1240. Yamashita, Y. & Tani, J. Emergence of Functional Hierarchy in a Multiple Timescale Neural Network Model: A Humanoid Robot Experiment. *PLoS Comput. Biol.* **4**, e1000220 (2008).

1241. Shibata Alnajjar, F., Yamashita, Y. & Tani, J. The Hierarchical and Functional Connectivity of Higher-Order Cognitive Mechanisms: Neurorobotic Model to Investigate the Stability and Flexibility of Working Memory. *Front. Neurorobotics* **7**, 2 (2013).

1242. Chaudhari, S., Polatkan, G., Ramanath, R. & Mithal, V. An Attentive Survey of Attention Models. *arXiv preprint arXiv:1904.02874* (2019).

1243. Luong, M.-T., Pham, H. & Manning, C. D. Effective Approaches to Attention-Based Neural Machine Translation. *arXiv preprint arXiv:1508.04025* (2015).

1244. Bahdanau, D., Cho, K. & Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473* (2014).

1245. Graves, A. *et al.* Hybrid Computing Using a Neural Network with Dynamic External Memory. *Nature* **538**, 471–476 (2016).

1246. Graves, A., Wayne, G. & Danihelka, I. Neural Turing Machines. *arXiv preprint arXiv:1410.5401* (2014).

1247. Tschannen, M., Bachem, O. & Lucic, M. Recent Advances in Autoencoder-Based Representation Learning. *arXiv preprint arXiv:1812.05069* (2018).

1248. Hinton, G. E. & Salakhutdinov, R. R. Reducing the Dimensionality of Data with Neural Networks. *science* **313**, 504–507 (2006).

1249. Kramer, M. A. Nonlinear Principal Component Analysis Using Autoassociative Neural Networks. *AIChE J.* **37**, 233–243 (1991).

1250. Zhou, Y., Arpit, D., Nwogu, I. & Govindaraju, V. Is Joint Training Better for Deep Auto-Encoders? *arXiv preprint arXiv:1405.1380* (2014).

1251. Jolliffe, I. T. & Cadima, J. Principal Component Analysis: A Review and Recent Developments. *Philos. Transactions Royal Soc. A: Math. Phys. Eng. Sci.* **374**, 20150202 (2016).

1252. Theis, L., Shi, W., Cunningham, A. & Huszár, F. Lossy Image Compression with Compressive Autoencoders. *arXiv preprint arXiv:1703.00395* (2017).

1253. Vincent, P. *et al.* Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *J. Mach. Learn. Res.* **11**, 3371–3408 (2010).

1254. Vincent, P., Larochelle, H., Bengio, Y. & Manzagol, P.-A. Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, 1096–1103 (2008).

1255. Gondara, L. Medical Image Denoising Using Convolutional Denoising Autoencoders. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, 241–246 (IEEE, 2016).

1256. Cho, K. Simple Sparsification Improves Sparse Denoising Autoencoders in Denoising Highly Corrupted Images. In *International Conference on Machine Learning*, 432–440 (2013).

1257. Cho, K. Boltzmann Machines and Denoising Autoencoders for Image Denoising. *arXiv preprint arXiv:1301.3468* (2013).

1258. Rifai, S., Vincent, P., Muller, X., Glorot, X. & Bengio, Y. Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In *International Conference on Machine Learning* (2011).

1259. Rifai, S. *et al.* Higher Order Contractive Auto-Encoder. In *Joint European conference on Machine Learning and Knowledge Discovery in Databases*, 645–660 (Springer, 2011).

1260. Kingma, D. P. & Welling, M. An Introduction to Variational Autoencoders. *arXiv preprint arXiv:1906.02691* (2019).

1261. Doersch, C. Tutorial on Variational Autoencoders. *arXiv preprint arXiv:1606.05908* (2016).

1262. Makhzani, A. & Frey, B. *k*-Sparse Autoencoders. *arXiv preprint arXiv:1312.5663* (2013).

1263. Nair, V. & Hinton, G. E. 3D Object Recognition with Deep Belief Nets. In *Advances in Neural Information Processing Systems*, 1339–1347 (2009).

1264. Arpit, D., Zhou, Y., Ngo, H. & Govindaraju, V. Why Regularized Auto-Encoders Learn Sparse Representation? In *International Conference on Machine Learning*, 136–144 (2016).

1265. Zeng, N. *et al.* Facial Expression Recognition via Learning Deep Sparse Autoencoders. *Neurocomputing* **273**, 643–649 (2018).

1266. Yin, Y., Ouyang, L., Wu, Z. & Yin, S. A Survey of Generative Adversarial Networks Based on Encoder-Decoder Model *Math. Comput. Sci.* **5**, 31 (2020).

1267. Yu, X., Zhang, X., Cao, Y. & Xia, M. VAEGAN: A Collaborative Filtering Framework Based on Adversarial Variational Autoencoders. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 4206–4212 (AAAI Press, 2019).

1268. Larsen, A. B. L., Sønderby, S. K., Larochelle, H. & Winther, O. Autoencoding Beyond Pixels Using a Learned Similarity Metric. In *International Conference on Machine Learning*, 1558–1566 (2016).

1269. Zhuang, F. & Moulin, P. A New Variational Method for Deep Supervised Semantic Image Hashing. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4532–4536 (IEEE, 2020).

1270. Jin, G., Zhang, Y. & Lu, K. Deep Hashing Based on VAE-GAN for Efficient Similarity Retrieval. *Chin. J. Electron.* **28**, 1191–1197 (2019).

1271. Khobahi, S. & Soltanalian, M. Model-Aware Deep Architectures for One-Bit Compressive Variational Autoencoding. *arXiv preprint arXiv:1911.12410* (2019).

1272. Wang, B., Liu, K. & Zhao, J. Deep Semantic Hashing with Multi-Adversarial Training. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 1453–1462 (2018).

1273. Patterson, N. & Wang, Y. Semantic Hashing with Variational Autoencoders (2016).

1274. Fan, Y. *et al.* Video Anomaly Detection and Localization via Gaussian Mixture Fully Convolutional Variational Autoencoder. *Comput. Vis. Image Underst.* **195**, 102920 (2020).

1275. Yao, R., Liu, C., Zhang, L. & Peng, P. Unsupervised Anomaly Detection Using Variational Auto-Encoder Based Feature Extraction. In *2019 IEEE International Conference on Prognostics and Health Management (ICPHM)*, 1–7 (IEEE, 2019).

1276. Xu, H. *et al.* Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications. In *Proceedings of the 2018 World Wide Web Conference*, 187–196 (2018).

1277. An, J. & Cho, S. Variational Autoencoder Based Anomaly Detection Using Reconstruction Probability. *Special Lect. on IE* **2**, 1–18 (2015).

1278. Gauerhof, L. & Gu, N. Reverse Variational Autoencoder for Visual Attribute Manipulation and Anomaly Detection. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2103–2112 (IEEE, 2020).

1279. Klys, J., Snell, J. & Zemel, R. Learning Latent Subspaces in Variational Autoencoders. In *Advances in Neural Information Processing Systems*, 6444–6454 (2018).

1280. Borysov, S. S., Rich, J. & Pereira, F. C. How to Generate Micro-Agents? A Deep Generative Modeling Approach to Population Synthesis. *Transp. Res. Part C: Emerg. Technol.* **106**, 73–97 (2019).

1281. Salim Jr, A. Synthetic Patient Generation: A Deep Learning Approach Using Variational Autoencoders. *arXiv preprint arXiv:1808.06444* (2018).

1282. Gómez-Bombarelli, R. *et al.* Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Cent. Sci.* **4**, 268–276 (2018).

1283. Zhavoronkov, A. *et al.* Deep Learning Enables Rapid Identification of Potent DDR1 Kinase Inhibitors. *Nat. Biotechnol.* **37**, 1038–1040 (2019).

1284. Griffiths, R.-R. & Hernández-Lobato, J. M. Constrained Bayesian Optimization for Automatic Chemical Design Using Variational Autoencoders. *Chem. Sci.* **11**, 577–586 (2020).

1285. Lim, J., Ryu, S., Kim, J. W. & Kim, W. Y. Molecular Generative Model Based on Conditional Variational Autoencoder for *de novo* Molecular Design. *J. Cheminformatics* **10**, 1–9 (2018).

1286. Wan, Z., Zhang, Y. & He, H. Variational Autoencoder Based Synthetic Data Generation for Imbalanced Learning. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1–7 (IEEE, 2017).

1287. Zhang, J. M., Harman, M., Ma, L. & Liu, Y. Machine Learning Testing: Survey, Landscapes and Horizons. *IEEE Transactions on Softw. Eng.* (2020).

1288. Amershi, S. *et al.* Software Engineering for Machine Learning: A Case Study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 291–300 (IEEE, 2019).

1289. Breck, E., Cai, S., Nielsen, E., Salib, M. & Sculley, D. The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction. In *2017 IEEE International Conference on Big Data (Big Data)*, 1123–1132 (IEEE, 2017).

1290. Sculley, D. *et al.* Hidden Technical Debt in Machine Learning Systems. In *Advances in Neural Information Processing Systems*, 2503–2511 (2015).

1291. Li, H., Xu, Z., Taylor, G., Studer, C. & Goldstein, T. Loss landscape mit license. Online: https://github.com/tomgoldstein/loss-landscape/blob/master/LICENSE (2017).

1292. Rodríguez, O. H. & Lopez Fernandez, J. M. A Semiotic Reflection on the Didactics of the Chain Rule. *The Math. Enthus.* **7**, 321–332 (2010).

1293. Kiefer, J. & Wolfowitz, J. Stochastic Estimation of the Maximum of a Regression Function. *The Annals Math. Stat.* **23**, 462–466 (1952).

1294. Robbins, H. & Monro, S. A Stochastic Approximation Method. *The Annals Math. Stat.* **22**, 400–407 (1951).

1295. Polyak, B. T. Some Methods of Speeding up the Convergence of Iteration Methods. *USSR Comput. Math. Math. Phys.* **4**, 1–17 (1964).

1296. Sutskever, I., Martens, J., Dahl, G. & Hinton, G. On the Importance of Initialization and Momentum in Deep Learning. In *International Conference on Machine Learning*, 1139–1147 (2013).

1297. Su, W., Boyd, S. & Candes, E. A Differential Equation for Modeling Nesterov's Accelerated Gradient Method: Theory and Insights. In *Advances in Neural Information Processing Systems*, 2510–2518 (2014).

1298. TensorFlow Source Code for Nesterov Momentum. Online: https://github.com/tensorflow/tensorflow/blob/23c218785eac5bfe737eec4f8081fd0ef8e0684d/tensorflow/python/training/momentum_test.py#L40 (2018).

1299. Ma, J. & Yarats, D. Quasi-Qyperbolic Momentum and ADAM for Deep Learning. *arXiv preprint arXiv:1810.06801* (2018).

1300. Lucas, J., Sun, S., Zemel, R. & Grosse, R. Aggregated Momentum: Stability Through Passive Damping. *arXiv preprint arXiv:1804.00325* (2018).

1301. Hinton, G., Srivastava, N. & Swersky, K. Neural Networks for Machine Learning Lecture 6a Overview of Mini-Batch Gradient Descent. Online: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf (2012).

1302. Kingma, D. P. & Ba, J. ADAM: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* (2014).

1303. Sun, S., Cao, Z., Zhu, H. & Zhao, J. A Survey of Optimization Methods from a Machine Learning Perspective. *IEEE Transactions on Cybern.* **50**, 3668–3681 (2019).

1304. Bottou, L., Curtis, F. E. & Nocedal, J. Optimization Methods for Large-Scale Machine Learning. *Siam Rev.* **60**, 223–311 (2018).

1305. Ruder, S. An Overview of Gradient Descent Optimization Algorithms. *arXiv preprint arXiv:1609.04747* (2016).

1306. Curry, H. B. The Method of Steepest Descent for Non-Linear Minimization Problems. *Q. Appl. Math.* **2**, 258–261 (1944).

1307. Lemaréchal, C. Cauchy and the Gradient Method. *Documenta Math. Extra* **251**, 254 (2012).

1308. Chen, T., Xu, B., Zhang, C. & Guestrin, C. Training Deep Nets with Sublinear Memory Cost. *arXiv preprint arXiv:1604.06174* (2016).

1309. Cybertron AI. Saving Memory Using Gradient-Checkpointing. Online: https://github.com/cybertronai/gradient-checkpointing (2019).

1310. Jin, P., Ginsburg, B. & Keutzer, K. Spatially Parallel Convolutions. *OpenReview.net* (2018).

1311. Whittington, J. C. R. & Bogacz, R. Theories of Error Back-Propagation in the Brain. *Trends Cogn. Sci.* **23**, 235–250 (2019).

1312. Green, C. S. & Bavelier, D. Exercising Your Brain: A Review of Human Brain Plasticity and Training-Induced Learning. *Psychol. Aging* **23**, 692 (2008).

1313. Bassett, D. S. *et al.* Dynamic Reconfiguration of Human Brain Networks During Learning. *Proc. Natl. Acad. Sci.* **108**, 7641–7646 (2011).

1314. O'Doherty, J. P. Reward Representations and Reward-Related Learning in the Human Brain: Insights from Neuroimaging. *Curr. Opin. Neurobiol.* **14**, 769–776 (2004).

1315. Luo, L., Xiong, Y., Liu, Y. & Sun, X. Adaptive Gradient Methods with Dynamic Bound of Learning Rate. *arXiv preprint arXiv:1902.09843* (2019).

1316. Reddi, S. J., Kale, S. & Kumar, S. On the Convergence of ADAM and Beyond. *arXiv preprint arXiv:1904.09237* (2019).

1317. Zhang, M., Lucas, J., Ba, J. & Hinton, G. E. Lookahead Optimizer: k Steps Forward, 1 Step Back. In *Advances in Neural Information Processing Systems*, 9597–9608 (2019).

1318. Dozat, T. Incorporating Nesterov Momentum into ADAM. OpenReview, Online: https://openreview.net/forum?id=OM0jvwB8jIp57ZJjtNEZ (2016).

1319. Huang, H., Wang, C. & Dong, B. Nostalgic Adam: Weighting More of the Past Gradients When Designing the Adaptive Learning Rate. *arXiv preprint arXiv:1805.07557* (2018).

1320. Baiesi, M. Power Gradient Descent. *arXiv preprint arXiv:1906.04787* (2019).

1321. Liu, L. *et al.* On the Variance of the Adaptive Learning Rate and Beyond. *arXiv preprint arXiv:1908.03265* (2019).

1322. Bello, I., Zoph, B., Vasudevan, V. & Le, Q. V. Neural Optimizer Search with Reinforcement Learning. *arXiv preprint arXiv:1709.07417* (2017).

1323. Andrychowicz, M. *et al.* Learning to Learn by Gradient Descent by Gradient Descent. In *Advances in Neural Information Processing Systems*, 3981–3989 (2016).

1324. Li, K. & Malik, J. Learning to Optimize. *arXiv preprint arXiv:1606.01885* (2016).

1325. Hochreiter, S., Younger, A. S. & Conwell, P. R. Learning to Learn Using Gradient Descent. In *International Conference on Artificial Neural Networks*, 87–94 (Springer, 2001).

1326. Duan, Y. *et al.* RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning. *arXiv preprint arXiv:1611.02779* (2016).

1327. Zou, D., Cao, Y., Zhou, D. & Gu, Q. Stochastic Gradient Descent Optimizes Over-Parameterized Deep ReLU Networks. *arXiv preprint arXiv:1811.08888* (2018).

1328. Watt, J. Two Natural Weaknesses of Gradient Descent. Online: https://jermwatt.github.io/machine_learning_refined/notes/3_First_order_methods/3_7_Problems.html (2020).

1329. Goh, G. Why Momentum Really Works. *Distill* (2017).

1330. Qian, N. On the Momentum Term in Gradient Descent Learning Algorithms. *Neural Networks* **12**, 145–151 (1999).

1331. Schmidt, R. M., Schneider, F. & Hennig, P. Descending Through a Crowded Valley – Benchmarking Deep Learning Optimizers. *arXiv preprint arXiv:2007.01547* (2020).

1332. Choi, D. *et al.* On Empirical Comparisons of Optimizers for Deep Learning. *arXiv preprint arXiv:1910.05446* (2019).

1333. Wilson, A. C., Roelofs, R., Stern, M., Srebro, N. & Recht, B. The Marginal Value of Adaptive Gradient Methods in Machine Learning. In *Advances in Neural Information Processing Systems*, 4148–4158 (2017).

1334. Dogo, E., Afolabi, O., Nwulu, N., Twala, B. & Aigbavboa, C. A Comparative Analysis of Gradient Descent-Based Optimization Algorithms on Convolutional Neural Networks. In *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*, 92–99 (IEEE, 2018).

1335. Seetharaman, P., Wichern, G., Pardo, B. & Roux, J. L. AutoClip: Adaptive Gradient Clipping for Source Separation Networks. *arXiv preprint arXiv:2007.14469* (2020).

1336. Gorbunov, E., Danilova, M. & Gasnikov, A. Stochastic Optimization with Heavy-Tailed Noise via Accelerated Gradient Clipping. *arXiv preprint arXiv:2005.10785* (2020).

1337. Yoshida, T. & Ohki, K. Natural Images are Reliably Represented by Sparse and Variable Populations of Neurons in Visual Cortex. *Nat. Commun.* **11**, 1–19 (2020).

1338. Probst, P., Bischl, B. & Boulesteix, A.-L. Tunability: Importance of Hyperparameters of Machine Learning Algorithms. *arXiv preprint arXiv:1802.09596* (2018).

1339. Ge, R., Kakade, S. M., Kidambi, R. & Netrapalli, P. The Step Decay Schedule: A Near Optimal, Geometrically Decaying Learning Rate Procedure. *arXiv preprint arXiv:1904.12838* (2019).

1340. Chen, J. & Kyrillidis, A. Decaying Momentum Helps Neural Network Training. *arXiv preprint arXiv:1910.04952* (2019).

1341. Yang, L. & Shami, A. On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice. *arXiv preprint arXiv:2007.15745* (2020).

1342. Chandra, K. *et al.* Gradient Descent: The Ultimate Optimizer. *arXiv preprint arXiv:1909.13371* (2019).

1343. Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. Optuna: A Next-Generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2623–2631 (2019).

1344. Lakhmiri, D., Digabel, S. L. & Tribes, C. HyperNOMAD: Hyperparameter Optimization of Deep Neural Networks Using Mesh Adaptive Direct Search. *arXiv preprint arXiv:1907.01698* (2019).

1345. Ilievski, I., Akhtar, T., Feng, J. & Shoemaker, C. A. Efficient Hyperparameter Optimization of Deep Learning Algorithms Using Deterministic RBF Surrogates. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 822–829 (AAAI Press, 2017).

1346. Lorenzo, P. R., Nalepa, J., Kawulok, M., Ramos, L. S. & Pastor, J. R. Particle Swarm Optimization for Hyper-Parameter Selection in Deep Neural Networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 481–488 (2017).

1347. Wilamowski, B. M. & Yu, H. Neural Network Learning Without Backpropagation. *IEEE Transactions on Neural Networks* **21**, 1793–1803 (2010).

1348. Blum, A., Dan, C. & Seddighin, S. Learning Complexity of Simulated Annealing. *arXiv preprint arXiv:2003.02981* (2020).

1349. Ingber, L. Simulated Annealing: Practice versus Theory. *Math. Comput. Model.* **18**, 29–57 (1993).

1350. Ayumi, V., Rere, L. R., Fanany, M. I. & Arymurthy, A. M. Optimization of Convolutional Neural Network Using Microcanonical Annealing Algorithm. In *2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 506–511 (IEEE, 2016).

1351. Rere, L. M. R., Fanany, M. I. & Arymurthy, A. M. Simulated Annealing Algorithm for Deep Learning. *Procedia Comput. Sci.* **72**, 137–144 (2015).

1352. Borysenko, O. & Byshkin, M. CoolMomentum: A Method for Stochastic Optimization by Langevin Dynamics with Simulated Annealing. *arXiv preprint arXiv:2005.14605* (2020).

1353. Fischetti, M. & Stringher, M. Embedded Hyper-Parameter Tuning by Simulated Annealing. *arXiv preprint arXiv:1906.01504* (2019).

1354. Sloss, A. N. & Gustafson, S. 2019 Evolutionary Algorithms Review. In *Genetic Programming Theory and Practice XVII*, 307–344 (Springer, 2020).

1355. Al-Sahaf, H. *et al.* A Survey on Evolutionary Machine Learning. *J. Royal Soc. New Zealand* **49**, 205–228 (2019).

1356. Shapiro, J. Genetic Algorithms in Machine Learning. In *Advanced Course on Artificial Intelligence*, 146–168 (Springer, 1999).

1357. Doerr, B., Le, H. P., Makhmara, R. & Nguyen, T. D. Fast genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 777–784 (2017).

1358. Such, F. P. *et al.* Deep Neuroevolution: Genetic Algorithms are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. *arXiv preprint arXiv:1712.06567* (2017).

1359. Sehgal, A., La, H., Louis, S. & Nguyen, H. Deep Reinforcement Learning using Genetic Algorithm for Parameter Optimization. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, 596–601 (IEEE, 2019).

1360. Hu, C., Zuo, Y., Chen, C., Ong, S. P. & Luo, J. Genetic Algorithm-Guided Deep Learning of Grain Boundary Diagrams: Addressing the Challenge of Five Degrees of Freedom. *Mater. Today* **38**, 49–57 (2020).

1361. Jennings, P. C., Lysgaard, S., Hummelshøj, J. S., Vegge, T. & Bligaard, T. Genetic Algorithms for Computational Materials Discovery Accelerated by Machine Learning. *npj Comput. Mater.* **5**, 1–6 (2019).

1362. Nigam, A., Friederich, P., Krenn, M. & Aspuru-Guzik, A. Augmenting Genetic Algorithms with Deep Neural Networks for Exploring the Chemical Space. *arXiv preprint arXiv:1909.11655* (2019).

1363. Potapov, A. & Rodionov, S. Genetic Algorithms with DNN-Based Trainable Crossover as an Example of Partial Specialization of General Search. In *International Conference on Artificial General Intelligence*, 101–111 (Springer, 2017).

1364. Powell, M. J. Direct Search Algorithms for Optimization Calculations. *Acta numerica* **7**, 287–336 (1998).

1365. Ranganathan, V. & Natarajan, S. A New Backpropagation Algorithm Without Gradient Descent. *arXiv preprint arXiv:1802.00027* (2018).

1366. Junior, F. E. F. & Yen, G. G. Particle Swarm Optimization of Deep Neural Networks Architectures for Image Classification. *Swarm Evol. Comput.* **49**, 62–74 (2019).

1367. Qolomany, B., Maabreh, M., Al-Fuqaha, A., Gupta, A. & Benhaddou, D. Parameters Optimization of Deep Learning Models Using Particle Swarm Optimization. In *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, 1285–1290 (IEEE, 2017).

1368. Kennedy, J. & Eberhart, R. Particle Swarm Optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1942–1948 (IEEE, 1995).

1369. Kennedy, J. The Particle Swarm: Social Adaptation of Knowledge. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)*, 303–308 (IEEE, 1997).

1370. Xu, Y. A Review of Machine Learning With Echo State Networks. *Proj. Rep.* (2020).

1371. Jaeger, H. Echo State Network. *Scholarpedia* **2**, 2330 (2007).

1372. Gallicchio, C. & Micheli, A. Deep Echo State Network (DeepESN): A Brief Survey. *arXiv preprint arXiv:1712.04323* (2017).

1373. Alaba, P. A. *et al.* Towards a More Efficient and Cost-Sensitive Extreme Learning Machine: A State-of-the-Art Review of Recent Trend. *Neurocomputing* **350**, 70–90 (2019).

1374. Ghosh, S. *et al.* A Survey on Extreme Learning Machine and Evolution of Its Variants. In *International Conference on Recent Trends in Image Processing and Pattern Recognition*, 572–583 (Springer, 2018).

1375. Albadra, M. A. A. & Tiuna, S. Extreme Learning Machine: A Review. *Int. J. Appl. Eng. Res.* **12**, 4610–4623 (2017).

1376. Tang, J., Deng, C. & Huang, G.-B. Extreme Learning Machine for Multilayer Perceptron. *IEEE Transactions on Neural Networks Learn. Syst.* **27**, 809–821 (2015).

1377. Huang, G.-B., Zhou, H., Ding, X. & Zhang, R. Extreme Learning Machine for Regression and Multiclass Classification. *IEEE Transactions on Syst. Man, Cybern. Part B (Cybernetics)* **42**, 513–529 (2011).

1378. Huang, G.-B., Zhu, Q.-Y. & Siew, C.-K. Extreme Learning Machine: Theory and Applications. *Neurocomputing* **70**, 489–501 (2006).

1379. Huang, G.-B., Zhu, Q.-Y. & Siew, C.-K. Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks. In *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*, vol. 2, 985–990 (IEEE, 2004).

1380. Li, Y. Deep Reinforcement Learning: An Overview. *arXiv preprint arXiv:1701.07274* (2017).

1381. Mondal, A. K. & Jamali, N. A Survey of Reinforcement Learning Techniques: Strategies, Recent Development, and Future Directions. *arXiv preprint arXiv:2001.06921* (2020).

1382. Haney, B. S. Deep Reinforcement Learning Patents: An Empirical Survey. *Available at SSRN 3570254* (2020).

1383. Nguyen, T. T., Nguyen, N. D. & Nahavandi, S. Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications. *IEEE Transactions on Cybern.* **50**, 3826–3839 (2020).

1384. Botvinick, M. *et al.* Reinforcement Learning, Fast and Slow. *Trends Cogn. Sci.* **23**, 408–422 (2019).

1385. Recht, B. A Tour of Reinforcement Learning: The View From Continuous Control. *Annu. Rev. Control. Robotics, Auton. Syst.* **2**, 253–279 (2019).

1386. Arulkumaran, K., Deisenroth, M. P., Brundage, M. & Bharath, A. A. A Brief Survey of Deep Reinforcement Learning. *arXiv preprint arXiv:1708.05866* (2017).

1387. Kiran, B. R. *et al.* Deep Reinforcement Learning for Autonomous Driving: A Survey. *arXiv preprint arXiv:2002.00444* (2020).

1388. Nageshrao, S., Tseng, H. E. & Filev, D. Autonomous Highway Driving Using Deep Reinforcement Learning. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2326–2331 (IEEE, 2019).

1389. Talpaert, V. *et al.* Exploring Applications of Deep Reinforcement Learning for Real-World Autonomous Driving Systems. *arXiv preprint arXiv:1901.01536* (2019).

1390. Luong, N. C. *et al.* Applications of Deep Reinforcement Learning in Communications and Networking: A Survey. *IEEE Commun. Surv. & Tutorials* **21**, 3133–3174 (2019).

1391. Di Felice, M., Bedogni, L. & Bononi, L. *Reinforcement Learning-Based Spectrum Management for Cognitive Radio Networks: A Literature Review and Case Study*, 1–38 (Springer Singapore, Singapore, 2018).

1392. Han, M. *et al.* A Review of Reinforcement Learning Methodologies for Controlling Occupant Comfort in Buildings. *Sustain. Cities Soc.* **51**, 101748 (2019).

1393. Mason, K. & Grijalva, S. A Review of Reinforcement Learning for Autonomous Building Energy Management. *Comput. & Electr. Eng.* **78**, 300–312 (2019).

1394. Mnih, V. *et al.* Human-Level Control Through Deep Reinforcement Learning. *Nature* **518**, 529–533 (2015).

1395. Nguyen, H. & La, H. Review of Deep Reinforcement Learning for Robot Manipulation. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, 590–595 (IEEE, 2019).

1396. Bhagat, S., Banerjee, H., Ho Tse, Z. T. & Ren, H. Deep Reinforcement Learning for Soft, Flexible Robots: Brief Review with Impending Challenges. *Robotics* **8**, 4 (2019).

1397. Zhao, T., Hachiya, H., Niu, G. & Sugiyama, M. Analysis and Improvement of Policy Gradient Estimation. In *Advances in Neural Information Processing Systems*, 262–270 (2011).

1398. Weng, L. Exploration strategies in deep reinforcement learning. Online: https://lilianweng.github.io/lil-log/2020/06/07/exploration-strategies-in-deep-reinforcement-learning.html (2020).

1399. Plappert, M. *et al.* Parameter Space Noise for Exploration. *arXiv preprint arXiv:1706.01905* (2018).

1400. Uhlenbeck, G. E. & Ornstein, L. S. On the Theory of the Brownian Motion. *Phys. Rev.* **36**, 823 (1930).

1401. Fujimoto, S., Van Hoof, H. & Meger, D. Addressing Function Approximation Error in Actor-Critic Methods. *arXiv preprint arXiv:1802.09477* (2018).

1402. Barth-Maron, G. *et al.* Distributed Distributional Deterministic Policy Gradients. *arXiv preprint arXiv:1804.08617* (2018).

1403. Kosaka, N. *Has it Explored Enough?* Master's thesis, Royal Holloway University of London, DOI: 10.13140/RG.2.2.11584.89604 (2019).

1404. Fortunato, M. *et al.* Noisy Networks for Exploration. *arXiv preprint arXiv:1706.10295* (2019).

1405. Hazan, E., Kakade, S., Singh, K. & Van Soest, A. Provably Efficient Maximum Entropy Exploration. In *International Conference on Machine Learning*, 2681–2691 (2019).

1406. Haarnoja, T., Tang, H., Abbeel, P. & Levine, S. Reinforcement Learning with Deep Energy-Based Policies. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1352–1361 (2017).

1407. Ahmed, Z., Le Roux, N., Norouzi, M. & Schuurmans, D. Understanding the Impact of Entropy on Policy Optimization. In *International Conference on Machine Learning*, 151–160 (2019).

1408. Aubret, A., Matignon, L. & Hassas, S. A Survey on Intrinsic Motivation in Reinforcement Learning. *arXiv preprint arXiv:1908.06976* (2019).

1409. Linke, C., Ady, N. M., White, M., Degris, T. & White, A. Adapting Behaviour via Intrinsic Reward: A Survey and Empirical Study. *arXiv preprint arXiv:1906.07865* (2019).

1410. Pathak, D., Agrawal, P., Efros, A. A. & Darrell, T. Curiosity-Driven Exploration by Self-Supervised Prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 16–17 (2017).

1411. Hoi, S. C., Sahoo, D., Lu, J. & Zhao, P. Online Learning: A Comprehensive Survey. *arXiv preprint arXiv:1802.02871* (2018).

1412. Wei, C.-Y., Hong, Y.-T. & Lu, C.-J. Online Reinforcement Learning in Stochastic Games. In *Advances in Neural Information Processing Systems*, 4987–4997 (2017).

1413. Levine, S., Kumar, A., Tucker, G. & Fu, J. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *arXiv preprint arXiv:2005.01643* (2020).

1414. Seita, D. Offline (Batch) Reinforcement Learning: A Review of Literature and Applications. Seita's Place, Online: https://danieltakeshi.github.io/2020/06/28/offline-rl (2020).

1415. Fedus, W. *et al.* Revisiting Fundamentals of Experience Replay. *arXiv preprint arXiv:2007.06700* (2020).

1416. Nair, A., Dalal, M., Gupta, A. & Levine, S. Accelerating Online Reinforcement Learning with Offline Datasets. *arXiv preprint arXiv:2006.09359* (2020).

1417. Lin, L.-J. Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching. *Mach. Learn.* **8**, 293–321 (1992).

1418. Zhang, S. & Sutton, R. S. A Deeper Look at Experience Replay. *arXiv preprint arXiv:1712.01275* (2017).

1419. He, X., Zhao, K. & Chu, X. AutoML: A Survey of the State-of-the-Art. *arXiv preprint arXiv:1908.00709* (2019).

1420. Malekhosseini, E., Hajabdollahi, M., Karimi, N. & Samavi, S. Modeling Neural Architecture Search Methods for Deep Networks. *arXiv preprint arXiv:1912.13183* (2019).

1421. Jaafra, Y., Laurent, J. L., Deruyver, A. & Naceur, M. S. Reinforcement Learning for Neural Architecture Search: A Review. *Image Vis. Comput.* **89**, 57–66 (2019).

1422. Elsken, T., Metzen, J. H. & Hutter, F. Neural Architecture Search: A Survey. *arXiv preprint arXiv:1808.05377* (2018)

1423. Waring, J., Lindvall, C. & Umeton, R. Automated Machine Learning: Review of the State-of-the-Art and Opportunities for Healthcare. *Artif. Intell. Medicine* **104**, 101822 (2020).

1424. Weill, C. *et al.* AdaNet: A Scalable and Flexible Framework for Automatically Learning Ensembles (2019). 1905.00080

1425. Weill, C. Introducing AdaNet: Fast and Flexible AutoML with Learning Guarantees. Google AI Blog, Online: https://ai.googleblog.com/2018/10/introducing-adanet-fast-and-flexible.html (2018).

1426. Liu, C. *et al.* Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 82–92 (2019).

1427. Gong, X., Chang, S., Jiang, Y. & Wang, Z. AutoGAN: Neural Architecture Search for Generative Adversarial Networks In *Proceedings of the IEEE International Conference on Computer Vision*, 3224–3234 (2019).

1428. Jin, H., Song, Q. & Hu, X. Auto-Keras: An Efficient Neural Architecture Search System. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1946–1956 (2019).

1429. Feurer, M. *et al.* Efficient and Robust Automated Machine Learning. In *Advances in Neural Information Processing Systems*, 2962–2970 (2015).

1430. Liang, H. *et al.* DARTS+: Improved Differentiable Architecture Search with Early Stopping. *arXiv preprint arXiv:1909.06035* (2019).

1431. LeDell, E. & Poirier, S. H2O AutoML: Scalable Automatic Machine Learning. In *Proceedings of the AutoML Workshop at ICML*, vol. 2020 (2020).

1432. Molino, P., Dudin, Y. & Miryala, S. S. Ludwig: A Type-Based Declarative Deep Learning Toolbox. *arXiv preprint arXiv:1909.07930* (2019).

1433. Young, S. R., Rose, D. C., Karnowski, T. P., Lim, S.-H. & Patton, R. M. Optimizing Deep Learning Hyper-Parameters Through an Evolutionary Algorithm. In *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, 1–5 (2015).

1434. Patton, R. M. *et al.* 167-PFLOPS Deep Learning for Electron Microscopy: From Learning Physics to Atomic Manipulation. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, 638–648 (IEEE, 2018).

1435. Kandasamy, K., Neiswanger, W., Schneider, J., Poczos, B. & Xing, E. P. Neural Architecture Search with Bayesian Optimisation and Optimal Transport. In *Advances in Neural Information Processing Systems*, 2016–2025 (2018).

1436. Nayman, N. *et al.* XNAS: Neural Architecture Search with Expert Advice. In *Advances in Neural Information Processing Systems*, 1977–1987 (2019).

1437. Jiang, W. *et al.* Accuracy vs. Efficiency: Achieving Both Through FPGA-Implementation Aware Neural Architecture Search. In *Proceedings of the 56th Annual Design Automation Conference 2019*, 1–6 (2019).

1438. Liu, C. *et al.* Progressive Neural Architecture Search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 19–34 (2018).

1439. Zhang, C., Ren, M. & Urtasun, R. Graph Hypernetworks for Neural Architecture Search. *arXiv preprint arXiv:1810.05749* (2018).

1440. Baker, B., Gupta, O., Raskar, R. & Naik, N. Accelerating Neural Architecture Search Using Performance Prediction *arXiv preprint arXiv:1705.10823* (2017).

1441. Zoph, B. & Le, Q. V. Neural Architecture Search with Reinforcement Learning. *arXiv preprint arXiv:1611.01578* (2016).

1442. Hanussek, M., Blohm, M. & Kintz, M. Can AutoML Outperform Humans? An Evaluation on Popular OpenML Datasets Using AutoML Benchmark. *arXiv preprint arXiv:2009.01564* (2020).

1443. Godoy, D. Hyper-Parameters in Action! Part II – Weight Initializers. Towards Data Science, Online: https://towardsdatascience.com/hyper-parameters-in-action-part-ii-weight-initializers-35aee1a28404 (2018).

1444. Nagarajan, V. & Kolter, J. Z. Generalization in Deep Networks: The Role of Distance From Initialization. *arXiv preprint arXiv:1901.01672* (2019).

1445. Glorot, X. & Bengio, Y. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256 (2010).

1446. Kumar, S. K. On Weight Initialization in Deep Neural Networks. *arXiv preprint arXiv:1704.08863* (2017).

1447. Saxe, A. M., McClelland, J. L. & Ganguli, S. Exact Solutions to the Nonlinear Dynamics of Learning in Deep Linear Neural Networks. *arXiv preprint arXiv:1312.6120* (2013).

1448. Henaff, M., Szlam, A. & LeCun, Y. Recurrent Orthogonal Networks and Long-Memory Tasks. *arXiv preprint arXiv:1602.06662* (2016).

1449. Le, Q. V., Jaitly, N. & Hinton, G. E. A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. *arXiv preprint arXiv:1504.00941* (2015).

1450. Mikolov, T., Joulin, A., Chopra, S., Mathieu, M. & Ranzato, M. Learning Longer Memory in Recurrent Neural Networks. *arXiv preprint arXiv:1412.7753* (2014).

1451. Pitis, S. Non-Zero Initial States for Recurrent Neural Networks. Online: https://r2rt.com/non-zero-initial-states-for-recurrent-neural-networks.html (2016).

1452. Mishkin, D. & Matas, J. All You Need is a Good Init. *arXiv preprint arXiv:1511.06422* (2015).

1453. Sussillo, D. & Abbott, L. Random Walk Initialization for Training Very Deep Feedforward Networks. *arXiv preprint arXiv:1412.6558* (2014).

1454. Dauphin, Y. N. & Schoenholz, S. MetaInit: Initializing Learning by Learning to Initialize. In *Advances in Neural Information Processing Systems*, 12645–12657 (2019).

1455. Kukačka, J., Golkov, V. & Cremers, D. Regularization for Deep Learning: A Taxonomy. *arXiv preprint arXiv:1710.10686* (2017).

1456. Kang, G. *Regularization in Deep Neural Networks*. Ph.D. thesis, University of Technology Sydney (2019).

1457. Liu, Z., Li, X., Kang, B. & Darrell, T. Regularization Matters in Policy Optimization. *arXiv preprint arXiv:1910.09191* (2019).

1458. Vettam, S. & John, M. Regularized Deep Learning with Non-Convex Penalties. *arXiv preprint arXiv:1909.05142* (2019).

1459. Golatkar, A. S., Achille, A. & Soatto, S. Time Matters in Regularizing Deep Networks: Weight Decay and Data Augmentation Affect Early Learning Dynamics, Matter Little Near Convergence. In *Advances in Neural Information Processing Systems*, 10678–10688 (2019).

1460. Tanay, T. & Griffin, L. D. A New Angle on L2 Regularization. *arXiv preprint arXiv:1806.11186* (2018).

1461. Van Laarhoven, T. L2 Regularization versus Batch and Weight Normalization. *arXiv preprint arXiv:1706.05350* (2017).

1462. Van Den Doel, K., Ascher, U. & Haber, E. The Lost Honour of L2-Based Regularization. *Large Scale Inverse Probl.* **13**, 181–203 (2012).

1463. Gribonval, R., Cevher, V. & Davies, M. E. Compressible Distributions for High-Dimensional Statistics. *IEEE Transactions on Inf. Theory* **58**, 5016–5034 (2012).

1464. Ng, A. Y. Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 78 (2004).

1465. Zou, H. & Hastie, T. Regularization and Variable Selection via the Elastic Net. *J. Royal Stat. Soc. Ser. B (Statistical Methodol.* **67**, 301–320 (2005).

1466. Tibshirani, R. Regression Shrinkage and Selection via the Lasso. *J. Royal Stat. Soc. Ser. B (Methodological)* **58**, 267–288 (1996).

1467. Hoerl, A. E. & Kennard, R. W. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics* **12**, 55–67 (1970).

1468. Zhang, J., He, T., Sra, S. & Jadbabaie, A. Why Gradient Clipping Accelerates Training: A Theoretical Justification for Adaptivity. *arXiv preprint arXiv:1905.11881* (2019).

1469. Chen, X., Wu, Z. S. & Hong, M. Understanding Gradient Clipping in Private SGD: A Geometric Perspective. *arXiv preprint arXiv:2006.15429* (2020).

1470. Menon, A. K., Rawat, A. S., Reddi, S. J. & Kumar, S. Can Gradient Clipping Mitigate Label Noise? In *International Conference on Learning Representations* (2019).

1471. Bengio, Y., Boulanger-Lewandowski, N. & Pascanu, R. Advances in Optimizing Recurrent Networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 8624–8628 (IEEE, 2013).

1472. Chen, M. X. *et al.* The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation. *arXiv preprint arXiv:1804.09849* (2018).

1473. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The J. Mach. Learn. Res.* **15**, 1929–1958 (2014).

1474. Labach, A., Salehinejad, H. & Valaee, S. Survey of Dropout Methods for Deep Neural Networks. *arXiv preprint arXiv:1904.13310* (2019).

1475. Li, Z., Gong, B. & Yang, T. Improved Dropout for Shallow and Deep Learning. In *Advances in Neural Information Processing Systems*, 2523–2531 (2016).

1476. Mianjy, P., Arora, R. & Vidal, R. On the Implicit Bias of Dropout. In *International Conference on Machine Learning*, 3540–3548 (2018).

1477. Warde-Farley, D., Goodfellow, I. J., Courville, A. & Bengio, Y. An Empirical Analysis of Dropout in Piecewise Linear Networks. *arXiv preprint arXiv:1312.6197* (2013).

1478. Garbin, C., Zhu, X. & Marques, O. Dropout vs. Batch Normalization: An Empirical Study of Their Impact to Deep Learning. *Multimed. Tools Appl.* **79**, 12777–12815 (2020).

1479. Cai, S. *et al.* Effective and Efficient Dropout for Deep Convolutional Neural Networks. *arXiv preprint arXiv:1904.03392* (2019).

1480. Ghiasi, G., Lin, T.-Y. & Le, Q. V. DropBlock: A Regularization Method for Convolutional Networks. In *Advances in Neural Information Processing Systems*, 10727–10737 (2018).

1481. Faramarzi, M., Amini, M., Badrinaaraayanan, A., Verma, V. & Chandar, S. PatchUp: A Regularization Technique for Convolutional Neural Networks. *arXiv preprint arXiv:2006.07794* (2020).

1482. Kang, G., Li, J. & Tao, D. Shakeout: A New Approach to Regularized Deep Neural Network Training. *IEEE Transactions on Pattern Analysis Mach. Intell.* **40**, 1245–1258 (2017).

1483. Kang, G., Li, J. & Tao, D. Shakeout: A New Regularized Deep Neural Network Training Scheme. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 1751–1757 (2016).

1484. Zhou, M. *et al.* Towards Understanding the Importance of Noise in Training Neural Networks. *arXiv preprint arXiv:1909.03172* (2019).

1485. Graves, A., Mohamed, A.-r. & Hinton, G. Speech Recognition with Deep Recurrent Neural Networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 6645–6649 (IEEE, 2013).

1486. Graves, A. Practical Variational Inference for Neural Networks. In *Advances in Neural Information Processing Systems*, 2348–2356 (2011).

1487. Sum, J., Leung, C.-S. & Ho, K. A Limitation of Gradient Descent Learning. *IEEE Transactions on Neural Networks Learn. Syst.* **31**, 2227–2232 (2019).

1488. Holmstrom, L. & Koistinen, P. Using Additive Noise in Back-Propagation Training. *IEEE Transactions on Neural Networks* **3**, 24–38 (1992).

1489. You, Z., Ye, J., Li, K., Xu, Z. & Wang, P. Adversarial Noise Layer: Regularize Neural Network by Adding Noise. In *2019 IEEE International Conference on Image Processing (ICIP)*, 909–913 (IEEE, 2019).

1490. Jenni, S. & Favaro, P. On Stabilizing Generative Adversarial Training with Noise. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 12145–12153 (2019).

1491. Sun, Y., Tian, Y., Xu, Y. & Li, J. Limited Gradient Descent: Learning With Noisy Labels. *IEEE Access* **7**, 168296–168306 (2019).

1492. Simsekli, U., Sagun, L. & Gurbuzbalaban, M. A Tail-Index Analysis of Stochastic Gradient Noise in Deep Neural Networks. *arXiv preprint arXiv:1901.06053* (2019).

1493. Neelakantan, A. *et al.* Adding Gradient Noise Improves Learning for Very Deep Networks. *arXiv preprint arXiv:1511.06807* (2015).

1494. Shorten, C. & Khoshgoftaar, T. M. A Survey on Image Data Augmentation for Deep Learning. *J. Big Data* **6**, 60 (2019).

1495. Raileanu, R., Goldstein, M., Yarats, D., Kostrikov, I. & Fergus, R. Automatic Data Augmentation for Generalization in Deep Reinforcement Learning. *arXiv preprint arXiv:2006.12862* (2020).

1496. Antczak, K. On Regularization Properties of Artificial Datasets for Deep Learning. *arXiv preprint arXiv:1908.07005* (2019).

1497. Ouali, Y., Hudelot, C. & Tami, M. An Overview of Deep Semi-Supervised Learning. *arXiv preprint arXiv:2006.05278* (2020).

1498. Zhu, J. Semi-Supervised Learning: the Case When Unlabeled Data is Equally Useful. *arXiv preprint arXiv:2005.11018* (2020).

1499. Aitchison, L. A Statistical Theory of Semi-Supervised Learning. *arXiv preprint arXiv:2008.05913* (2020).

1500. Bagherzadeh, J. & Asil, H. A Review of Various Semi-Supervised Learning Models with a Deep Learning and Memory Approach. *Iran J. Comput. Sci.* **2**, 65–80 (2019).

1501. Rasmus, A., Berglund, M., Honkala, M., Valpola, H. & Raiko, T. Semi-Supervised Learning with Ladder Networks. In *Advances in Neural Information Processing Systems*, 3546–3554 (2015).

1502. Lee, D.-H. Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. In *Workshop on Challenges in Representation Learning, ICML*, vol. 3 (2013).

1503. Sun, S., Mao, L., Dong, Z. & Wu, L. Multiview Transfer Learning and Multitask Learning. In *Multiview Machine Learning*, 85–104 (Springer, 2019).

1504. Ruder, S. An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv preprint arXiv:1706.05098* (2017).

1505. Thung, K.-H. & Wee, C.-Y. A Brief Review on Multi-Task Learning. *Multimed. Tools Appl.* **77**, 29705–29725 (2018).

1506. Zhang, Y. & Yang, Q. A Survey on Multi-Task Learning. *arXiv preprint arXiv:1707.08114* (2017).

1507. Caruana, R. Multitask Learning. *Mach. Learn.* **28**, 41–75 (1997).

1508. Odena, A., Olah, C. & Shlens, J. Conditional Image Synthesis With Auxiliary Classifier GANs. *arXiv preprint arXiv:1610.09585* (2016).

1509. Shu, R., Bui, H. & Ermon, S. AC-GAN Learns a Biased Distribution. In *NIPS Workshop on Bayesian Deep Learning*, vol. 8 (2017).

1510. Gong, M., Xu, Y., Li, C., Zhang, K. & Batmanghelich, K. Twin Auxilary Classifiers GAN. In *Advances in Neural Information Processing Systems*, 1330–1339 (2019).

1511. Han, L., Stathopoulos, A., Xue, T. & Metaxas, D. Unbiased Auxiliary Classifier GANs with MINE. *arXiv preprint arXiv:2006.07567* (2020).

1512. Better Performance with the tf.data API. TensorFlow Documentation, Online: https://www.tensorflow.org/guide/data_performance (2020).

1513. Li, B., Wu, F., Lim, S.-N., Belongie, S. & Weinberger, K. Q. On feature normalization and data augmentation. *arXiv preprint arXiv:2002.11102* (2020).

1514. Bhanja, S. & Das, A. Impact of Data Normalization on Deep Neural Network for Time Series Forecasting. *arXiv preprint arXiv:1812.05519* (2018).

1515. van Hasselt, H. P., Guez, A., Hessel, M., Mnih, V. & Silver, D. Learning Values Across Many Orders of Magnitude. In *Advances in Neural Information Processing Systems*, 4287–4295 (2016).

1516. Li, M., Soltanolkotabi, M. & Oymak, S. Gradient Descent with Early Stopping is Provably Robust to Label Noise for Overparameterized Neural Networks. In *International Conference on Artificial Intelligence and Statistics*, 4313–4324 (2020).

1517. Flynn, T., Yu, K. M., Malik, A., D'Imperio, N. & Yoo, S. Bounding the Expected Run-Time of Nonconvex Optimization with Early Stopping. *arXiv preprint arXiv:2002.08856* (2020).

1518. Nagaraj, D., Jain, P. & Netrapalli, P. SGD Without Replacement: Sharper Rates for General Smooth Convex Functions. In *International Conference on Machine Learning*, 4703–4711 (2019).

1519. Gürbüzbalaban, M., Ozdaglar, A. & Parrilo, P. Why Random Reshuffling Beats Stochastic Gradient Descent. *Math. Program.* (2019).

1520. Haochen, J. & Sra, S. Random Shuffling Beats SGD After Finite Epochs. In *International Conference on Machine Learning*, 2624–2633 (2019).

1521. Shamir, O. Without-Replacement Sampling for Stochastic Gradient Methods. In *Advances in Neural Information Processing Systems*, 46–54 (2016).

1522. Bottou, L. Curiously Fast Convergence of Some Stochastic Gradient Descent Algorithms. In *Proceedings of the Symposium on Learning and Data Science* (2009).

1523. tf.data.Dataset. TensorFlow Documentation, Online: https://www.tensorflow.org/api_docs/python/tf/data/Dataset (2020).

1524. Harrington, P. d. B. Multiple Versus Single Set Validation of Multivariate Models to Avoid Mistakes. *Critical Rev. Anal. Chem.* **48**, 33–46 (2018).

1525. Breiman, L. Bagging Predictors. *Mach. Learn.* **24**, 123–140 (1996).

1526. Breiman, L. Random Forests. *Mach. Learn.* **45**, 5–32 (2001).

1527. Goel, E., Abhilasha, E., Goel, E. & Abhilasha, E. Random Forest: A Review. *Int. J. Adv. Res. Comput. Sci. Softw. Eng.* **7**, 251–257 (2017).

1528. Probst, P., Wright, M. N. & Boulesteix, A.-L. Hyperparameters and Tuning Strategies for Random Forest. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **9**, e1301 (2019).

1529. Xu, Y. & Goodacre, R. On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning. *J. Analysis Test.* **2**, 249–262 (2018).

1530. Guyon, I. A Scaling Law for the Validation-Set Training-Set Size Ratio. *AT&T Bell Lab.* **1** (1997).

1531. Newman, M. E. J. Power Laws, Pareto Distributions and Zipf's Law. *Contemp. Phys.* **46**, 323–351 (2005).

1532. Opeyemi, B. Deployment of Machine Learning Models Demystified (Part 1). Towards Data Science, Online: https://towardsdatascience.com/deployment-of-machine-learning-model-demystified-part-1-1181d91815d2 (2019).

1533. Opeyemi, B. Deployment of Machine Learning Model Demystified (Part 2). Medium, Online: https://medium.com/@opeyemibami/deployment-of-machine-learning-models-demystified-part-2-63eadaca1571 (2019).

1534. Wu, C.-J. *et al.* Machine Learning at Facebook: Understanding Inference at the Edge. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 331–344 (IEEE, 2019).

1535. Cai, H., Gan, C. & Han, S. Once for All: Train One Network and Specialize it for Efficient Deployment. *arXiv preprint arXiv:1908.09791* (2019).

1536. Suresh, A. & Ganesh Kumar, P. Optimization of Metascheduler for Cloud Machine Learning Services. *Wirel. Pers. Commun.* **114**, 367–388 (2020).

1537. Kumar, Y., Kaul, S. & Sood, K. Effective Use of the Machine Learning Approaches on Different Clouds. In *Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur-India* (2019).

1538. Dubois, D. J., Trubiani, C. & Casale, G. Model-driven Application Refactoring to Minimize Deployment Costs in Preemptible Cloud Resources. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, 335–342 (IEEE, 2016).

1539. Oracle *et al.* GraphPipe: Machine Learning Model Deployment Made Simple.

1540. FlatBuffers: Memory Efficient Serialization Library. FlatBuffers Documentation, Online: https://google.github.io/flatbuffers (2020).

1541. Blalock, D., Ortiz, J. J. G., Frankle, J. & Guttag, J. What is the State of Neural Network Pruning? *arXiv preprint arXiv:2003.03033* (2020).

1542. Pasandi, M. M., Hajabdollahi, M., Karimi, N. & Samavi, S. Modeling of Pruning Techniques for Deep Neural Networks Simplification. *arXiv preprint arXiv:2001.04062* (2020).

1543. Wu, H., Judd, P., Zhang, X., Isaev, M. & Micikevicius, P. Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation. *arXiv preprint arXiv:2004.09602* (2020).

1544. Nayak, P., Zhang, D. & Chai, S. Bit Efficient Quantization for Deep Neural Networks. *arXiv preprint arXiv:1910.04877* (2019).

1545. Zhou, Y., Moosavi-Dezfooli, S.-M., Cheung, N.-M. & Frossard, P. Adaptive Quantization for Deep Neural Network. *arXiv preprint arXiv:1712.01048* (2017).

1546. Yang, J. *et al.* Quantization Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7308–7316 (2019).

1547. Zhuang, B. *et al.* Effective Training of Convolutional Neural Networks with Low-Bitwidth Weights and Activations. *arXiv preprint arXiv:1908.04680* (2019).

1548. Li, H. *et al.* Training Quantized Nets: A Deeper Understanding. In *Advances in Neural Information Processing Systems*, 5811–5821 (2017).

1549. Wang, S. & Kanwar, P. BFloat16: The Secret to High Performance on Cloud TPUs. Google Cloud, Online: https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus (2019).

1550. Marco, V. S., Taylor, B., Wang, Z. & Elkhatib, Y. Optimizing Deep Learning Inference on Embedded Systems Through Adaptive Model Selection. *ACM Transactions on Embed. Comput. Syst. (TECS)* **19**, 1–28 (2020).

1551. Jackson, B. How to Optimize Images for Web and Performance. Kinsta Blog, Online: https://kinsta.com/blog/optimize-images-for-web (2020).

1552. Leventić, H., Nenadić, K., Galić, I. & Livada, Č. Compression Parameters Tuning for Automatic Image Optimization in Web Applications. In *2016 International Symposium ELMAR*, 181–184 (IEEE, 2016).

1553. Olah, C., Mordvintsev, A. & Schubert, L. Feature Visualization. Distill, Online: https://distill.pub/2017/feature-visualization (2017).

1554. Xie, N., Ras, G., van Gerven, M. & Doran, D. Explainable Deep Learning: A Field Guide for the Uninitiated. *arXiv preprint arXiv:2004.14545* (2020).

1555. Vilone, G. & Longo, L. Explainable Artificial Intelligence: A Systematic Review. *arXiv preprint arXiv:2006.00093* (2020).

1556. Arrieta, A. B. *et al.* Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges Toward Responsible AI. *Inf. Fusion* **58**, 82–115 (2020).

1557. Puiutta, E. & Veith, E. Explainable Reinforcement Learning: A Survey. *arXiv preprint arXiv:2005.06247* (2020).

1558. Gunning, D. & Aha, D. W. DARPA's Explainable Artificial Intelligence Program. *AI Mag.* **40**, 44–58 (2019).

1559. Samek, W. & Müller, K.-R. Towards Explainable Artificial Intelligence. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, 5–22 (Springer, 2019).

1560. Hase, P. & Bansal, M. Evaluating Explainable AI: Which Algorithmic Explanations Help Users Predict Model Behavior? *arXiv preprint arXiv:2005.01831* (2020).

1561. Ullah, I. *et al.* A Brief Survey of Visual Saliency Detection. *Multimed. Tools Appl.* **79**, 34605–34645 (2020).

1562. Borji, A., Cheng, M.-M., Hou, Q., Jiang, H. & Li, J. Salient Object Detection: A Survey. *Comput. Vis. Media* 1–34 (2019).

1563. Cong, R. *et al.* Review of Visual Saliency Detection with Comprehensive Information. *IEEE Transactions on circuits Syst. for Video Technol.* **29**, 2941–2959 (2018).

1564. Borji, A., Cheng, M.-M., Jiang, H. & Li, J. Salient Object Detection: A Benchmark. *IEEE Transactions on Image Process.* **24**, 5706–5722 (2015).

1565. Rebuffi, S.-A., Fong, R., Ji, X. & Vedaldi, A. There and Back Again: Revisiting Backpropagation Saliency Methods. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8839–8848 (2020).

1566. Wang, Y., Su, H., Zhang, B. & Hu, X. Learning Reliable Visual Saliency for Model Explanations. *IEEE Transactions on Multimed.* **22**, 1796–1807 (2019).

1567. Kim, B. *et al.* Why are Saliency Maps Noisy? Cause of and Solution to Noisy Saliency Maps. *arXiv preprint arXiv:1902.04893* (2019).

1568. Selvaraju, R. R. *et al.* Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, 618–626 (2017).

1569. Morbidelli, P., Carrera, D., Rossi, B., Fragneto, P. & Boracchi, G. Augmented Grad-CAM: Heat-Maps Super Resolution Through Augmentation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4067–4071 (IEEE, 2020).

1570. Omeiza, D., Speakman, S., Cintas, C. & Weldermariam, K. Smooth Grad-CAM++: An Enhanced Inference Level Visualization Technique for Deep Convolutional Neural Network Models. *arXiv preprint arXiv:1908.01224* (2019).

1571. Chattopadhay, A., Sarkar, A., Howlader, P. & Balasubramanian, V. N. Grad-Cam++: Generalized Gradient-Based Visual Explanations for Deep Convolutional Networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 839–847 (IEEE, 2018).

1572. Patro, B. N., Lunayach, M., Patel, S. & Namboodiri, V. P. U-Cam: Visual Explanation Using Uncertainty Based Class Activation Maps. In *Proceedings of the IEEE International Conference on Computer Vision*, 7444–7453 (2019).

1573. Borji, A. Saliency Prediction in the Deep Learning Era: Successes and Limitations. *IEEE Transactions on Pattern Analysis Mach. Intell.* (2019).

1574. Wang, W. *et al.* Revisiting Video Saliency Prediction in the Deep Learning Era. *IEEE Transactions on Pattern Analysis Mach. Intell.* (2019).

1575. Chen, L., Chen, J., Hajimirsadeghi, H. & Mori, G. Adapting Grad-CAM for Embedding Networks. In *The IEEE Winter Conference on Applications of Computer Vision*, 2794–2803 (2020).

1576. Ramaswamy, H. G. *et al.* Ablation-CAM: Visual Explanations for Deep Convolutional Network via Gradient-free Localization. In *The IEEE Winter Conference on Applications of Computer Vision*, 983–991 (2020).

1577. Wang, H. *et al.* Score-CAM: Score-Weighted Visual Explanations for Convolutional Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 24–25 (2020).

1578. Cancela, B., Bolón-Canedo, V., Alonso-Betanzos, A. & Gama, J. A Scalable Saliency-Based Feature Selection Method with Instance-Level Information. *Knowledge-Based Syst.* **192**, 105326 (2020).

1579. Cheng, M.-M., Mitra, N. J., Huang, X., Torr, P. H. & Hu, S.-M. Global Contrast Based Salient Region Detection. *IEEE Transactions on Pattern Analysis Mach. Intell.* **37**, 569–582 (2014).

1580. Nguyen, A., Yosinski, J. & Clune, J. Understanding Neural Networks via Feature Visualization: A Survey. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, 55–76 (Springer, 2019).

1581. Xiao, W. & Kreiman, G. Gradient-Free Activation Maximization for Identifying Effective Stimuli. *arXiv preprint arXiv:1905.00378* (2019).

1582. Erhan, D., Bengio, Y., Courville, A. & Vincent, P. Visualizing Higher-Layer Features of a Deep Network. *Univ. Montr.* **1341** (2009).

1583. Mordvintsev, A., Olah, C. & Tyka, M. Inceptionism: Going Deeper into Neural Networks. Google AI Blog, Online: https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html (2015).

1584. Maaten, L. v. d. & Hinton, G. Visualizing Data Using t-SNE. *J. Mach. Learn. Res.* **9**, 2579–2605 (2008).

1585. Wattenberg, M., Viégas, F. & Johnson, I. How to Use t-SNE Effectively. *Distill* **1**, e2 (2016).

1586. Van Der Maaten, L. Barnes-Hut-SNE. *arXiv preprint arXiv:1301.3342* (2013).

1587. Barnes, J. & Hut, P. A Hierarchical$O(N \log N)$ Force-Calculation Algorithm. *Nature* **324**, 446–449 (1986).

1588. Wang, Z. J. *et al.* CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization. *arXiv preprint arXiv:2004.15004* (2020).

1589. Wang, Z. J. *et al.* CNN 101: Interactive Visual Learning for Convolutional Neural Networks. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–7 (2020).

1590. Kahng, M., Thorat, N., Chau, D. H. P., Viégas, F. B. & Wattenberg, M. GAN Lab: Understanding Complex Deep Generative Models Using Interactive Visual Experimentation. *IEEE Transactions on Vis. Comput. Graph.* **25**, 1–11 (2018).

1591. Gangavarapu, T., Jaidhar, C. & Chanduka, B. Applicability of Machine Learning in Spam and Phishing Email Filtering: Review and Approaches. *Artif. Intell. Rev.* **53**, 5019–5081 (2020).

1592. Dada, E. G. *et al.* Machine Learning for Email Spam Filtering: Review, Approaches and Open Research Problems. *Heliyon* **5**, e01802 (2019).

1593. Bhuiyan, H., Ashiquzzaman, A., Juthi, T. I., Biswas, S. & Ara, J. A Survey of Existing E-Mail Spam Filtering Methods Considering Machine Learning Techniques. *Glob. J. Comput. Sci. Technol.* **18** (2018).

1594. Zhang, J. & Zeng, W. Mining Scientific and Technical Literature: From Knowledge Extraction to Summarization. In *Trends and Applications of Text Summarization Techniques* (IGI Global, 2020).

1595. Dangovski, R., Jing, L., Nakov, P., Tatalović, M. & Soljačić, M. Rotational Unit of Memory: A Novel Representation Unit for RNNs with Scalable Applications. *Transactions Assoc. for Comput. Linguist.* **7**, 121–138 (2019).

1596. Scholarcy: The AI-Powered Article Summarizer. Online: https://www.scholarcy.com (2020).

1597. Romanov, A., Lomotin, K. & Kozlova, E. Application of Natural Language Processing Algorithms to the Task of Automatic Classification of Russian Scientific Texts. *Data Sci. J.* **18**, 37 (2019).

1598. Gonçalves, S., Cortez, P. & Moro, S. A Deep Learning Classifier for Sentence Classification in Biomedical and Computer Science Abstracts. *Neural Comput. Appl.* **32**, 6793–6807 (2019).

1599. Hughes, M., Li, I., Kotoulas, S. & Suzumura, T. Medical Text Classification Using Convolutional Neural Networks. *Stud. Heal. Technol. Informatics* **235**, 246–50 (2017).

1600. Liu, J., Xu, Y. & Zhu, Y. Automated Essay Scoring Based on Two-Stage Learning. *arXiv preprint arXiv:1901.07744* (2019).

1601. Dong, F., Zhang, Y. & Yang, J. Attention-Based Recurrent Convolutional Neural Network for Automatic Essay Scoring. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, 153–162 (2017).

1602. Taghipour, K. & Ng, H. T. A Neural Approach to Automated Essay Scoring. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, 1882–1891 (2016).

1603. Alikaniotis, D., Yannakoudakis, H. & Rei, M. Automatic Text Scoring Using Neural Networks. *arXiv preprint arXiv:1606.04289* (2016).

1604. Foltỳnek, T., Meuschke, N. & Gipp, B. Academic Plagiarism Detection: A Systematic Literature Review. *ACM Comput. Surv. (CSUR)* **52**, 1–42 (2019).

1605. Meuschke, N., Stange, V., Schubotz, M., Kramer, M. & Gipp, B. Improving Academic Plagiarism Detection for STEM documents by Analyzing Mathematical Content and Citations. In *2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, 120–129 (IEEE, 2019).

1606. Ullah, F., Wang, J., Farhan, M., Habib, M. & Khalid, S. Software Plagiarism Detection in Multiprogramming Languages Using Machine Learning Approach. *Concurr. Comput. Pract. Exp.* e5000 (2018).

1607. Lakkaraju, H. *et al.* A Machine Learning Framework to Identify Students at Risk of Adverse Academic Outcomes. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1909–1918 (2015).

1608. Foster, D. *Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play* (O'Reilly Media, 2019).

1609. Zhan, H., Dai, L. & Huang, Z. Deep Learning in the Field of Art. In *Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science*, 717–719 (2019).

1610. Dhariwal, P. *et al.* Jukebox: A Generative Model for Music. *arXiv preprint arXiv:2005.00341* (2020).

1611. Briot, J.-P. & Pachet, F. Deep Learning for Music Generation: Challenges and Directions. *Neural Comput. Appl.* **32**, 981–993 (2020).

1612. Briot, J.-P., Hadjeres, G. & Pachet, F.-D. *Deep Learning Techniques for Music Generation* (Springer, 2020).

1613. Brown, T. B. *et al.* Language Models are Few-Shot Learners. *arXiv preprint arXiv:2005.14165* (2020).

1614. Radford, A. *et al.* Better Language Models and Their Implications. OpenAI Blog, Online: https://openai.com/blog/better-language-models (2019).

1615. Chen, H., Le, T. H. M. & Babar, M. A. Deep Learning for Source Code Modeling and Generation: Models, Applications and Challenges. *ACM Comput. Surv. (CSUR)* **53** (2020).

1616. Allamanis, M., Barr, E. T., Devanbu, P. & Sutton, C. A Survey of Machine Learning for Big Code and Naturalness. *ACM Comput. Surv. (CSUR)* **51**, 1–37 (2018).

1617. Autocompletion with deep learning. TabNine Blog, Online: https://www.tabnine.com/blog/deep (2019).

1618. Svyatkovskiy, A., Deng, S. K., Fu, S. & Sundaresan, N. IntelliCode Compose: Code Generation Using Transformer. *arXiv preprint arXiv:2005.08025* (2020).

1619. Hammad, M., Babur, Ö., Basit, H. A. & Brand, M. v. d. DeepClone: Modeling Clones to Generate Code Predictions. *arXiv preprint arXiv:2007.11671* (2020).

1620. Schuster, R., Song, C., Tromer, E. & Shmatikov, V. You Autocomplete Me: Poisoning Vulnerabilities in Neural Code Completion. *arXiv preprint arXiv:2007.02220* (2020).

1621. Svyatkovskoy, A. *et al.* Fast and Memory-Efficient Neural Code Completion. *arXiv preprint arXiv:2004.13651* (2020).

1622. Hellendoorn, V. J., Proksch, S., Gall, H. C. & Bacchelli, A. When Code Completion Fails: A Case Study on Real-World Completions. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, 960–970 (IEEE, 2019).

1623. Balog, M., Gaunt, A. L., Brockschmidt, M., Nowozin, S. & Tarlow, D. DeepCoder: Learning to Write Programs. In *International Conference on Learning Representations (ICLR 2017)* (OpenReview.net, 2017).

1624. Murali, V., Qi, L., Chaudhuri, S. & Jermaine, C. Neural Sketch Learning for Conditional Program Generation. *arXiv preprint arXiv:1703.05698* (2018).

1625. Demir, S., Mutlu, U. & Özdemir, Ö. Neural Academic Paper Generation. *arXiv preprint arXiv:1912.01982* (2019).

1626. SciNote. Manuscript Writer. Online: https://www.scinote.net/manuscript-writer (2020).

1627. Stribling, J., Krohn, M. & Aguayo, D. SCIgen - An Automatic CS Paper Generator. Online: https://pdos.csail.mit.edu/archive/scigen (2005).

1628. Raghu, M. & Schmidt, E. A Survey of Deep Learning for Scientific Discovery. *arXiv preprint arXiv:2003.11755* (2020).

1629. Kepner, J., Cho, K. & Claffy, K. New Phenomena in Large-Scale Internet Traffic. *arXiv preprint cs.NI/1904.04396* (2019).

1630. Adekitan, A. I., Abolade, J. & Shobayo, O. Data Mining Approach for Predicting the Daily Internet Data Traffic of a Smart University. *J. Big Data* **6**, 11 (2019).

1631. Xu, X., Wang, J., Peng, H. & Wu, R. Prediction of Academic Performance Associated with Internet Usage Behaviors Using Machine Learning Algorithms. *Comput. Hum. Behav.* **98**, 166–173 (2019).

1632. Granger, R. Toward the Quantification of Cognition. *arXiv preprint arXiv:2008.05580* (2020).

1633. Musk, E. *et al.* An Integrated Brain-Machine Interface Platform with Thousands of Channels. *J. Med. Internet Res.* **21**, e16194 (2019).

1634. Tshitoyan, V. *et al.* Unsupervised Word Embeddings Capture Latent Knowledge from Materials Science Literature. *Nature* **571**, 95–98 (2019).

1635. Ruf, J. & Wang, W. Neural Networks for Option Pricing and Hedging: A Literature Review. *J. Comput. Finance, Forthcom.* **24** (2020).

1636. Huang, B., Huan, Y., Xu, L. D., Zheng, L. & Zou, Z. Automated Trading Systems Statistical and Machine Learning Methods and Hardware Implementation: A Survey. *Enterp. Inf. Syst.* **13**, 132–144 (2019).

1637. Raghavan, M., Barocas, S., Kleinberg, J. & Levy, K. Mitigating Bias in Algorithmic Hiring: Evaluating Claims and Practices. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 469–481 (2020).

1638. Mahmoud, A. A., Shawabkeh, T. A., Salameh, W. A. & Al Amro, I. Performance Predicting in Hiring Process and Performance Appraisals Using Machine Learning. In *2019 10th International Conference on Information and Communication Systems (ICICS)*, 110–115 (IEEE, 2019).

1639. Raub, M. Bots, Bias and Big Data: Artificial Intelligence, Algorithmic Bias and Disparate Impact Liability in Hiring Practices. *Arkansas Law Rev.* **71**, 529 (2018).

1640. Newman, N. Reengineering Workplace Bargaining: How Big Data Drives Lower Wages and How Reframing Labor Law can Restore Information Equality in the Workplace. *Univ. Cincinnati Law Rev.* **85**, 693 (2017).

1641. Price, W. & Nicholson, I. Grants. *Berkeley Technol. Law J.* **34**, 1 (2019).

1642. Zhuang, H. & Acuna, D. E. The Effect of Novelty on the Future Impact of Scientific Grants. *arXiv preprint arXiv:1911.02712* (2019).

1643. Zhang, W. E., Sheng, Q. Z., Alhazmi, A. & Li, C. Adversarial Attacks on Deep-Learning Models in Natural Language Processing: A survey. *ACM Transactions on Intell. Syst. Technol. (TIST)* **11**, 1–41 (2020).

1644. Ma, X. *et al.* Understanding Adversarial Attacks on Deep Learning Based Medical Image Analysis Systems. *Pattern Recognit.* **110**, 107332 (2020).

1645. Yuan, X., He, P., Zhu, Q. & Li, X. Adversarial examples: Attacks and Defenses for Deep Learning. *IEEE Transactions on Neural Networks Learn. Syst.* **30**, 2805–2824 (2019).

1646. Akhtar, N. & Mian, A. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. *IEEE Access* **6**, 14410–14430 (2018).

1647. Goodfellow, I. J., Shlens, J. & Szegedy, C. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572* (2014).

1648. Wen, Y., Li, S. & Jia, K. Towards Understanding the Regularization of Adversarial Robustness on Neural Networks. *OpenReview.net* (2019).

1649. Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D. & Jana, S. Certified Robustness to Adversarial Examples with Differential Privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, 656–672 (IEEE, 2019).

1650. Li, Y. *et al.* Optimal Transport Classifier: Defending Against Adversarial Attacks by Regularized Deep Embedding. *arXiv preprint arXiv:1811.07950* (2018).

1651. Xie, C. *et al.* Adversarial Examples Improve Image Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 819–828 (2020).

1652. Deniz, O., Pedraza, A., Vallez, N., Salido, J. & Bueno, G. Robustness to Adversarial Examples can be Improved With Overfitting. *Int. J. Mach. Learn. Cybern.* **11**, 935–944 (2020).

1653. Kinoshita, Y. & Kiya, H. Fixed Smooth Convolutional Layer for Avoiding Checkerboard Artifacts in CNNs. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3712–3716 (IEEE, 2020).

1654. Xiao, H., Rasul, K. & Vollgraf, R. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747* (2017).

1.2 Reflection

This introductory chapter covers my review paper[96] titled "Review: Deep Learning in Electron Microscopy"[1]. It is the first in-depth review of deep learning in electron microscopy and offers a practical perspective that is aimed at developers with limited familiarity. My review was crafted to be covered by the introductory chapter of my PhD thesis, so focus is placed on my research methodology. Going through its sections in order of appearance, "Introduction" covers and showcases my earlier research, "Resources" introduces resources that enabled my research, "Electron Microscopy" covers how I simulated exit wavefunctions and integrated ANNs with electron microscopes, "Components" introduces functions used to construct my ANNs, "Architecture" details ANN archetypes used in my research, "Optimization" covers how my ANNs were trained, and "Discussion" offers my perspective on deep learning in electron microscopy.

There are many review papers on deep learning. Some reviews of deep learning focus on computer science[97–101], whereas others focus on specific applications such as computational imaging[102], materials science[103–105], and the physical sciences[106]. As a result, I anticipated that another author might review deep learning in electron microscopy. To avoid my review being easily surpassed, I leveraged my experience to offer practical perspectives and comparative discussions to address common causes of confusion. In addition, content is justified by extensive references to make it easy to use as a starting point for future research. Finally, I was concerned that information about how to get started with deep learning in electron microscopy was fragmented and unclear to unfamiliar developers. This was often problematic when I was asked about getting started with machine learning, and I was especially conscious of it as my friend, Rajesh Patel, asked me for advice when I started writing my review. Consequently, I included a section that introduces useful resources for deep learning in electron microscopy.

Chapter 2

Warwick Electron Microscopy Datasets

2.1 Scientific Paper

This paper covers the following paper[2] and its supplementary information[9].

> J. M. Ede. Warwick Electron Microscopy Datasets. *Machine Learning: Science and Technology*, 1(4): 045003, 2020
>
> J. M. Ede. Supplementary Information: Warwick Electron Microscopy Datasets. Zenodo, Online: `https://doi.org/10.5281/zenodo.3899740`, 2020

IOP Publishing Mach. Learn.: Sci. Technol. 1 (2020) 045003 https://doi.org/10.1088/2632-2153/ab9c3c

MACHINE
LEARNING
Science and Technology

CrossMark

OPEN ACCESS

RECEIVED
13 March 2020

REVISED
20 May 2020

ACCEPTED FOR PUBLICATION
12 June 2020

PUBLISHED
11 September 2020

Original Content from this work may be used under the terms of the Creative Commons Attribution 4.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

Warwick electron microscopy datasets

Jeffrey M Ede
University of Warwick, Department of Physics, Coventry, CV4 7AL, United Kingdom
E-mail: j.m.ede@warwick.ac.uk

Keywords: datasets, electron microscopy, machine learning, variational autoencoder, t-distributed stochastic neighbor embedding, visualization

Abstract

Large, carefully partitioned datasets are essential to train neural networks and standardize performance benchmarks. As a result, we have set up new repositories to make our electron microscopy datasets available to the wider community. There are three main datasets containing 19769 scanning transmission electron micrographs, 17266 transmission electron micrographs, and 98340 simulated exit wavefunctions, and multiple variants of each dataset for different applications. To visualize image datasets, we trained variational autoencoders to encode data as 64-dimensional multivariate normal distributions, which we cluster in two dimensions by t-distributed stochastic neighbor embedding. In addition, we have improved dataset visualization with variational autoencoders by introducing encoding normalization and regularization, adding an image gradient loss, and extending t-distributed stochastic neighbor embedding to account for encoded standard deviations. Our datasets, source code, pretrained models, and interactive visualizations are openly available at https://github.com/Jeffrey-Ede/datasets.

1. Introduction

We have set up new repositories [1] to make our large new electron microscopy datasets available to both electron microscopists and the wider community. There are three main datasets containing 19769 experimental scanning transmission electron microscopy [2] (STEM) images, 17266 experimental transmission electron microscopy [2] (TEM) images and 98340 simulated TEM exit wavefunctions [3]. Experimental datasets represent general research and were collected by dozens of University of Warwick scientists working on hundreds of projects between January 2010 and June 2018. We have been using our datasets to train artificial neural networks (ANNs) for electron microscopy [3–7], where standardizing results with common test sets has been essential for comparison. This paper provides details of and visualizations for datasets and their variants, and is supplemented by source code, pretrained models, and both static and interactive visualizations [8].

Machine learning is increasingly being applied to materials science [9, 10], including to electron microscopy [11]. Encouraging scientists, ANNs are universal approximators [12] that can leverage an understanding of physics to represent [13] the best way to perform a task with arbitrary accuracy. In theory, this means that ANNs can always match or surpass the performance of contemporary methods. However, training, validating and testing requires large, carefully partitioned datasets [14, 15] to ensure that ANNs are robust to general use. To this end, our datasets are partitioned so that each subset has different characteristics. For example, TEM or STEM images can be partitioned so that subsets are collected by different scientists, and simulated exit wavefunction partitions can correspond to Crystallography Information Files [16] (CIFs) for materials published in different journals.

Most areas of science are facing a reproducibility crisis [17], including artificial intelligence [18]. Adding to this crisis, natural scientists do not always benchmark ANNs against standardized public domain test sets; making results difficult or impossible to compare. In electron microscopy, we believe this is a symptom of most datasets being small, esoteric or not having default partitions for machine learning. For example, most datasets in the Electron Microscopy Public Image Archive [19, 20] are for specific materials and are not partitioned. In contrast, standard machine learning datasets such as CIFAR-10 [21, 22], MNIST [23], and ImageNet [24] have default partitions for machine learning and contain tens of thousands or millions of examples. By publishing our large, carefully partitioned machine learning datasets, and setting an example by using them to standardize our research, we aim to encourage higher standardization of machine learning research in the electron microscopy community.

There are many popular algorithms for high-dimensional data visualization [25–32] that can map N high-dimensional vectors of features $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$, $\mathbf{x}_i \in \mathbb{R}^u$ to low-dimensional vectors $\{\mathbf{y}_1, ..., \mathbf{y}_N\}$, $\mathbf{y}_i \in \mathbb{R}^v$. A standard approach for data clustering in $v \in \{1, 2, 3\}$ dimensions is t-distributed stochastic neighbor embedding [33, 34] (tSNE). To embed data by tSNE, Kullback-Leibler (KL) divergence,

$$L_{\text{tSNE}} = \sum_i \sum_{j \neq i} p_{ij} \log \left(\frac{p_{ij}}{q_{ij}} \right) , \tag{1}$$

is minimized by gradient descent [35] for normally distributed pairwise similarities in real space, p_{ij}, and heavy-tailed Student t-distributed pairwise similarities in an embedding space, q_{ij}. For symmetric tSNE [33],

$$p_{i|j} = \frac{\exp\left(-||\mathbf{x}_i - \mathbf{x}_j||_2^2 / 2\alpha_j^2\right)}{\sum\limits_{k \neq j} \exp\left(-||\mathbf{x}_k - \mathbf{x}_j||_2^2 / 2\alpha_j^2\right)} , \tag{2}$$

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N} , \tag{3}$$

$$q_{ij} = \frac{\left(1 + ||\mathbf{y}_i - \mathbf{y}_j||_2^2\right)^{-1}}{\sum\limits_{k \neq i} (1 + ||\mathbf{y}_k - \mathbf{y}_i||_2^2)^{-1}} . \tag{4}$$

To control how much tSNE clusters data, perplexities of $p_{i|j}$ for $j \in \{1, ..., N\}$ are adjusted to a user-provided value by fitting α_j. Perplexity, $\exp(H)$, is an exponential function of entropy, H, and most tSNE visualizations are robust to moderate changes to its value.

Feature extraction is often applied to decrease input dimensionality, typically to $u \lesssim 100$, before clustering data by tSNE. Decreasing input dimensionality can decrease data noise and computation for large datasets, and is necessary for some high-dimensional data as distances, $||\mathbf{x}_i - \mathbf{x}_j||_2$, used to compute p_{ij} are affected by the curse of dimensionality [36]. For image data, a standard approach [33] to extract features is probabilistic [37, 38] or singular value decomposition [39] (SVD) based principal component analysis [40] (PCA). However, PCA is limited to linearly separable features. Other hand-crafted feature extraction methods include using a histogram of oriented gradients [41], speeded-up robust features [42], local binary patterns [43], wavelet decomposition [44] and other methods [45]. The best features to extract for a visualization depend on its purpose. However, most hand-crafted feature extraction methods must be tuned for different datasets. For example, Minka's algorithm [46] is included in the scikit-learn [47] implementation of PCA by SVD to obtain optimal numbers of principal components to use.

To increase representation power, nonlinear and dataset-specific features can be extracted with deep learning. For example, by using the latent space of an autoencoder [48, 49] (AE) or features before logits in a classification ANN [50]. Indeed, we have posted AEs for electron microscopy with pre-trained models [51, 52] that could be improved. However, AE latent vectors can exhibit inhomogeneous dimensional characteristics and pathological semantics, limiting correlation between latent features and semantics. To encode well-behaved latent vectors suitable for clustering by tSNE, variational autoencoders [53, 54] (VAEs) can be trained to encode data as multivariate probability distributions. For example, VAEs are often regularized to encode multivariate normal distributions by adding KL divergence of encodings from a standard normal distribution to its loss function [53]. The regularization homogenizes dimensional characteristics and sampling noise correlates semantics with latent features.

2. Dataset visualization

To visualize datasets presented in this paper, we trained VAEs shown in figure 1 to embed 96× 96 images in $u = 64$ dimensions before clustering in $v = 2$ dimensions by tSNE. Our VAE consists of two convolutional neural networks [55, 56] (CNNs): an encoder and a generator. The encoder embeds batches of B input images, I, as mean vectors, $\{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_B\}$, and standard deviation vectors, $\{\boldsymbol{\sigma}_1, ..., \boldsymbol{\sigma}_B\}$, to parameterize multivariate normal distributions. During training, input images are linearly transformed to have minimum

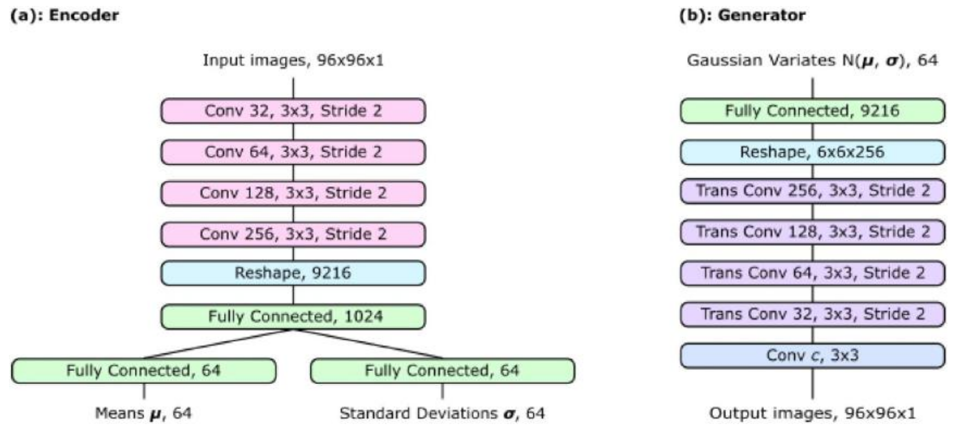


Figure 1. Simplified VAE architecture. (a) An encoder outputs means, $\boldsymbol{\mu}$, and standard deviations, $\boldsymbol{\sigma}$, to parameterize multivariate normal distributions, $\mathbf{z} \sim \mathrm{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$. (b) A generator predicts input images from $\mathbf{z}$.

and maximum values of 0 and 1, respectively, and we apply a random combination of flips and 90° rotations to augment training data by a factor of eight. The generator, G, is trained to cooperate with the encoder to output encoder inputs by sampling latent vectors, $\mathbf{z}_i = \boldsymbol{\mu}_i + \boldsymbol{\sigma}_i \boldsymbol{\epsilon}_i$, where $\boldsymbol{\mu}_i = \{\mu_{i1}, \ldots, \mu_{iu}\}$, $\boldsymbol{\sigma}_i = \{\sigma_{i1}, \ldots, \sigma_{iu}\}$, and $\boldsymbol{\epsilon}_i = \{\epsilon_{i1}, \ldots, \epsilon_{iu}\}$ are random variates sampled from standard normal distributions, $\varepsilon_{ij} \sim N(0, 1)$. Each convolutional or fully connected layer is followed by batch normalization [57] then ReLU [58] activation, except the output layers of the encoder and generator. An absolute nonlinearity, $f(x) = |x|$, is applied to encode positive standard deviations.

Traditional VAEs are trained to optimize a balance, λ_{MSE}, between mean squared errors (MSEs) of generated images and KL divergence of encodings from a multivariate standard normal distribution [53],

$$L_{\mathrm{trad}} = \lambda_{\mathrm{MSE}} \mathrm{MSE}(G(\mathbf{z}), I) + \frac{1}{2Bu} \sum_{i=1}^{B} \sum_{j=1}^{u} \mu_{ij}^2 + \sigma_{ij}^2 - \log(\sigma_{ij}^2) - 1. \tag{5}$$

However, traditional VAE training is sensitive to λ_{MSE} [59] and other hyperparameters [60]. If λ_{MSE} is too low, the encoder will learn learn to consistently output $\sigma_{ij} \simeq 1$, limiting regularization. Else if λ_{MSE} is too high, the encoder will learn to output $\sigma_{ij} \ll |\mu_{ij}|$, limiting regularization. As a result, traditional VAE hyperparameters must be carefully tuned for different ANN architectures and datasets. To improve VAE regularization and robustness to different datasets, we normalize encodings parameterizing normal distributions to

$$\mu_{ij} \leftarrow \frac{\lambda_{\mu}(\mu_{ij} - \mu_{\mathrm{avg},j})}{\mu_{\mathrm{std},j}}, \tag{6}$$

$$\sigma_{ij} \leftarrow \frac{\sigma_{ij}}{2\sigma_{\mathrm{std},j}}, \tag{7}$$

where batch means and standard deviations are

$$\mu_{\mathrm{avg},j} = \frac{1}{B} \sum_{k=1}^{B} \mu_{kj}, \tag{8}$$

$$\mu_{\mathrm{std},j}^2 = \frac{1}{B} \sum_{k=1}^{B} \mu_{kj}^2 - \left(\frac{1}{B} \sum_{k=1}^{B} \mu_{kj} \right)^2, \tag{9}$$

$$\sigma_{\mathrm{std},j}^2 = \frac{1}{B} \sum_{k=1}^{B} \sigma_{kj}^2 - \left(\frac{1}{B} \sum_{k=1}^{B} \sigma_{kj} \right)^2. \tag{10}$$

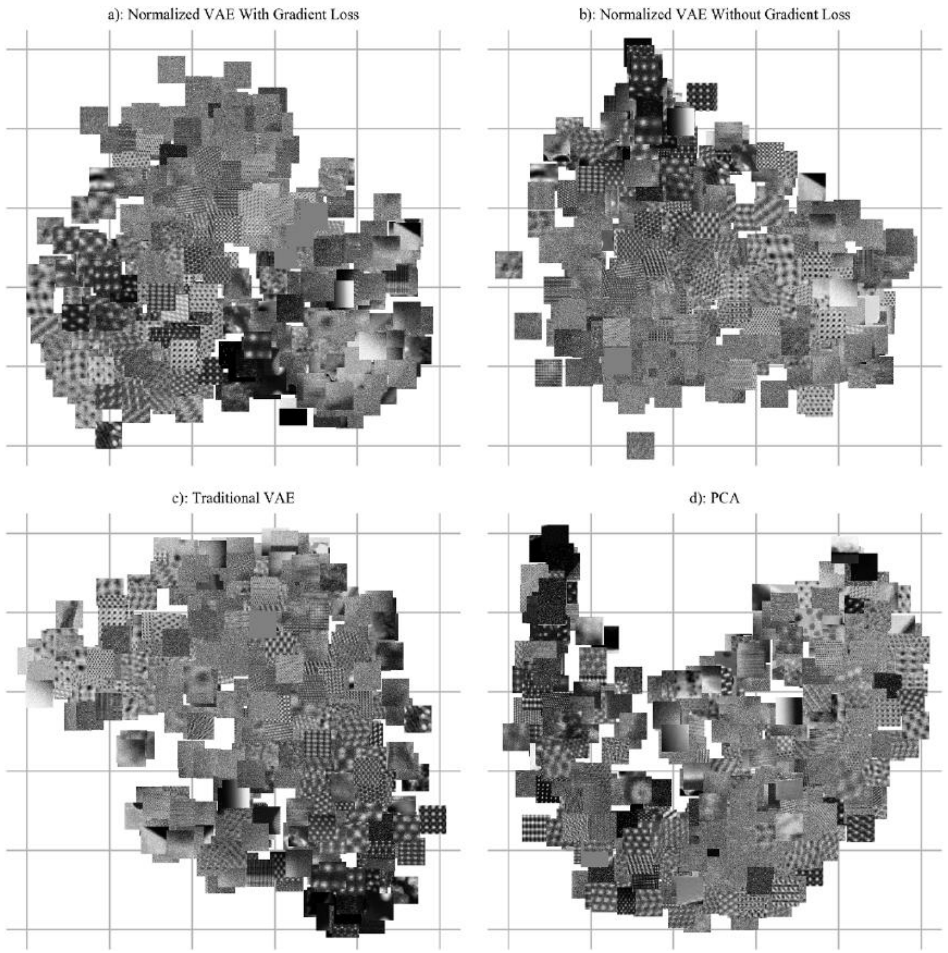


Figure 2. Images at 500 randomly selected images in two-dimensional tSNE visualizations of 19769 96× 96 crops from STEM images for various embedding methods. Clustering is best in (a) and gets worse in order (a)→(b)→(c)→(d).

Encoding normalization is a modified form of batch normalization [57] for VAE latent spaces. As part of encoding normalization, we introduce a new hyperparameter, λ_μ, to scale the ratio of expectations $\mathrm{E}(|\mu_{ij}|)/\mathrm{E}(|\sigma_{ij}|)$. We use $\lambda_\mu = 2.5$ in this paper; however, we confirm that training is robust to values $\lambda_\mu \in \{1.0, 2.0, 2.5\}$ for a range of datasets and ANN architectures. Batch means are subtracted from $\boldsymbol{\mu}$ and not $\boldsymbol{\sigma}$ so that $\sigma_{ij} \geq 0$. In addition, we multiply $\sigma_{\mathrm{std},j}$ by an arbitrary factor of 2 so that $\mathrm{E}(|\mu_{ij}|) \approx \mathrm{E}(|\sigma_{ij}|)$ for $\lambda_\mu = 1$.

Encoding normalization enables the KL divergence loss in equation 5 to be removed as latent space regularization is built into the encoder architecture. However, we find that removing the KL loss can result in VAEs encoding either very low or very high σ_{ij}. In effect, an encoder can learn to use $\boldsymbol{\sigma}$ apply a binary mask to $\boldsymbol{\mu}$ if a generator learns that latent features with very high absolute values are not meaningful. To prevent extreme σ_{ij}, we add a new encoding regularization loss, $\mathrm{MSE}(\boldsymbol{\sigma}, 1)$, to the encoder. Human vision is sensitive to edges [61], so we also add a gradient-based loss to improve realism. Adding a gradient-based loss is a computationally inexpensive alternative to training a variational autoencoder generative adversarial network [62] (VAE-GAN) and often achieves similar performance. Our total training loss is

$$L = \lambda_{\mathrm{MSE}} \mathrm{MSE}(G(\mathbf{z}), I) + \lambda_{\mathrm{Sobel}} \mathrm{MSE}(S(G(\mathbf{z})), S(I)) + \mathrm{MSE}(\boldsymbol{\sigma}, 1)\,, \tag{11}$$

where we chose $\lambda_{\mathrm{MSE}} = \lambda_{\mathrm{Sobel}} = 50$, and $S(x)$ computes a concatenation of horizontal and vertical Sobel derivatives [63] of x. We found that training is robust to choices of $\lambda_{\mathrm{MSE}} = \lambda_{\mathrm{Sobel}}$ where $\lambda_{\mathrm{MSE}} \mathrm{MSE}(G(\mathbf{z}), I) + \lambda_{\mathrm{Sobel}} \mathrm{MSE}(S(G(\mathbf{z})), S(I))$ is in $[0.5, 25.0]$, and have not investigated losses outside this interval. We trained VAEs to minimize L by ADAM [64] optimized stochastic gradient descent [35, 65]. At training iteration $t \in [1, T]$, we used a stepwise exponentially decayed learning rate [66],

$$\eta = \eta_{\text{start}} a^{\text{floor}(bt/T)} \,, \tag{12}$$

and a DEMON [67] first moment of the momentum decay rate,

$$\beta_1 = \frac{\beta_{\text{start}}(1 - t/T)}{(1 - \beta_{\text{start}}) + \beta_{\text{start}}(1 - t/T)} \,, \tag{13}$$

where we chose initial values $\eta_{\text{start}} = 0.001$ and $\beta_{\text{start}} = 0.9$, exponential base $a = 0.5$, $b = 8$ steps, and $T = 600000$ iterations. We used a batch size of $B = 64$ and emphasize that a large batch size decreases complication of encoding normalization by varying batch statistics. Training our VAEs takes about 12 hours on a desktop computer with an Nvidia GTX 1080 Ti GPU and an Intel i7-6700 CPU. To use VAE latent spaces to cluster data, means are often embedded by tSNE. However, this does not account for highly varying σ used to calculate latent features. To account for uncertainty, we modify calculation of pairwise similarities, $p_{i|j}$, in equation 2 to include both $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i$ encoded for every example, $i \in [1, N]$, in our datasets,

$$p_{i|j} = \exp\left(-\frac{1}{2\alpha_j^2}\sum_k w_{ijk}(\mu_{ik} - \mu_{jk})^2\right)\left(\sum_{m \neq j}\exp\left(-\frac{1}{2\alpha_j^2}\sum_k w_{mjk}(\mu_{mk} - \mu_{jk})^2\right)\right)^{-1} \,, \tag{14}$$

where we chose weights

$$w_{ijk} = \frac{1}{\sigma_{ik}^2 + \sigma_{jk}^2 + \epsilon}\left(\sum_l \frac{1}{\sigma_{il}^2 + \sigma_{jl}^2 + \epsilon}\right)^{-1} \,. \tag{15}$$

We add $\varepsilon = 0.01$ for numerical stability, and to account for uncertainty in $\boldsymbol{\sigma}$ due to encoder imperfections or variation in batch statistics. Following Oskolkov [68], we fit α_j to perplexities given by $N^{1/2}$, where N is the number of examples in a dataset, and confirm that changing perplexities by ± 100 has little effect on visualizations for our $N \simeq 20000$ TEM and STEM datasets. To ensure convergence, we run tSNE computations for 10000 iterations. In comparison, KL divergence is stable by 5000 iterations for our datasets. In preliminary experiments, we observe that tSNE with $\boldsymbol{\sigma}$ results in comparable visualizations to tSNE without $\boldsymbol{\sigma}$, and we think that tSNE with $\boldsymbol{\sigma}$ may be a slight improvement. For comparison, pairs of visualizations with and without $\boldsymbol{\sigma}$ are indicated in supplementary information.

Our improvements to dataset visualization by tSNE are showcased in figure 2 for various embedding methods. The visualizations are for a new dataset containing 19769 96×96 crops from STEM images, which will be introduced in section 3. To suppress high-frequency noise during training, images were blurred by a 5×5 symmetric Gaussian kernel with a 2.5 px standard deviation. Clusters are most distinct in figure 2(a) for encoding normalized VAE training with a gradient loss described by equation 11. Ablating the gradient loss in figure 2(b) results in similar clustering; however, the VAE struggles to separate images of noise and fine atom columns. In contrast, clusters are not clearly separated in figure 2(c) for a traditional VAE described by equation 5. Finally, embedding the first 50 principal components extracted by a scikit-learn [69] implementation of probabilistic PCA in figure 2(d) does not result in clear clustering.

3. Scanning transmission electron micrographs

We curated 19769 STEM images from University of Warwick electron microscopy dataservers to train ANNs for compressed sensing [5, 7]. Atom columns are visible in roughly two-thirds of images, and similar proportions are bright and dark field. In addition, most signals are noisy [76] and are imaged at several times their Nyquist rates [77]. To reduce data transfer times for large images, we also created variant containing 161069 non-overlapping 512×512 crops from full images. For rapid development, we have also created new variants containing 96×96 images downsampled or cropped from full images. In this section we give details of each STEM dataset, referring to them using their names in our repositories.

STEM Full Images: 19769 32-bit TIFFs containing STEM images taken with a University of Warwick JEOL ARM 200F electron microscope by dozens of scientists working on hundreds of projects. Images were originally saved in DigitalMicrograph DM3 or DM4 files created by Gatan Microscopy Suite [78] software and have their original sizes and intensities. The dataset is partitioned into 14826 training, 1977 validation,

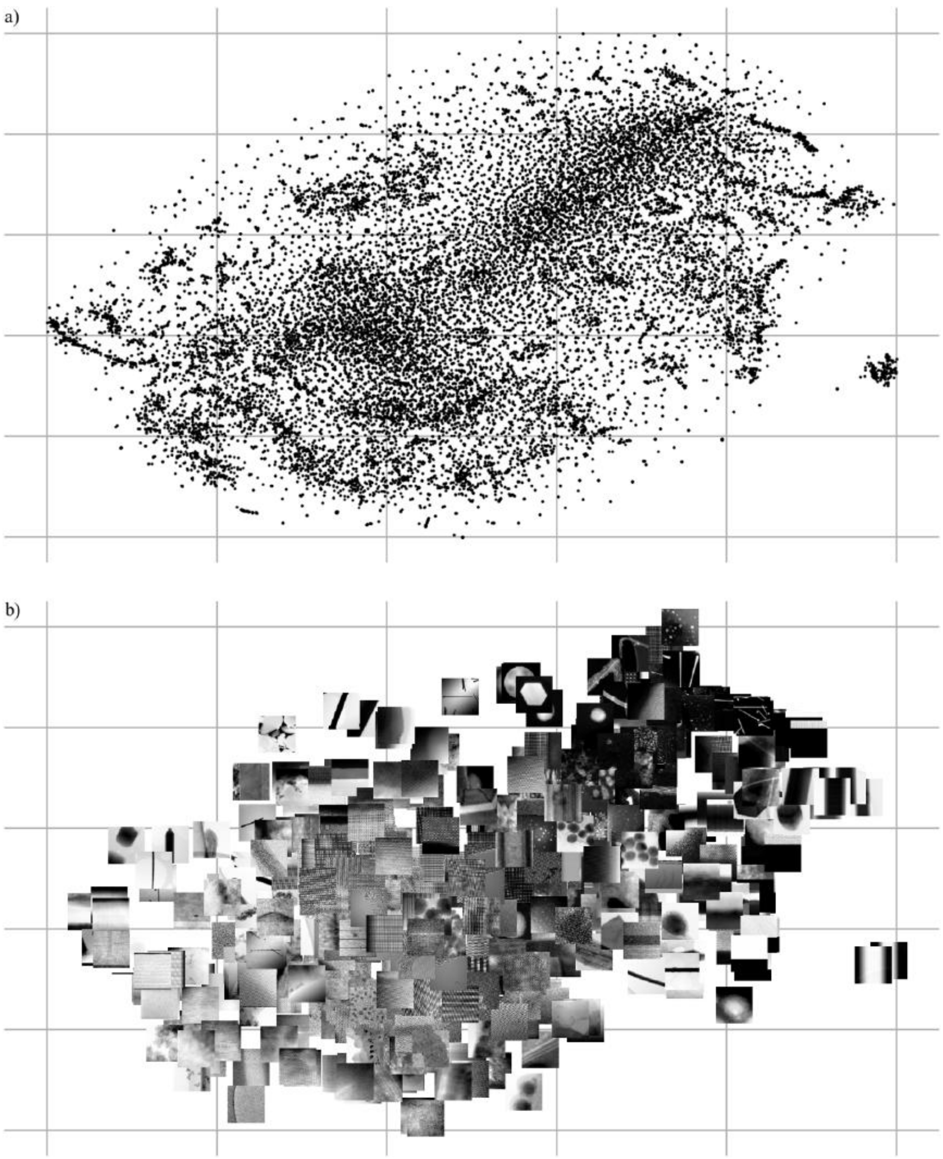


Figure 3. Two-dimensional tSNE visualization of 64-dimensional VAE latent spaces for 19769 STEM images that have been downsampled to 96× 96. The same grid is used to show (a) map points and (b) images at 500 randomly selected points.

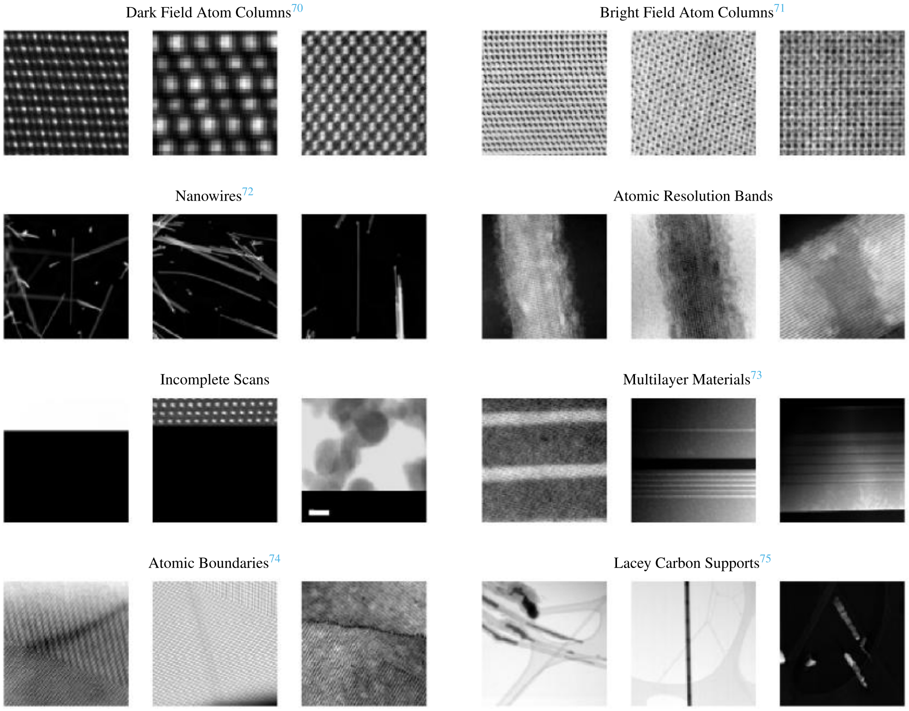
and 2966 test set images. The dataset was made by concatenating contributions from different scientists, so partitioning the dataset before shuffling also partitions scientists.

STEM Crops: 161069 32-bit TIFFs containing 512× 512 non-overlapping regions cropped from STEM Full Images. The dataset is partitioned into 110933 training, 21259 validation, and 28877 test set images. This dataset is biased insofar that larger images were divided into more crops.

STEM 96× 96: A 32-bit NumPy [79, 80] array with shape [19769, 96, 96, 1] containing 19769 STEM Full Images area downsampled to 96× 96 with MATLAB and default antialiasing.

STEM 96× 96 Crops: A 32-bit NumPy array with shape [19769, 96, 96, 1] containing 19769 96× 96 regions cropped from STEM Full Images. Each crop is from a different image.

Variety of STEM 96× 96 images is shown in figure 3 by clustering means and standard deviations of VAE latent spaces in two dimensions by tSNE. Details are in section 2. An interactive visualization that displays



Dark Field Atom Columns[70]

Bright Field Atom Columns[71]

Nanowires[72]

Atomic Resolution Bands

Incomplete Scans

Multilayer Materials[73]

Atomic Boundaries[74]

Lacey Carbon Supports[75]

Table 1. Examples and descriptions of STEM images in our datasets. References put some images into context to make them more tangible to unfamiliar readers.

images when map points are hovered over is also available [8]. This paper is aimed at a general audience so readers may not be familiar with STEM. Subsequently, example images are tabulated with references and descriptions in table 1 to make them more tangible.

4. Transmission electron micrographs

We curated 17266 2048× 2048 high-signal TEM images from University of Warwick electron microscopy dataservers to train ANNs to improve signal-to-noise [4]. However, our dataset was only available upon request. It is now openly available [1]. For convenience, we have also created a new variant containing 96× 96 images that can be used for rapid ANN development. In this section we give details of each TEM dataset, referring to them using their names in our repositories.

TEM Full Images: 17266 32-bit TIFFs containing 2048× 2048 TEM images taken with University of Warwick JEOL 2000, JEOL 2100, JEOL 2100+, and JEOL ARM 200F electron microscope by dozens of scientists working on hundreds of projects. Images were originally saved in DigitalMicrograph DM3 or DM4 files created by Gatan Microscopy Suite [78] software and have been cropped to largest possible squares and area resized to 2048× 2048 with MATLAB and default antialiasing. Images with at least 2500 electron counts per pixel were then linearly transformed to have minimum and maximum values of 0 and 1, respectively. We discarded images with less than 2500 electron counts per pixel as images were curated to train an electron micrograph denoiser [4]. The dataset is partitioned into 11350 training, 2431 validation, and 3486 test set images. The dataset was made by concatenating contributions from different scientists, so each partition contains data collected by a different subset of scientists.

TEM 96× 96: A 32-bit NumPy array with shape [17266, 96, 96, 1] containing 17266 TEM Full Images area downsampled to 96× 96 with MATLAB and default antialiasing. Training, validation, and test set images are concatenated in that order.

Variety of TEM 96× 96 images is shown in figure 4 by clustering means and standard deviations of VAE latent spaces in two dimensions by tSNE. Details are in section 2. An interactive visualization that displays

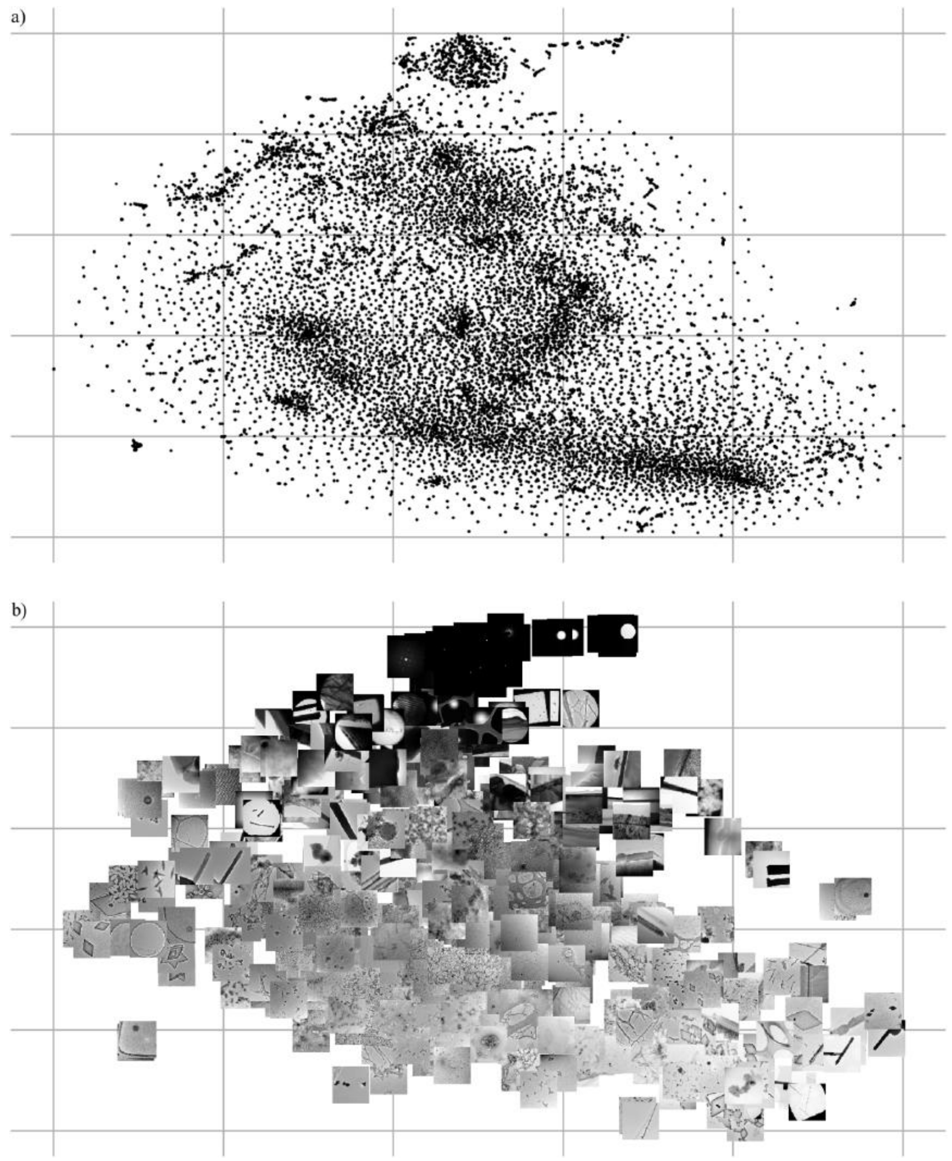
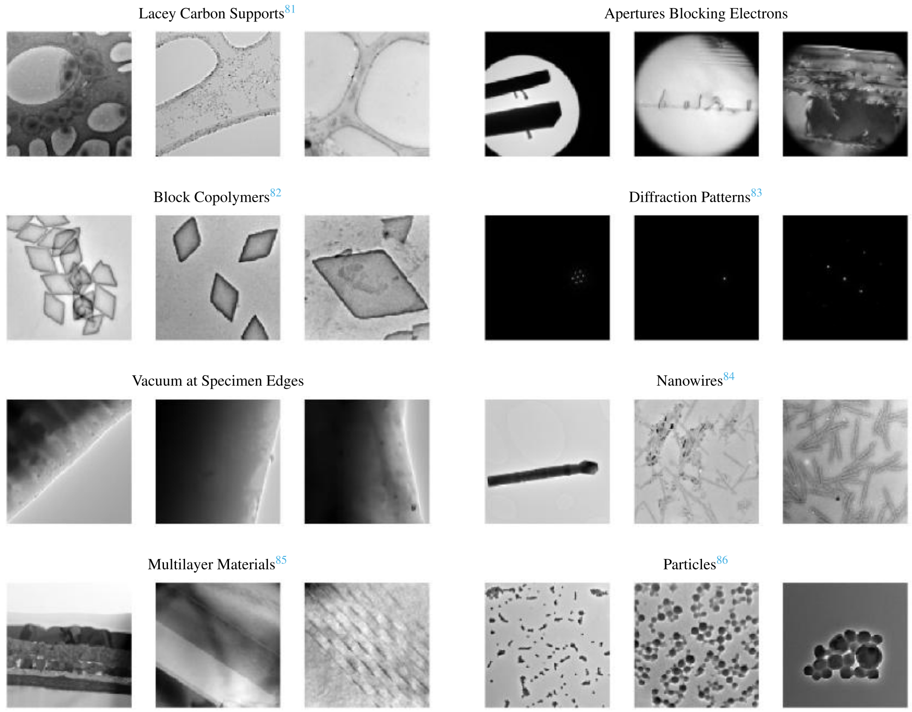


Figure 4. Two-dimensional tSNE visualization of 64-dimensional VAE latent spaces for 17266 TEM images that have been downsampled to 96× 96. The same grid is used to show (a) map points and (b) images at 500 randomly selected points.

images when map points are hovered over is also available [8]. This paper is aimed at a general audience so readers may not be familiar with TEM. Subsequently, example images are tabulated with references and descriptions in table 2 to make them more tangible.

5. Exit wavefunctions

We simulated 98340 TEM exit wavefunctions to train ANNs to reconstruct phases from amplitudes [3]. Half of wavefunction information is undetected by conventional TEM as only the amplitude, and not the phase, of an image is recorded. Wavefunctions were simulated at 512× 512 then centre-cropped to 320× 320 to remove simulation edge artefacts. Wavefunctions have been simulated for real physics where Kirkland potentials [87] for each atom are summed from $n = 3$ terms, and by truncating Kirkland potential summations to $n = 1$ to simulate an alternative universe where atoms have different potentials. Wavefunctions simulated for an alternate universe can be used to test ANN robustness to simulation physics.



Lacey Carbon Supports[81]

Apertures Blocking Electrons

Block Copolymers[82]

Diffraction Patterns[83]

Vacuum at Specimen Edges

Nanowires[84]

Multilayer Materials[85]

Particles[86]

Table 2. Examples and descriptions of TEM images in our datasets. References put some images into context to make them more tangible to unfamiliar readers.

For rapid development, we also downsampled $n = 3$ wavefunctions from 320×320 to 96×96. In this section we give details of each exit wavefunction dataset, referring to them using their names in our repositories.

CIFs: 12789 CIFs downloaded from the Crystallography Open Database [88–93] (COD). The CIFs are for materials published in inorganic chemistry journals. There are 150 New Journal of Chemistry, 1034 American Mineralogist, 1998 Journal of the American Chemical Society and 5457 Inorganic Chemistry CIFs used to simulate training set wavefunctions, 1216 Physics and Chemistry of Materials CIFs used to simulate validation set wavefunctions, and 2927 Chemistry of Materials CIFs used to simulate test set wavefunctions. In addition, the CIFs have been preprocessed to be input to clTEM wavefunction simulations.

URLs: COD Uniform Resource Locators [94] (URLs) that CIFs were downloaded from.

Wavefunctions: 36324 complex 64-bit NumPy files containing 320×320 wavefunctions. The wavefunctions are for a large range of materials and physical hyperparameters. The dataset is partitioned into 24530 training, 3399 validation, and 8395 test set wavefunctions. Metadata Javascript Object Notation [95] (JSON) files link wavefunctions to CIFs and contain some simulation hyperparameters.

Wavefunctions Unseen Training: 1544 64-bit NumPy files containing 320×320 wavefunctions. The wavefunctions are for training set CIFs and are for a large range of materials and physical hyperparameters. Metadata JSONs link wavefunctions to CIFs and contain some simulation hyperparameters.

Wavefunctions Single: 4825 complex 64-bit NumPy files containing 320×320 wavefunctions. The wavefunctions are for a single material, $In_{1.7}K_2Se_8Sn_{2.28}$ [96], and a large range of physical hyperparameters. The dataset is partitioned into 3861 training, and 964 validation set wavefunctions. Metadata JSONs link wavefunctions to CIFs and contain some simulation hyperparameters.

Wavefunctions Restricted: 11870 complex 64-bit NumPy files containing 320×320 wavefunctions. The wavefunctions are for a large range of materials and a small range of physical hyperparameters. The dataset is partitioned into 8002 training, 1105 validation, and 2763 test set wavefunctions. Metadata JSON files link wavefunctions to CIFs and contain some simulation hyperparameters.

Wavefunctions 96× 96: A 32-bit NumPy array with shape [36324, 96, 96, 2] containing 36324 wavefunctions. The wavefunctions were simulated for a large range of materials and physical hyperparameters, and bilinearly downsampled with skimage [47] from 320× 320 to 96× 96 using default antialiasing. In Python [97], Real components are at index [...,0], and imaginary components are at index [...,1]. The dataset can be partitioned in 24530 training, 3399 validation, and 8395 test set wavefunctions, which have been concatenated in that order. To be clear, the training subset is at Python indexes [:24530].

Wavefunctions 96× 96 Restricted: A 32-bit NumPy array with shape [11870, 96, 96, 2] containing 11870 wavefunctions. The wavefunctions were simulated for a large range of materials and a small range of physical hyperparameters, and bilinearly downsampled with skimage from 320× 320 to 96× 96 using default antialiasing. The dataset can be partitioned in 8002 training, 1105 validation, and 2763 test set wavefunctions, which have been concatenated in that order.

Wavefunctions 96× 96 Single: A 32-bit NumPy array with shape [4825, 96, 96, 2] containing 11870 wavefunctions. The wavefunctions were simulated for $In_{1.7}K_2Se_8Sn_{2.28}$ and a large range of physical hyperparameters, and bilinearly downsampled with skimage from 320× 320 to 96× 96 using default antialiasing. The dataset can be partitioned in 3861 training, and 964 validation set wavefunctions, which have been concatenated in that order.

Wavefunctions $n = 1$: 37457 complex 64-bit NumPy files containing 320× 320 wavefunctions. The wavefunctions are for a large range of materials and physical hyperparameters. The dataset is partitioned into 25352 training, 3569 validation, and 8563 test set wavefunctions. These wavefunctions are for an alternate universe where atoms have different potentials.

Wavefunctions $n = 1$ Unseen Training: 1501 64-bit NumPy files containing 320× 320 wavefunctions. The wavefunctions are for training set CIFs and are for a large range of materials and physical hyperparameters. Metadata JSONs link wavefunctions to CIFs and contain some simulation hyperparameters. These wavefunctions are for an alternate universe where atoms have different potentials.

Wavefunctions $n = 1$ Single: 4819 complex 64-bit NumPy files containing 320× 320 wavefunctions. The wavefunctions are for a single material, $In_{1.7}K_2Se_8Sn_{2.28}$, and a large range of physical hyperparameters. The dataset is partitioned into 3856 training, and 963 validation set wavefunctions. Metadata JSONs link wavefunctions to CIFs and contain some simulation hyperparameters. These wavefunctions are for an alternate universe where atoms have different potentials.

Experimental Focal Series: 1000 experimental focal series. Each series consists of 14 32-bit 512× 512 TEM images, area downsampled from 4096× 4096 with MATLAB and default antialiasing. The images are in TIFF [98] format. All series were created with a common, quadratically increasing [99] defocus series. However, spatial scales vary and would need to be fitted as part of wavefunction reconstruction.

In detail, exit wavefunctions for a large range of physical hyperparameters were simulated with clTEM [100, 101] for acceleration voltages in $\{80, 200, 300\}$ kV, material depths uniformly distributed in $[5, 100)$ nm, material widths in $[5, 10)$ nm, and crystallographic zone axes (h, k, l) $h, k, l \in \{0, 1, 2\}$. Materials were padded on all sides with vacuum 0.8 nm wide and 0.3 nm deep to reduce simulation artefacts. Finally, crystal tilts were perturbed by zero-centred Gaussian random variates with 0.1° standard deviations. We used default values for other clTEM hyperparameters. Simulations for a small range of physical hyperparameters used lower upper bounds that reduced simulation hyperparameter ranges by factors close to 1/4. All wavefunctions are linearly transformed to have a mean amplitude of 1.

All wavefunctions show atom columns, so tSNE visualizations are provided in supplementary information to conserve space. The visualizations are for Wavefunctions 96× 96, Wavefunctions 96× 96 Restricted and Wavefunctions 96× 96 Single.

6. Discussion

The best dataset variant varies for different applications. Full-sized datasets can always be used as other dataset variants are derived from them. However, loading and processing full-sized examples may bottleneck training, and it is often unnecessary. Instead, smaller 512× 512 crops, which can be loaded more quickly the full-sized images, can often be used to train ANNs to be applied convolutionally [102] to or tiled across [4] full-sized inputs. In addition, our 96× 96 datasets can be used for rapid initial development before scaling up to full-sized datasets, similar to how ANNs might be trained with CIFAR-10 before scaling up to ImageNet. However, subtle application- and dataset-specific considerations may also influence the best dataset choice. For example, an ANN trained with downsampled 96× 96 inputs may not generalize to 96× 96 crops from full-sized inputs as downsampling may introduce artifacts [103] and change noise or other data characteristics.

In practice, electron microscopists image most STEM and TEM signals at several times their Nyquist rates [77]. This eases visual inspection, decreases sub-Nyquist aliasing [104], improves display on computer monitors, and is easier than carefully tuning sampling rates to capture the minimum data needed to resolve signals. High sampling may also reveal additional high-frequency information when images are inspected after an experiment. However, this complicates ANN development as it means that information per pixel is often higher in downsampled images. For example, partial scans across STEM images that have been dowsampled to 96× 96 require higher coverages than scans across 96× 96 crops for ANNs to learn to complete images with equal performance [5]. It also complicates the comparison of different approaches to compressed sensing. For example, we suggested that sampling 512× 512 crops at a regular grid of probing locations outperforms sampling along spiral paths as a subsampling grid can still access most information [5].

Test set performance should be calculated for a standardized dataset partition to ease comparison with other methods. Nevertheless, training and validation partitions can be varied to investigate validation variance for partitions with different characteristics. Default training and validation sets for STEM and TEM datasets contain contributions from different scientists that have been concatenated or numbered in order, so new validation partitions can be selected by concatenating training and validation partitions and moving the window used to select the validation set. Similarly, exit wavefunctions were simulated with CIFs from different journals that were concatenated or numbered sequentially. There is leakage [105, 106] between training, validation and test sets due to overlap between materials published in different journals and between different scientists' work. However, further leakage can be minimized by selecting dataset partitions before any shuffling and, for wavefunctions, by ensuring that simulations for each journal are not split between partitions.

Experimental STEM and TEM image quality is variable. Images were taken by scientists with all levels of experience and TEM images were taken on multiple microscopes. This means that our datasets contain images that might be omitted from other datasets. For example, the tSNE visualization for STEM in figure 3 includes incomplete scans, ~ 50 blank images, and images that only contain noise. Similarly, the tSNE visualization for TEM in figure 4 revealed some images where apertures block electrons, and that there are small number of unprocessed standard diffraction and convergent beam electron diffraction [107] patterns. Although these conventionally low-quality images would not normally be published, they are important to ensure that ANNs are robust for live applications. In addition, inclusion of conventionally low-quality images may enable identification of this type of data. We encourage readers to try our interactive tSNE visualizations [8] for detailed inspection of our datasets.

In this paper, we present tSNE visualizations of VAE latent spaces to show image variety. However, our VAEs can be directly applied to a wide range of additional applications. For example, successful tSNE clustering of latent spaces suggests that VAEs could be used to create a hash table [108, 109] for an electron micrograph search engine. VAEs can also be applied to semantic manipulation [110], and clustering in tSNE visualizations may enable subsets of latent space that generate interesting subsets of data distributions to be identified. Other applications include using clusters in tSNE visualizations to label data for supervised learning, data compression, and anomaly detection [111, 112]. To encourage further development, we have made our source code and pretrained VAEs openly available [8].

7. Conclusion

We have presented details of and visualizations for large new electron microscopy datasets that are openly available from our new repositories. Datasets have been carefully partitioned into training, validation, and test sets for machine learning. Further, we provide variants containing 512× 512 crops to reduce data loading times, and examples downsampled to 96× 96 for rapid development. To improve dataset visualization with VAEs, we introduce encoding normalization and regularization, and add an image gradient loss. In addition, we propose extending tSNE to account for encoded standard deviations. Source code, pretrained VAEs, precompiled tSNE binaries, and interactive dataset visualizations are provided in supplementary repositories to help users become familiar with our datasets and visualizations. By making our datasets available, we aim to encourage standardization of performance benchmarks in electron microscopy and increase participation of the wider computer science community in electron microscopy research.

8. Supplementary Information

Ten additional tSNE visualizations are provided as supplementary information. They are for:

- Extracting 50 principal components by probabilistic PCA for the STEM 96× 96, STEM 96× 96 Crops, TEM 96× 96, Wavefunctions 96× 96, Wavefunctions 96× 96 Restricted and Wavefunctions 96× 96 Single datasets. PCA is a quick and effective method to extract features. As a result, we think that visualizations for PCA are interesting benchmarks.
- VAE latent spaces with σ propagation for the STEM 96× 96 Crops dataset. Crops show smaller features than downsampled images.
- VAE latent spaces without σ propagation for the STEM 96× 96, STEM 96× 96 Crops and TEM 96× 96 datasets. They are comparable to visualizations created with σ propagation.

Interactive versions of tSNE visualizations that display data when map points are hovered over are available [8] for every figure. In addition, we propose an algorithm to increase whitespace utilization in tSNE visualizations by uniformly separating points, and show that our VAEs can be used as the basis of image search engines. Supplementary information is openly available at https://doi.org/10.5281/zenodo.3899740 and stacks.iop.org/MLST/1/045003/mmedia.

9. Data Availability

The data that support the findings of this study are openly available at https://doi.org/10.5281/zenodo.3834197. For additional information contact the corresponding author (J.M.E.).

Acknowledgement

Funding: J.M.E. acknowledges EPSRC EP/N035437/1 and EPSRC Studentship 1917382.

Competing Interests

The author declares no competing interests.

ORCID iD

Jeffrey M Ede https://orcid.org/0000-0002-9358-5364

References

[1] Ede J M 2020 Electron microscopy datasets (Available online at: https://github.com/Jeffrey-Ede/datasets/wiki)

[2] FEI C 2010 An introduction to electron microscopy (Available online at: https://www.fei.com/documents/introduction-to-microscopy-document)

[3] Ede J M, Peters J J P, Sloan J and Beanland R 2020 Exit wavefunction reconstruction from single transmission electron micrographs with deep learning (arXiv:2001.10938)

[4] Ede J M and Beanland R 2019 Improving electron micrograph signal-to-noise with an atrous convolutional encoder-decoder *Ultramicroscopy* **202** 18–25

[5] Ede J M and Beanland R 2020 Partial scanning transmission electron microscopy with deep learning *Sci. Rep.* **10** 8332

[6] Ede J M and Beanland R 2020 Adaptive learning rate clipping stabilizes learning *Mach. Learn. Sci. Technol.* **1** 015011

[7] Ede J M 2019 Deep learning supersampled scanning transmission electron microscopy (arXiv:1910.10467)

[8] Ede J M 2020 Visualization of electron microscopy datasets with deep learning (Available online at: https://github.com/Jeffrey-Ede/datasets)

[9] Schmidt J, Marques M R, Botti S and Marques M A 2019 Recent advances and applications of machine learning in solid-state materials science *npj Comput. Mater.* **5** 1–36

[10] von Lilienfeld O A 2020 Introducing Machine Learning: Science and Technology *Mach. Learn. Sci. Technol.* **1** 010201

[11] Belianinov A *et al* 2015 Big data and deep data in scanning and electron microscopies: deriving functionality from multidimensional data sets *Adv. Struct. Chem. Imaging* **1** 1–25

[12] Hornik K, Stinchcombe M and White H 1989 Multilayer feedforward networks are universal approximators *Neural Netw.* **2** 359–66

[13] Lin H W, Tegmark M and Rolnick D 2017 Why does deep and cheap learning work so well? *J. Stat. Phys.* **168** 1223–47

[14] Raschka S 2018 Model evaluation, model selection and algorithm selection in machine learning (arXiv:1811.12808)

[15] Roh Y, Heo G and Whang S E 2019 A survey on data collection for machine learning: A big data-AI integration perspective *IEEE Trans. Knowl. Data Eng.* 10.1109/TKDE.2019.2946162

[16] Hall S R, Allen F H and Brown I D 1991 The crystallographic information file (CIF): A new standard archive file for crystallography *Acta Crystallogr. Sect. A: Foundations Crystallogr.* **47** 655–85

[17] Baker M 2016 Reproducibility Crisis? *Nature* **533** 353–66

[18] Hutson M 2018 Artificial intelligence faces reproducibility crisis *Science* **359** 725–6

[19] Iudin A, Korir P K, Salavert-Torres J, Kleywegt G J and Patwardhan A 2016 EMPIAR: A public archive for raw electron microscopy image data *Nat. Methods* **13** 387

[20] Hey T, Butler K, Jackson S and Thiyagalingam J 2020 Machine learning and big scientific data *Philosophical Trans. of the Royal Society A* **378** 20190054

[21] Krizhevsky A, Nair V and Hinton G 2014 The CIFAR-10 dataset (Available online at: http://www:cs:toronto:edu/ kriz/cifar:html)

[22] Krizhevsky A and Hinton G 2009 Learning multiple layers of features from tiny images. Tech. Rep., Citeseer

[23] LeCun Y, Cortes C and Burges C 2010 MNIST handwritten digit database. AT&T Labs (Available online at: http://yann.lecun.com/exdb/mnist)

[24] Russakovsky O *et al* 2015 ImageNet large scale visual recognition challenge *Int. J. Comput. Vis.* **115** 211–52

[25] Tenenbaum J B, De Silva V and Langford J C 2000 A global geometric framework for nonlinear dimensionality reduction *Science* **290** 2319–23

[26] Roweis S T and Saul L K 2000 Nonlinear dimensionality reduction by locally linear embedding *Science* **290** 2323–6

[27] Zhang Z and Wang J 2007 MLLE: Modified locally linear embedding using multiple weights *Advances in Neural Information Processing Systems 19: Proc. of the 2006 Conf.* pp 1593–600

[28] Donoho D L and Grimes C 2003 Hessian eigenmaps: locally linear embedding techniques for high-dimensional data *Proc. Natl Acad. Sci.* **100** 5591–6

[29] Belkin M and Niyogi P 2003 Laplacian eigenmaps for dimensionality reduction and data representation *Neural Comput.* **15** 1373–96

[30] Zhang Z and Zha H 2004 Principal manifolds and nonlinear dimensionality reduction via tangent space alignment *SIAM J. Sci. Comput.* **26** 313–38

[31] Buja A *et al* 2008 Data visualization with multidimensional scaling *J. Comput. Graph. Stat.* **17** 444–72

[32] Van Der Maaten L 2014 Accelerating t-SNE using tree-based algorithms *J. Mach. Learn. Res.* **15** 3221–45

[33] Maaten L v d and Hinton G 2008 Visualizing data using t-SNE *J. Mach. Learn. Res.* **9** 2579–605

[34] Wattenberg M, Viégas F and Johnson I 2016 How to use t-SNE effectively *Distill* **1** e2

[35] Ruder S 2016 An overview of gradient descent optimization algorithms (arXiv:1609.04747)

[36] Schubert E and Gertz M 2017 Intrinsic t-stochastic neighbor embedding for visualization and outlier detection *Int. Conf. on Similarity Search and Applications* (Berlin: Springer) pp 188–203

[37] Halko N, Martinsson P-G and Tropp J A 2011 Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions *SIAM Rev.* **53** 217–88

[38] Martinsson P-G, Rokhlin V Tygert M 2011 A randomized algorithm for the decomposition of matrices *Appl. Comput. Harmon. Anal.* **30** 47–68

[39] Wall M E, Rechtsteiner A Rocha L M 2003 Singular value decomposition and principal component analysis *A Practical Approach to Microarray Data Analysis* (Berlin: Springer) pp 91–109

[40] Jolliffe I T and Cadima J 2016 Principal component analysis: A review and recent developments *Philosophical Trans. of the Royal Society A: Mathematical, Physical and Engineering Sciences* **374** 20150202

[41] Dalal N and Triggs B 2005 Histograms of oriented gradients for human detection *2005 IEEE Computer Conf. on Computer Vision and Pattern Recognition (CVPR'05)* IEEE vol 1 pp 886–93

[42] Bay H, Ess A, Tuytelaars T Van Gool L 2008 Speeded-Up robust features (SURF) *Comput. Vis. Image Underst.* **110** 346–59

[43] Ojala T, Pietikainen M and Maenpaa T 2002 Multiresolution gray-scale and rotation invariant texture classification with local binary pattern *IEEE Trans. Pattern Anal. Mach. Intell.* **24** 971–87

[44] Mallat S G 1989 A Theory for multiresolution signal decomposition: The wavelet representation *IEEE Transactions on Pattern Analysis Mach. Intell.* **11** 674–93

[45] Latif A *et al* 2019 Content-based image retrieval and feature extraction: a comprehensive review *Math. Probl. Eng.* **2019** 10.1155/2019/9658350

[46] Minka T P 2001 Automatic choice of dimensionality for PCA *Adv Neural Inf Process Syst.* **13** 598–604

[47] Van der Walt S *et al* 2014 scikit-image: image processing in python *PeerJ* **2** e453

[48] Tschannen M, Bachem O and Lucic M 2018 Recent advances in autoencoder-based representation learning (arXiv:1812.05069)

[49] Kramer M A 1991 Nonlinear principal component analysis using autoassociative neural networks *AIChE J.* **37** 233–43

[50] Marcelino P 2018 Transfer learning from pre-trained models Towards data science (Available online at: https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751)

[51] Ede J M 2018 Kernels, MLPs and autoencoders (Available online at: https://github.com/Jeffrey-Ede/Denoising-Kernels-MLPs-Autoencoders)

[52] Ede J M 2018 Autoencoders, kernels and multilayer perceptrons for electron micrograph restoration and compression (arXiv:1808.09916)

[53] Kingma D P and Welling M 2014 Auto-encoding variational Bayes (arXiv:1312.6114)

[54] Kingma D P and Welling M 2019 An introduction to variational autoencoders (arXiv:1906.02691)

[55] McCann M T, Jin K H and Unser M 2017 Convolutional neural networks for inverse problems in imaging: A review *IEEE Signal Process. Mag.* **34** 85–95

[56] Krizhevsky A, Sutskever I and Hinton G E 2012 ImageNet classification with deep convolutional neural networks *Adv Neural Inf Process Syst.* **25** 1097–105

[57] Ioffe S and Szegedy C 2015 Batch normalization: accelerating deep network training by reducing internal covariate shift (arXiv:1502.03167)

[58] Nair V and Hinton G E 2010 Rectified linear units improve restricted Boltzmann machines *Proc. of the 27th Int. Conf. on Machine Learning (ICML-10)* pp 807–14

[59] Higgins I *et al* 2017 beta-VAE: learning basic visual concepts with a constrained variational framework *Int. Conf. on Learning Representations* vol **2** p 6

[60] Hu Q and Greene C S 2019 Parameter tuning is a key part of dimensionality reduction via deep variational autoencoders for single cell RNA transcriptomics *Symp. on Biocomputing. Symp. on Biocomputing* NIH Public Access vol 24 p 362

[61] McIlhagga W 2018 Estimates of edge detection filters in human vision *Vis. Res.* **153** 30–6

[62] Larsen A B L, Sønderby S K, Larochelle H and Winther O 2015 Autoencoding beyond pixels using a learned similarity metric (arXiv:1512.09300)

[63] Vairalkar M K and Nimbhorkar S 2012 Edge detection of images using Sobel operator *Int. Journal of Emerging Technology and Advanced Engineering* **2** 291–3

[64] Kingma D P and Ba J 2014 ADAM: A method for stochastic optimization (arXiv:1412.6980)

[65] Zou D, Cao Y, Zhou D and Gu Q 2018 Stochastic gradient descent optimizes over-parameterized deep ReLU networks (arXiv:1811.08888)

[66] Ge R, Kakade S M, Kidambi R and Netrapalli P 2019 The step decay schedule: a near optimal, geometrically decaying learning rate procedure (arXiv:1904.12838)

[67] Chen J and Kyrillidis A 2019 Decaying momentum helps neural network training (arXiv:1910.04952)

[68] Oskolkov N 2019 How to tune hyperparameters of tSNE Towards Data Science (Available online at: https://towardsdatascience.com/how-to-tune-hyperparameters-of-tsne-7c0596a18868)

[69] Pedregosa F *et al* 2011 scikit-learn: machine learning in python *J. Mach. Learn. Res.* **12** 2825–30

[70] Van den Bos K H *et al* 2016 Unscrambling mixed elements using high angle annular dark field scanning transmission electron microscopy *Phys. Rev. Lett.* **116** 246101

[71] Zhou D *et al* 2016 Sample tilt effects on atom column position determination in ABF-STEM imaging *Ultramicroscopy* **160** 110–17

[72] Bu L *et al* 2016 Surface engineering of hierarchical platinum-cobalt nanowires for efficient electrocatalysis *Nat. Commun.* **7** 1–10

[73] Monclús M *et al* 2018 Effect of layer thickness on the mechanical behaviour of oxidation-strengthened Zr/Nb nanoscale multilayers *J. Mater. Sci.* **53** 5860–78

[74] Pyrz W D *et al* 2010 Atomic-Level imaging of Mo-V-O complex oxide phase intergrowth, grain boundaries and defects using HAADF-STEM *Proc. Natl Acad. Sci.* **107** 6152–7

[75] McGilvery C M, Goode A E, Shaffer M S and McComb D W 2012 Contamination of holey/lacey carbon films in STEM *Micron* **43** 450–5

[76] Seki T, Ikuhara Y and Shibata N 2018 Theoretical framework of statistical noise in scanning transmission electron microscopy *Ultramicroscopy* **193** 118–25

[77] Landau H 1967 Sampling, data transmission and the Nyquist rate *Proc. of the IEEE* **55** 1701–6

[78] Gatan microscopy suite 2019 (Available online at: www.gatan.com/products/tem-analysis/gatan-microscopy-suite-software)

[79] NPY format 2019 (Available online at: https://docs.scipy.org/doc/numpy/reference/generated/numpy.lib.format.html)

[80] Kern R 2007 NEP 1—a simple file format for NumPy arrays (Available online at: https://numpy.org/neps/nep-0001-npy-format.html)

[81] Karlsson G 2001 Thickness measurements of lacey carbon films *J. Microsc.* **203** 326–8

[82] Inam M *et al* 2017 1D vs. 2D Shape selectivity in the crystallization-driven self-assembly of polylactide block copolymers *Chem. Sci.* **8** 4223–30

[83] Bendersky L A and Gayle F W 2001 Electron diffraction using transmission electron microscopy *J. Res. Natl Inst. Stand. Technol.* **106** 997

[84] Wu Y, Messer B and Yang P 2001 Superconducting MgB_2 nanowires *Adv. Mater.* **13** 1487–9

[85] Pang B *et al* 2017 The microstructural characterization of multiferroic $LaFeO_3$-$YMnO_3$ multilayers grown on (001)- and (111)-$SrTiO_3$ substrates by transmission electron microscopy *Materials* **10** 839

[86] Dong Z *et al* 2016 Individual particles of cryoconite deposited on the mountain glaciers of the Tibetan Plateau: Insights into chemical composition and sources *Atmos. Environ.* **138** 114–24

[87] Kirkland E J 2010 *Advanced computing in electron microscopy* (Berlin: Springer Science & Business Media)

[88] Quirós M, Gražulis S, Girdzijauskaitė S, Merkys A and Vaitkus A 2018 Using SMILES strings for the description of chemical connectivity in the Crystallography Open Database *J. Cheminformatics* **10** 1–17

[89] Merkys A *et al* 2016 COD:: CIF::Parser: An error-correcting CIF parser for the Perl language *J. Appl. Crystallogr.* **49** 292–301

[90] Gražulis S, Merkys A, Vaitkus A and Okulič-Kazarinas M 2015 Computing stoichiometric molecular composition from crystal structures *J. Appl. Crystallogr.* **48** 85–91

[91] Gražulis S *et al* 2012 Crystallography Open Database (COD): An open-access collection of crystal structures and platform for world-wide collaboration *Nucleic Acids Res.* **40** D420–D427

[92] Gražulis S *et al* 2009 Crystallography Open Database – an open-access collection of crystal structures *J. Appl. Crystallogr.* **42** 726–9

[93] Downs R T and Hall-Wallace M 2003 The American Mineralogist crystal structure database *Am. Mineral.* **88** 247–50

[94] Berners-Lee T, Masinter L and McCahill M 1994 RFC1738: Uniform Resource Locators (URL) *RFC*

[95] ISO/IEC JTC 1/SC 22 2017 International standard ISO/IEC21778: information technology - the JSON data interchange syntax (Available online at: https://www.iso.org/standard/71616.html)

[96] Hwang S-J, Iyer R G, Trikalitis P N, Ogden A G and Kanatzidis M G 2004 Cooling of melts: kinetic stabilization and polymorphic transitions in the $KInSnSe_4$ System *Inorg. Chem.* **43** 2237–9

[97] Python software foundation 2020 Python 3.6 (Available online at: www.python.org)

[98] Adobe developers association 1992 *et al* TIFF Revision 6.0. (Available online at: www.adobe.io/content/dam/udp/en/open/standards/tiff/TIFF6.pdf)

[99] Haigh S, Jiang B, Alloyeau D, Kisielowski C and Kirkland A 2013 Recording low and high spatial frequencies in exit wave reconstructions *Ultramicroscopy* **133** 26–34

[100] Peters J J P and Dyson M A 2019 clTEM (Available online at: https://github.com/JJPPeters/clTEM)

[101] Dyson M A 2014 *Advances in Computational Methods for Transmission Electron Microscopy Simulation and Image Processing* Ph.D. thesis University of Warwick

[102] Zhu J-Y, Park T, Isola P and Efros A A 2017 Unpaired image-to-image translation using cycle-consistent adversarial networks *Proc. of the IEEE Int. Conf. on Computer Vision* pp 2223–32

[103] MicroImages 2010 Resampling methods. technical Guide (Available online at: https://www.microimages.com/documentation/TechGuides/77resampling.pdf)

[104] Amidror I 2015 Sub-Nyquist artefacts and sampling Moiré effects *Royal Soc. Open Sci.* **2** 140550

[105] Open data science 2019 How to fix data leakage - your model's greatest enemy. towards data science (Available online at: https://medium.com/@ODSC/how-to-fix-data-leakage-your-models-greatest-enemy-e34fa26abac5)

[106] Bussola N, Marcolini A, Maggio V, Jurman G and Furlanello C 2019 Not again! Data leakage in digital pathology (arXiv:1909.06539)

[107] Tanaka M 1994 Convergent-beam electron diffraction *Acta Crystallogr. Sect. A: Foundations Crystallogr.* **50** 261–86

[108] Patterson N and Wang Y Semantic hashing with variational autoencoders https://pdfs.semanticscholar.org/f2c3/3951f347b5e0f7ac4946f0672fdb4ca5394b.pdf

[109] Jin G, Zhang Y and Lu K 2019 Deep hashing based on VAE-GAN for efficient similarity retrieval *Chin. J. Electron.* **28** 1191–7

[110] Klys J, Snell J Zemel R 2018 Learning latent subspaces in variational autoencoders *Adv Neural Inf Process Syst.* **31** 6444–54

[111] Yao R, Liu C, Zhang L and Peng P 2019 Unsupervised anomaly detection using variational auto-encoder based feature extraction *2019 IEEE Int. Conf. on Prognostics and Health Management (ICPHM)* IEEE pp 1–7

[112] Xu H *et al* 2018 Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications *Proc. of the 2018 World Wide Conf.* pp 187–96

Supplementary Information: Warwick Electron Microscopy Datasets

Jeffrey M. Ede[1,*]

[1]University of Warwick, Department of Physics, Coventry, CV4 7AL, UK
[*]j.m.ede@warwick.ac.uk

S1 Additional Visualizations

Figure numbers for a variety of two-dimensional tSNE visualisations are tabulated in table S1 to ease comparison. Visualizations are for the first 50 principal components extracted by a scikit-learn[1] implementation of probabilistic PCA, means encoded in 64-dimensional VAE latent spaces without modified tSNE losses to account for standard deviations, and means encoded in 64-dimensional VAE latent spaces with modified tSNE losses to account for standard deviations. Interactive versions of tSNE visualizations that display data when map points are hovered over are also available for every visualization[2]. In addition, our source code, graph points and datasets are openly available.

Dataset	PCA	VAE $\{\mu\}$	VAE $\{\mu, \sigma\}$
STEM 96×96	S1	S7	3 (main article)
STEM 96×96 Crops	S2	S8	S10
TEM 96×96	S3	S9	4 (main article)
Wavefunctions 96×96	S4	-	-
Wavefunctions 96×96 Restricted	S5	-	-
Wavefunctions 96×96 Single	S6	-	-

Table S1. To ease comparison, we have tabulated figure numbers for tSNE visualizations. Visualizations are for principal components, VAE latent space means, and VAE latent space means weighted by standard deviations..

Visualization of complex exit wavefunctions is complicated by the display of their real and imaginary components. However, real and imaginary components are related[3], and can be visualized in the same image by displaying them in different colour channels. For example, we show real and imaginary components in red and blue colour channels, respectively, in figures S4-S6. Note that a couple of extreme points are cropped from some of the tSNE visualizations of principal components in figures S1-S6. However, this only affected ~0.01% of points and therefore does not have a substantial effect on visualizations. In contrast, tSNE visualizations of VAE latent spaces did not have extreme points.

S2 Uniformly Separated tSNE

Limitedly, most tSNE visualizations do not fully utilize whitespace. This is problematic as space is often limited in journals, websites and other media. As a result, we propose algorithm 1 to uniformly separate map points. This increases whitespace utilization while keeping clustered points together. Example applications are shown in figures S11-S13, where images nearest points on a regular grid are shown at grid points. Uniformly separating map points removes information about pairwise distances encoded in the tSNE distributions. However, distances and cluster sizes in tSNE visualizations are not overly meaningful[4]. Overall, we think that uniformly separated tSNE is an interesting option that could be improved by further development. To this end, our source code and graph points for uniformly separated tSNE visualizations are openly available[2].

S3 Image Search Engines

Our VAEs can be used as the basis of image search engines. To find similar images, we compute Euclidean distances between means encoded for search inputs and images in the STEM or TEM datasets, then select images at lowest distances. Examples of top-5 search results for various input images are shown in figure S14 and figure S15 for TEM and STEM, respectively. Search results are most accurate for common images and are less accurate for unusual images. The main difference between the performance of our search engines and Google, Bing or other commercial image search engines is the result of commercial ANNs being trained with over 100× more training data, 3500× more computational resources and larger images c.f. Xception[5]

However, the performance of our search engines is okay and our VAEs could easily be scaled up. Our source code, pretrained models and VAE encodings for each dataset are openly available[2].

Algorithm 1 Two-dimensional Bayesian inverse transformed tSNE. We default to a $h = w = 25$ grid.

Initialize N two-dimensional tSNE map points, $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$, where $\mathbf{x}_i = \{x_{i1}, x_{i2}\}$.
Linearly transform dimensions to have values in $[0, 1]$,

$$x_{ij} \leftarrow \frac{x_{ij} - \min_i(x_{ij})}{\max_i(x_{ij}) - \min_i(x_{ij})} \,. \tag{1}$$

Divide points into an evenly spaced grid with $h \times w$ cells.
Compute number of points in each cell, n_{ab}, $a \in [1, h]$, $b \in [1, w]$.
Cumulative numbers of points using the recurrent relations,

$$c_a = c_{a-1} + \sum_{b=1}^{w} n_{ab} \,, \tag{2}$$

$$c_{b|a} = c_{b-1|a} + n_{ab} \,, \tag{3}$$

where $c_0 = c_{0|a} = 0$.
Estimate Bayesian conditional cumulative distribution functions,

$$C_a = c_a / c_h \,, \tag{4}$$

$$C_{b|a} = c_{b|a} / c_a \,. \tag{5}$$

Map grid points, $\{(a - 0.5)/h, (b - 0.5)/w\}$, to distribution points, $\{(u - 0.5)/h, (v - 0.5)/w\}$, where u and v are minimum values that satisfy

$$(a - 0.5)/h \leq C_u \,, \tag{6}$$

$$(b - 0.5)/w \leq C_{v|u} \,. \tag{7}$$

Interpolate uniformly separated grid positions, $\mathbf{Y}$, for $\mathbf{X}$ based on pairs of grid and distribution points. We use Clough-Tocher cubic Bezier interpolation[6].

References

1. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).

2. Ede, J. M. Visualization of Electron Microscopy Datasets with Deep Learning. Online: https://github.com/Jeffrey-Ede/datasets, DOI: 10.5281/zenodo.3834197 (2020).

3. Ede, J. M., Peters, J. J. P., Sloan, J. & Beanland, R. Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning. *arXiv preprint arXiv:2001.10938* (2020).

4. Wattenberg, M., Viégas, F. & Johnson, I. How to Use t-SNE Effectively. *Distill* **1**, e2 (2016).

5. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1251–1258 (2017).

6. Alfeld, P. A Trivariate Clough—Tocher Scheme for Tetrahedral Data. *Comput. Aided Geom. Des.* **1**, 169–181 (1984).

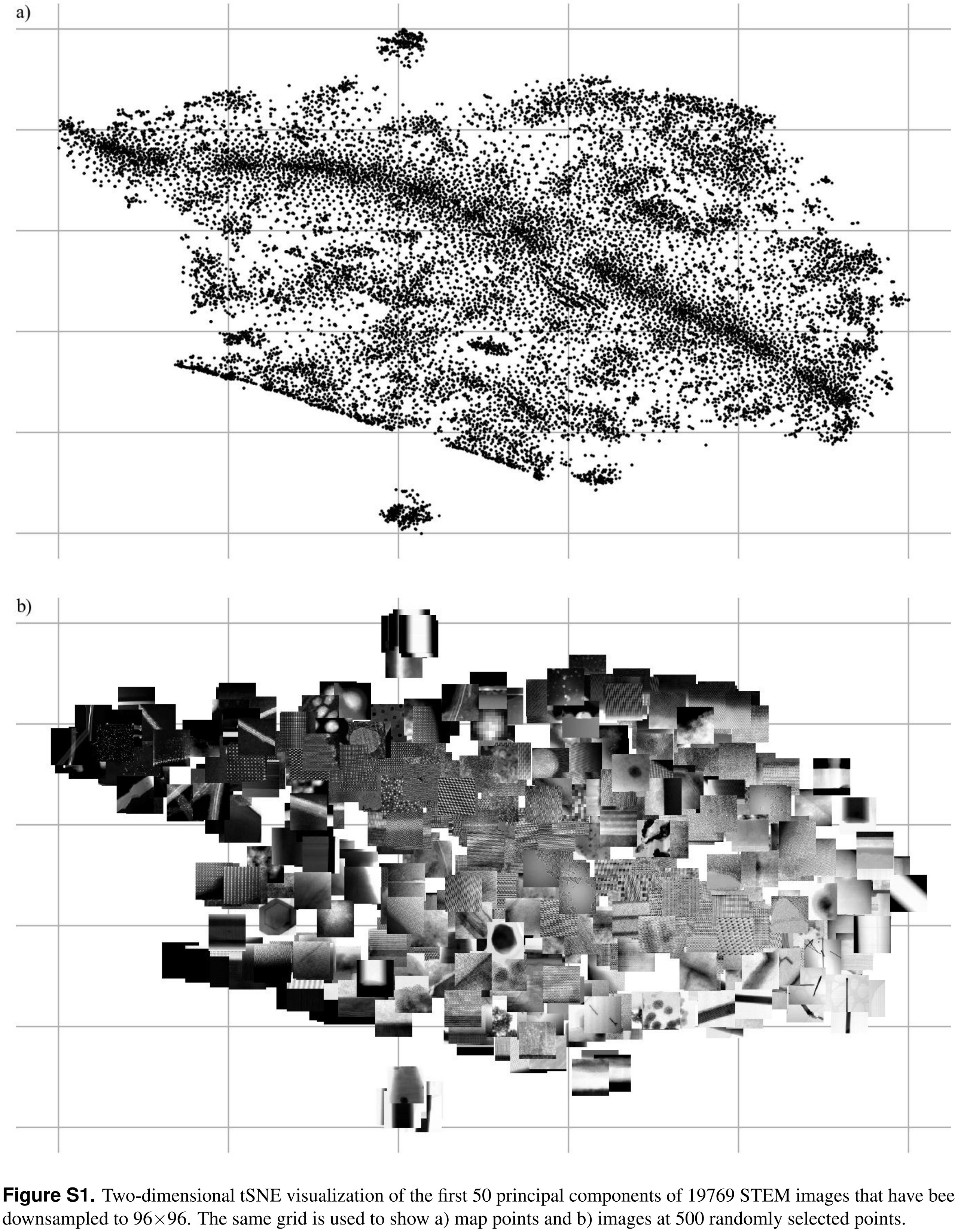


Figure S1. Two-dimensional tSNE visualization of the first 50 principal components of 19769 STEM images that have been downsampled to 96×96. The same grid is used to show a) map points and b) images at 500 randomly selected points.

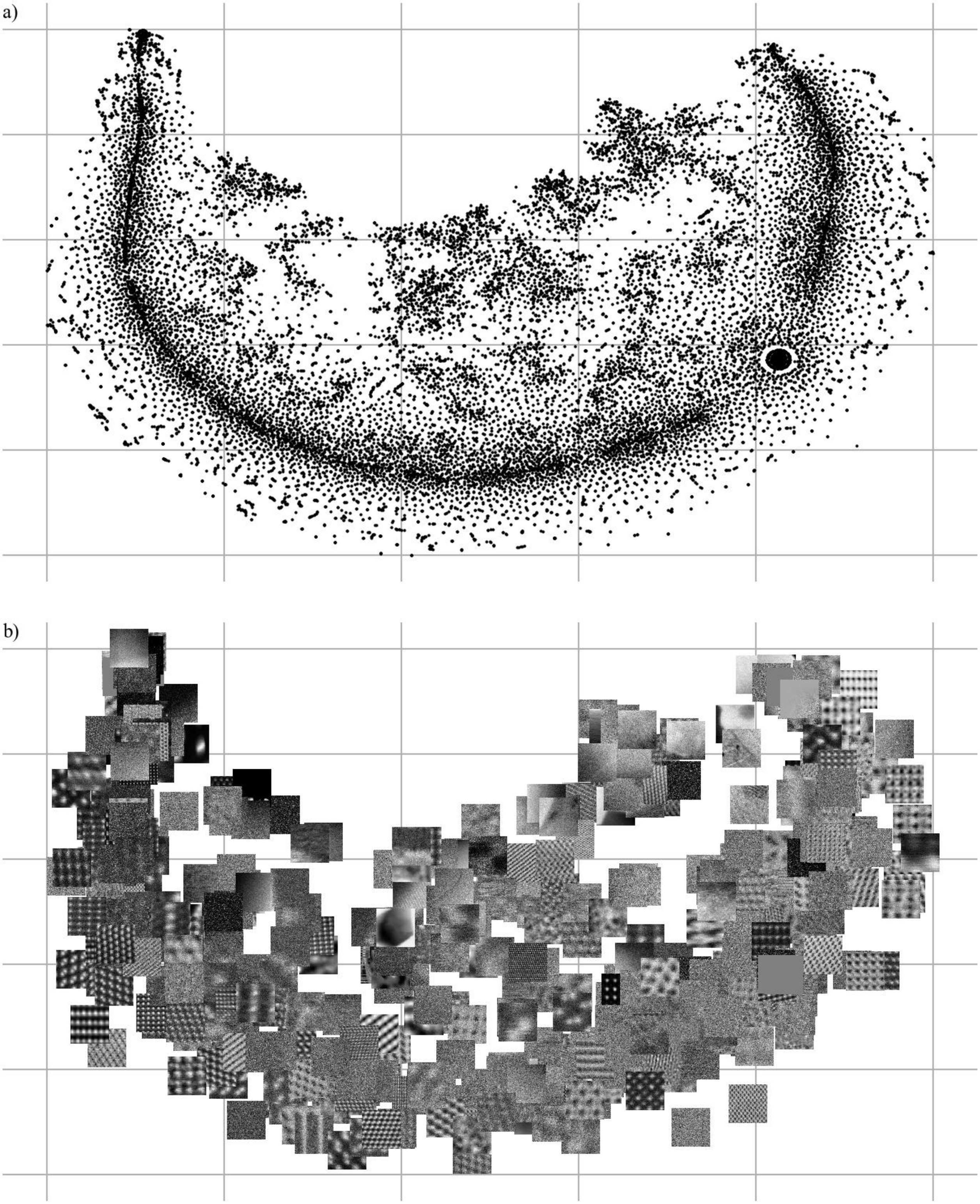


Figure S2. Two-dimensional tSNE visualization of the first 50 principal components of 19769 96×96 crops from STEM images. The same grid is used to show a) map points and b) images at 500 randomly selected points.

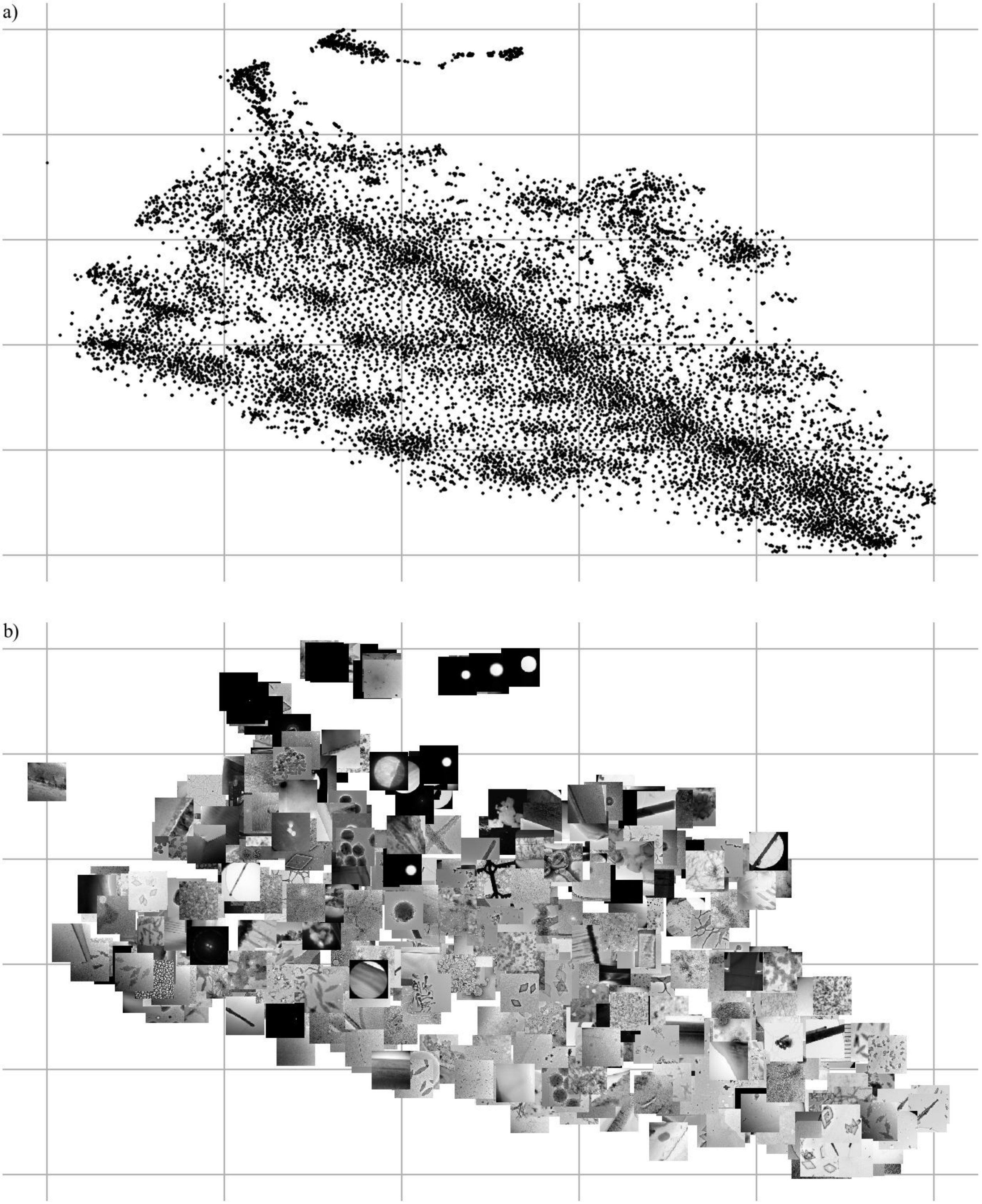


Figure S3. Two-dimensional tSNE visualization of the first 50 principal components of 17266 TEM images that have been downsampled to 96×96. The same grid is used to show a) map points and b) images at 500 randomly selected points.

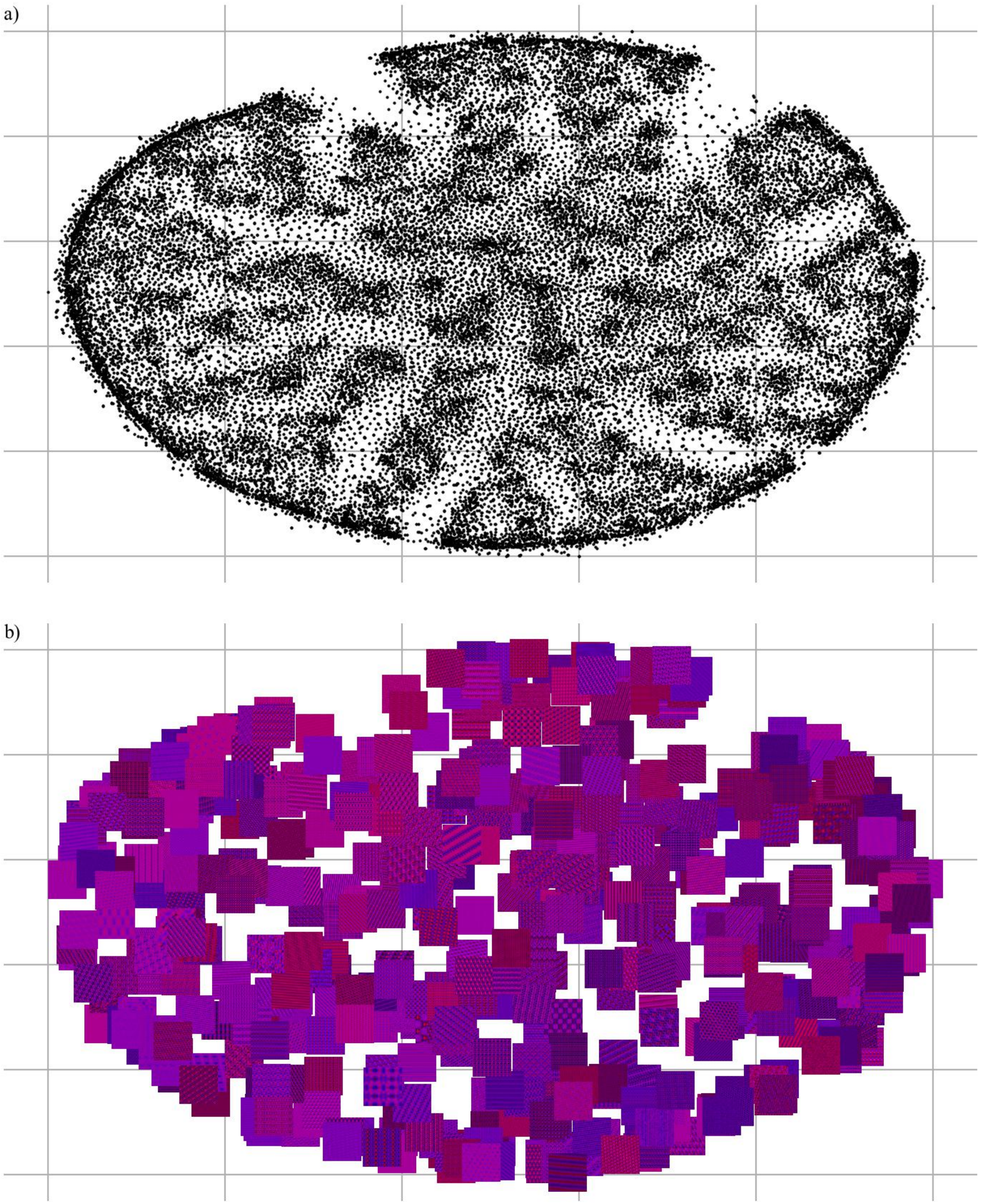


Figure S4. Two-dimensional tSNE visualization of the first 50 principal components of 36324 exit wavefunctions that have been downsampled to 96×96. Wavefunctions were simulated for thousands of materials and a large range of physical hyperparameters. The same grid is used to show a) map points and b) wavefunctions at 500 randomly selected points. Red and blue colour channels show real and imaginary components, respectively.

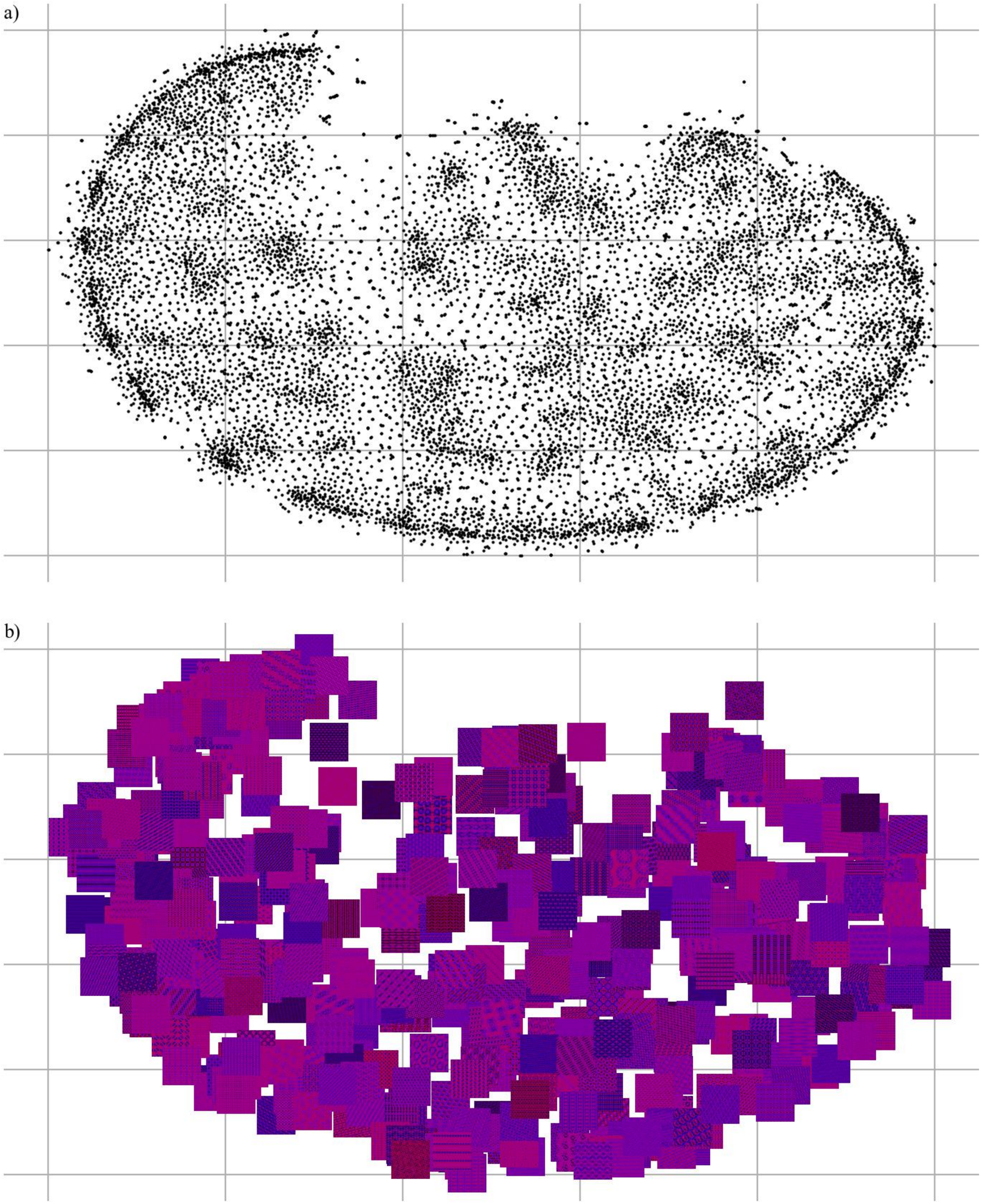


Figure S5. Two-dimensional tSNE visualization of the first 50 principal components of 11870 exit wavefunctions that have been downsampled to 96×96. Wavefunctions were simulated for thousands of materials and a small range of physical hyperparameters. The same grid is used to show a) map points and b) wavefunctions at 500 randomly selected points. Red and blue colour channels show real and imaginary components, respectively.

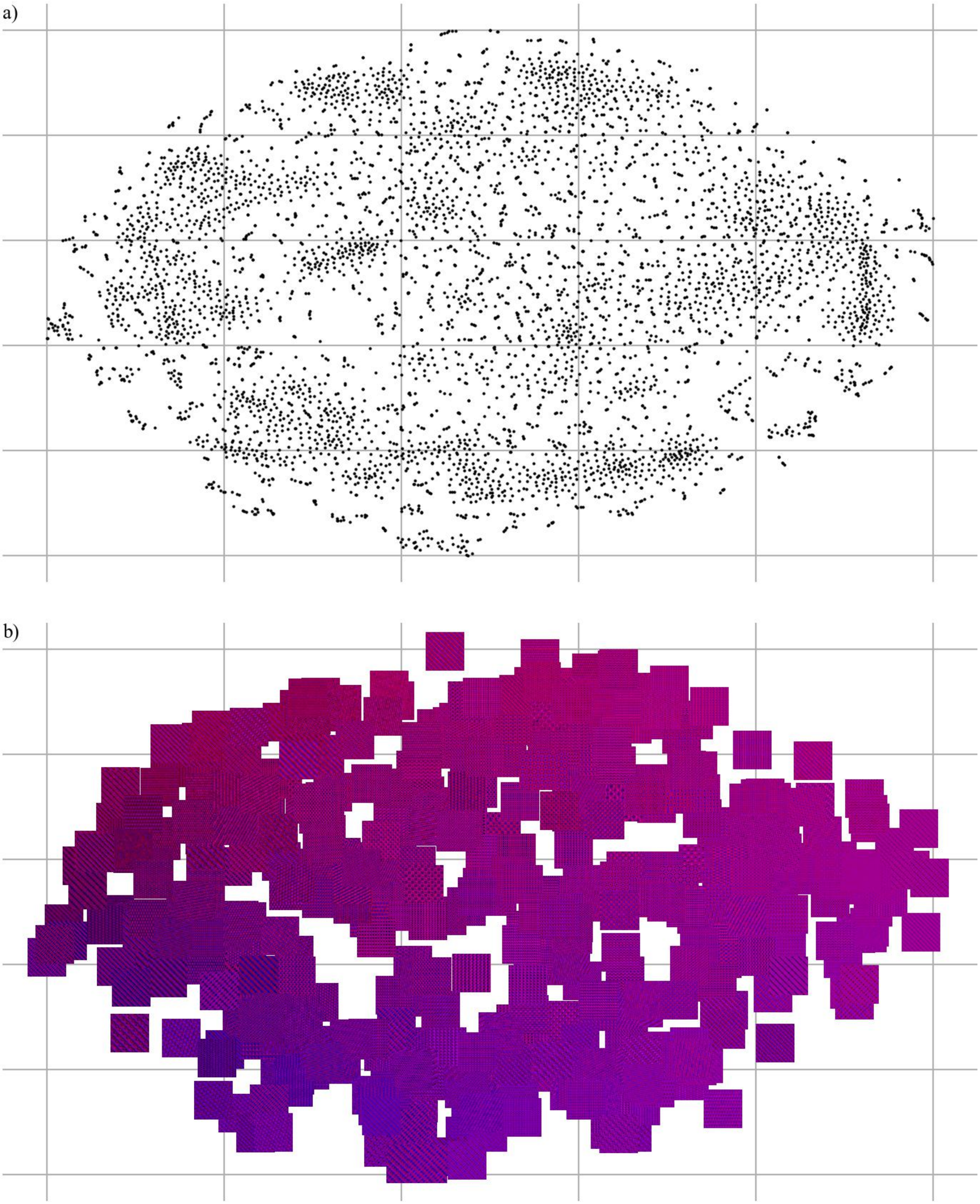


Figure S6. Two-dimensional tSNE visualization of the first 50 principal components of 4825 exit wavefunctions that have been downsampled to 96×96. Wavefunctions were simulated for thousands of materials and a small range of physical hyperparameters. The same grid is used to show a) map points and b) wavefunctions at 500 randomly selected points. Red and blue colour channels show real and imaginary components, respectively.

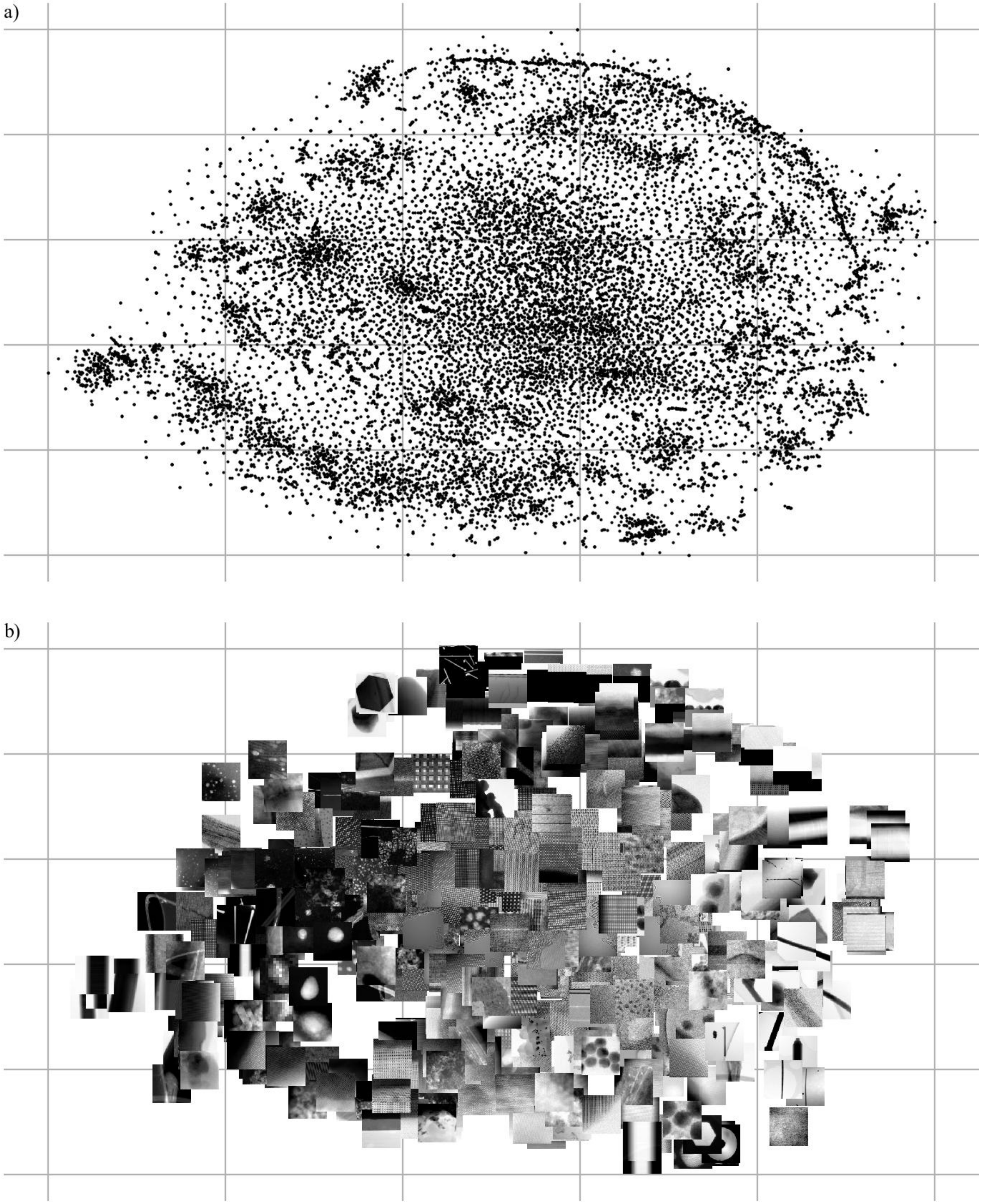


Figure S7. Two-dimensional tSNE visualization of means parameterized by 64-dimensional VAE latent spaces for 19769 STEM images that have been downsampled to 96×96. The same grid is used to show a) map points and b) images at 500 randomly selected points.

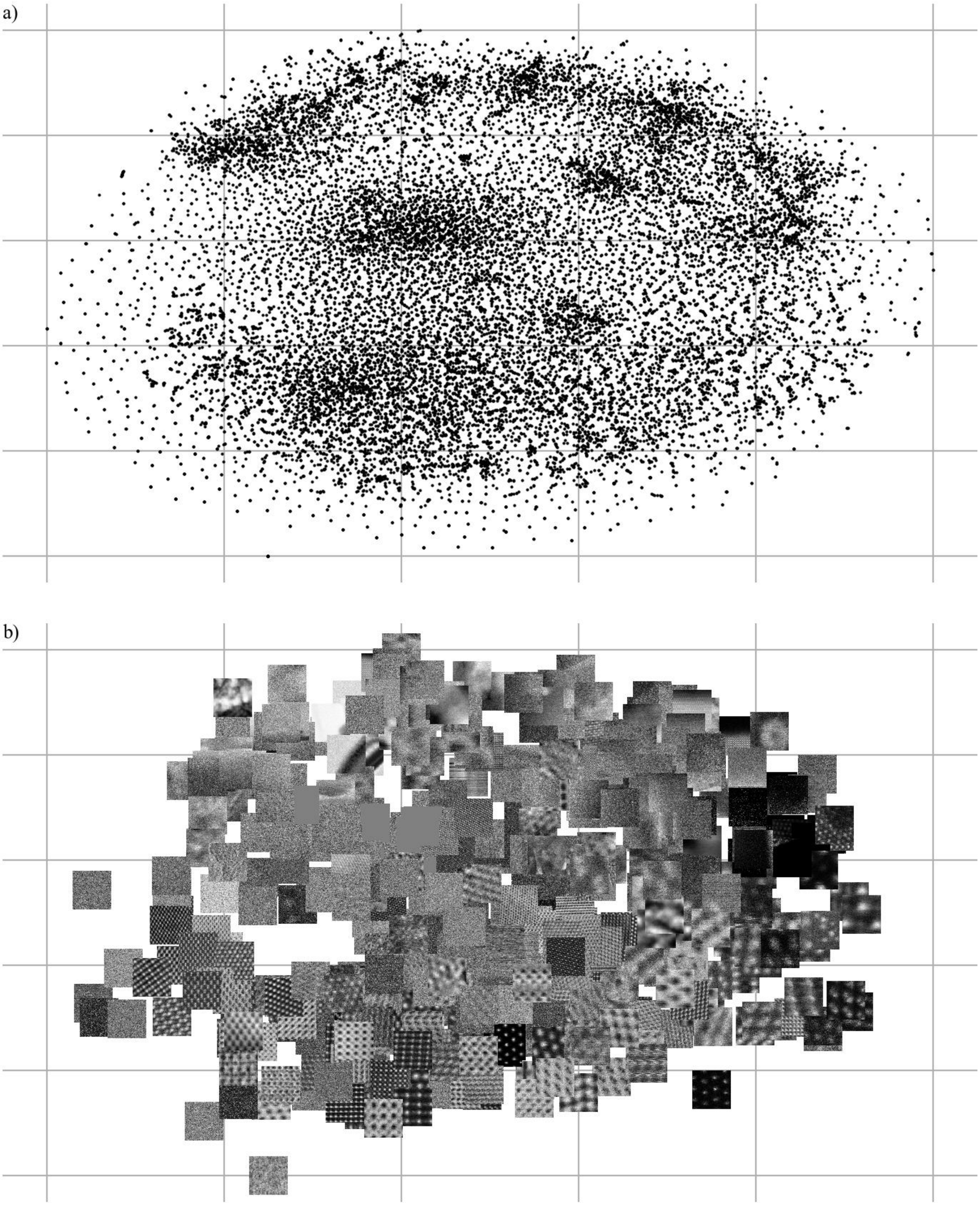


Figure S8. Two-dimensional tSNE visualization of means parameterized by 64-dimensional VAE latent spaces for 19769 96×96 crops from STEM images. The same grid is used to show a) map points and b) images at 500 randomly selected points.

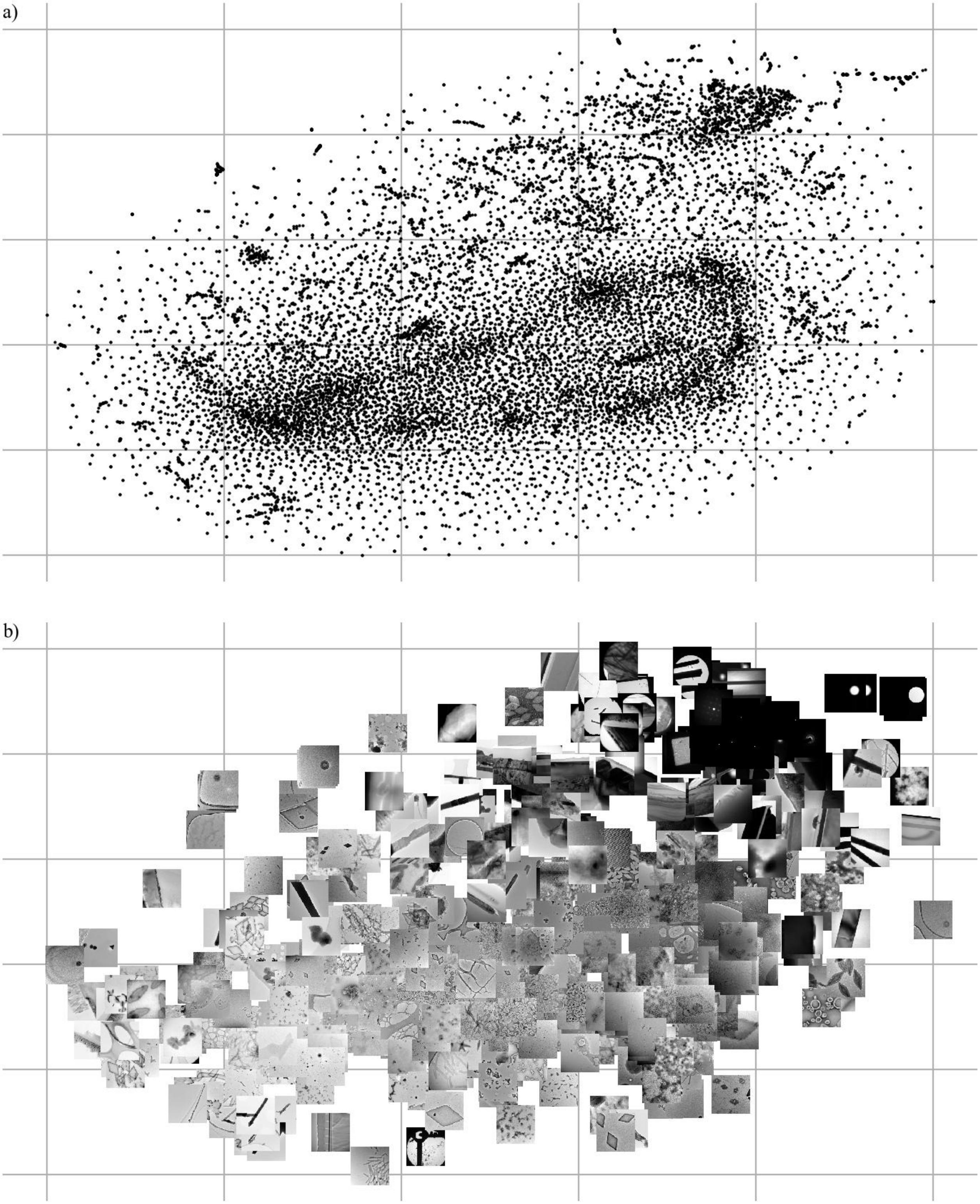


Figure S9. Two-dimensional tSNE visualization of means parameterized by 64-dimensional VAE latent spaces for 19769 TEM images that have been downsampled to 96×96. The same grid is used to show a) map points and b) images at 500 randomly selected points.

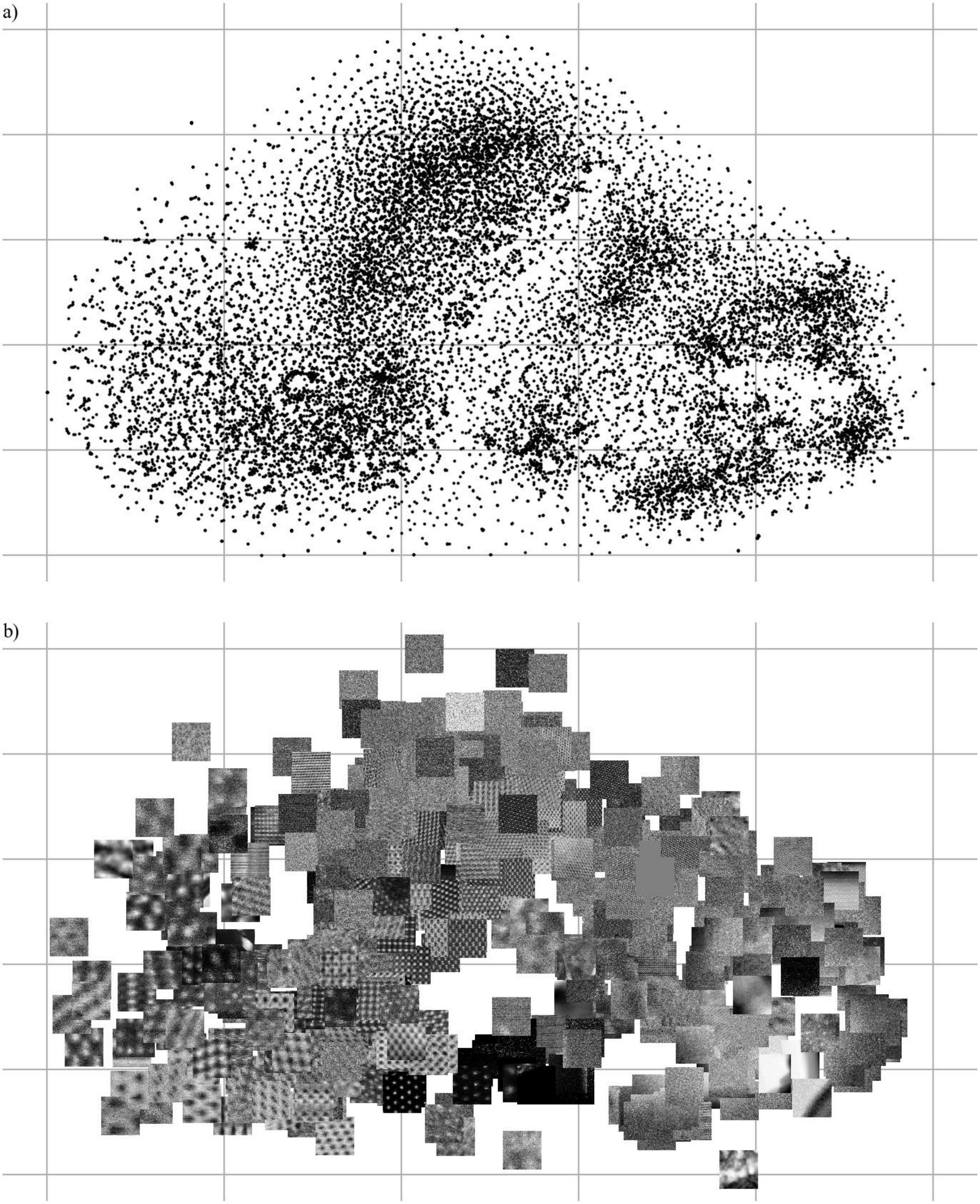


Figure S10. Two-dimensional tSNE visualization of means and standard deviations parameterized by 64-dimensional VAE latent spaces for 19769 96×96 crops from STEM images. The same grid is used to show a) map points and b) images at 500 randomly selected points.

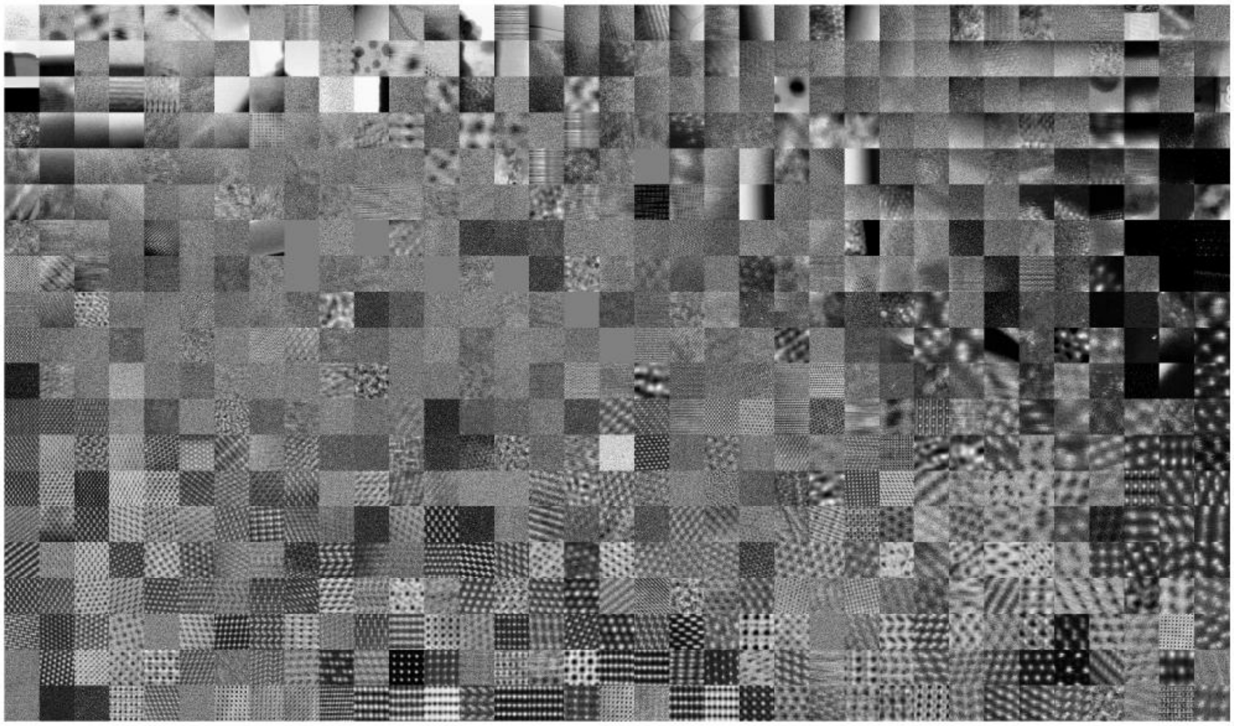


Figure S11. Two-dimensional uniformly separated tSNE visualization of 64-dimensional VAE latent spaces for 19769 96×96 crops from STEM images.

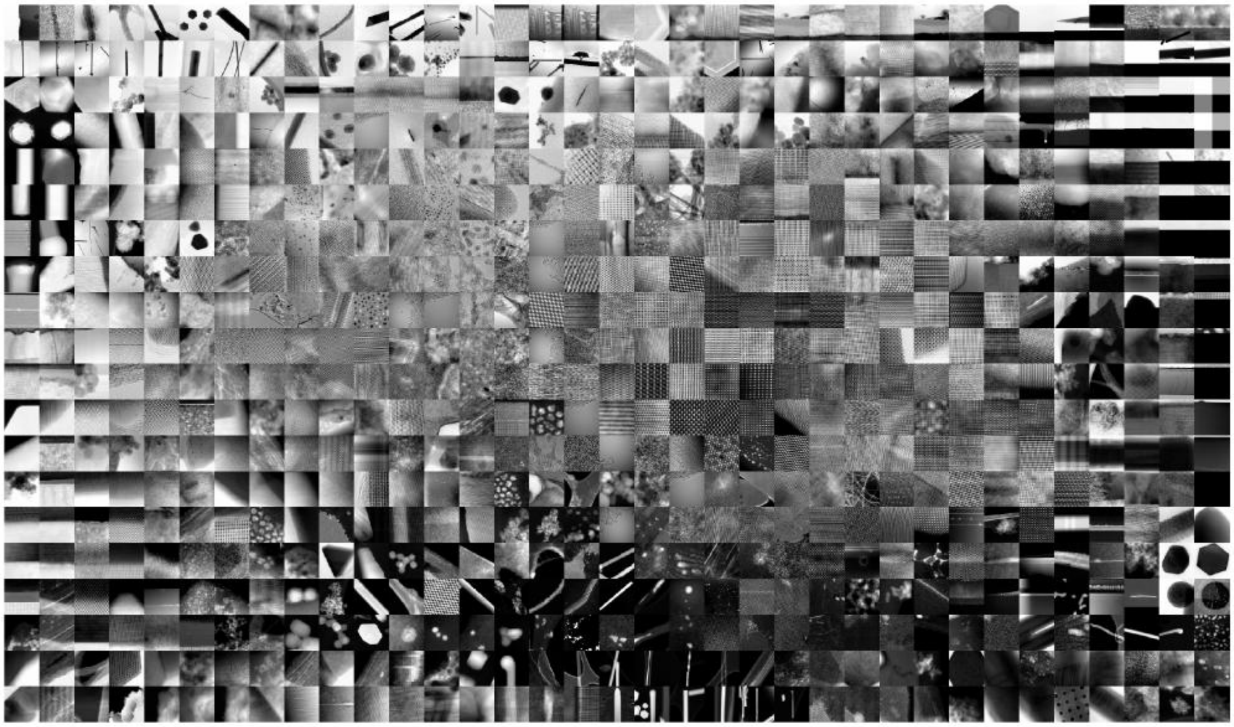


Figure S12. Two-dimensional uniformly separated tSNE visualization of 64-dimensional VAE latent spaces for 19769 STEM images that have been downsampled to 96×96.

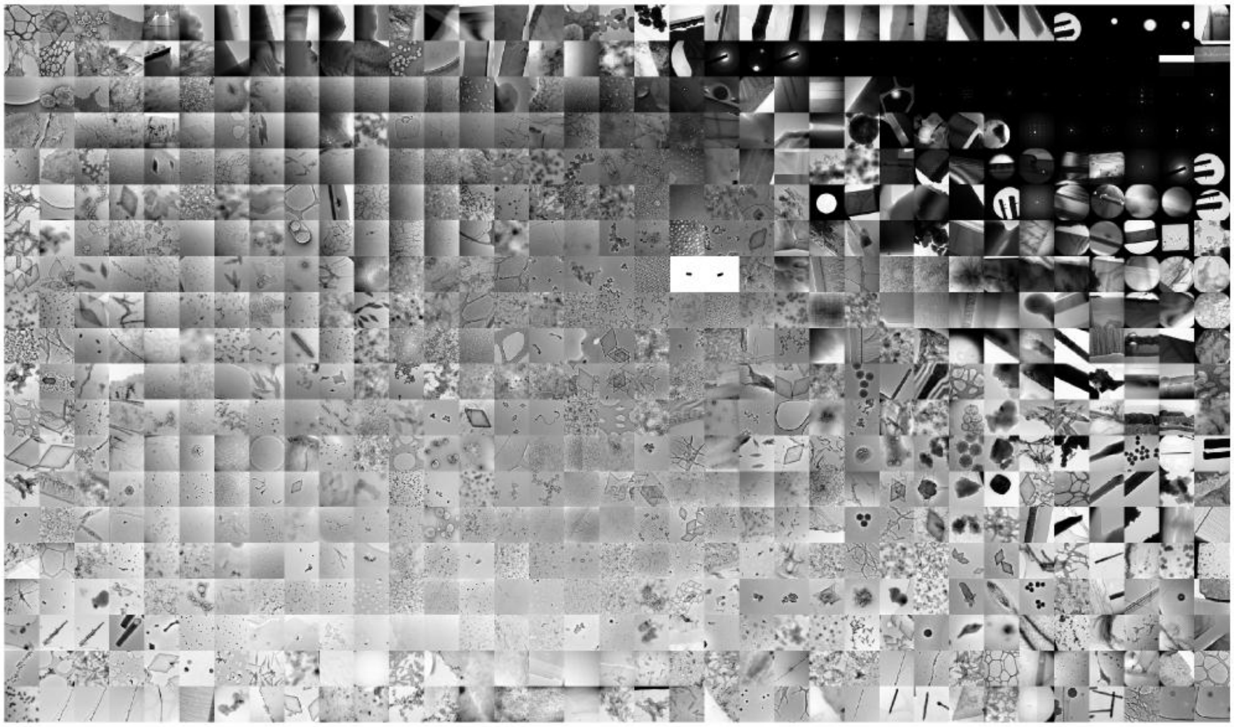


Figure S13. Two-dimensional uniformly separated tSNE visualization of 64-dimensional VAE latent spaces for 17266 TEM images that have been downsampled to 96×96.

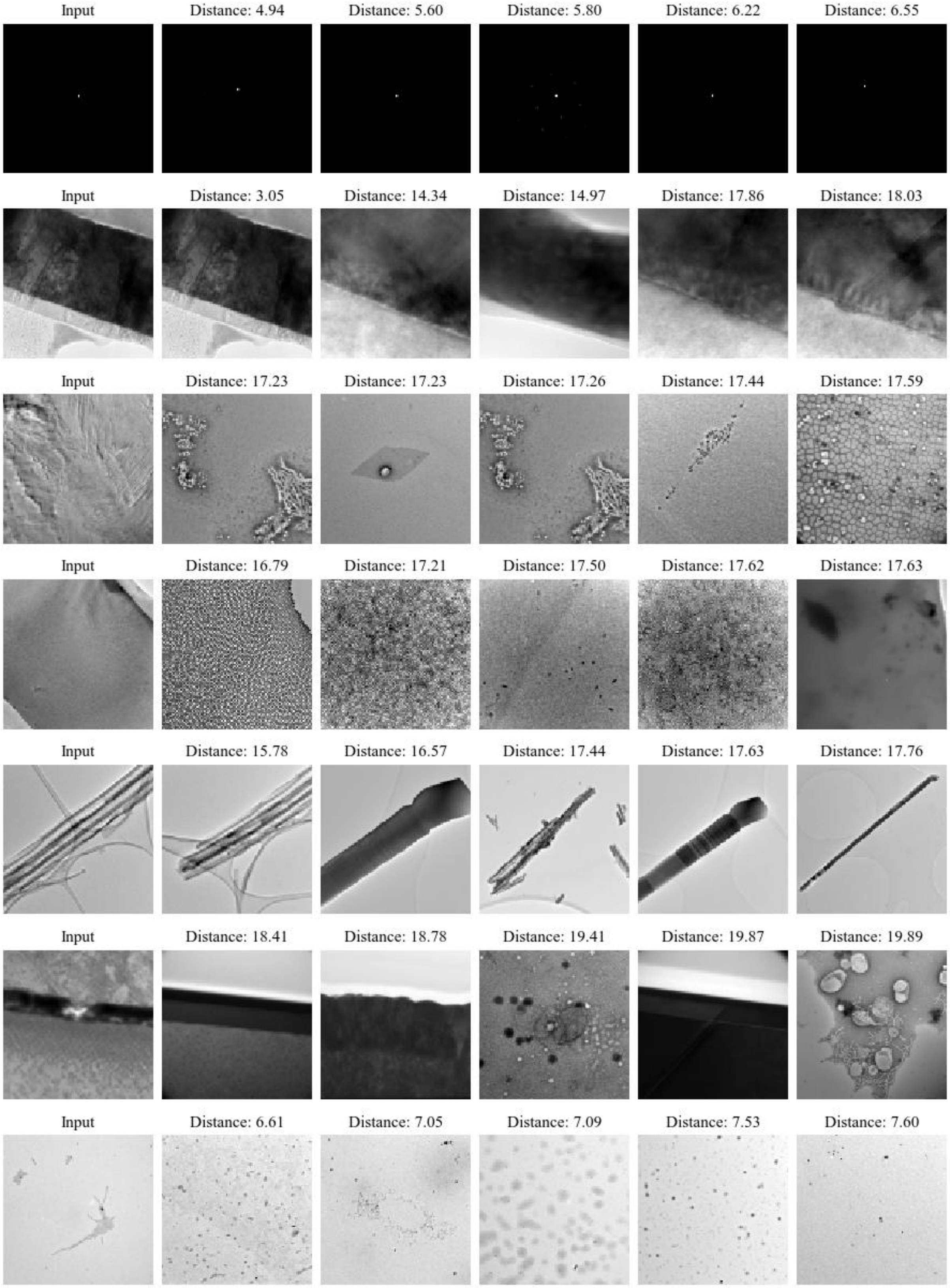


Figure S14. Examples of top-5 search results for 96×96 TEM images. Euclidean distances between μ encoded for search inputs and results are smaller for more similar images.

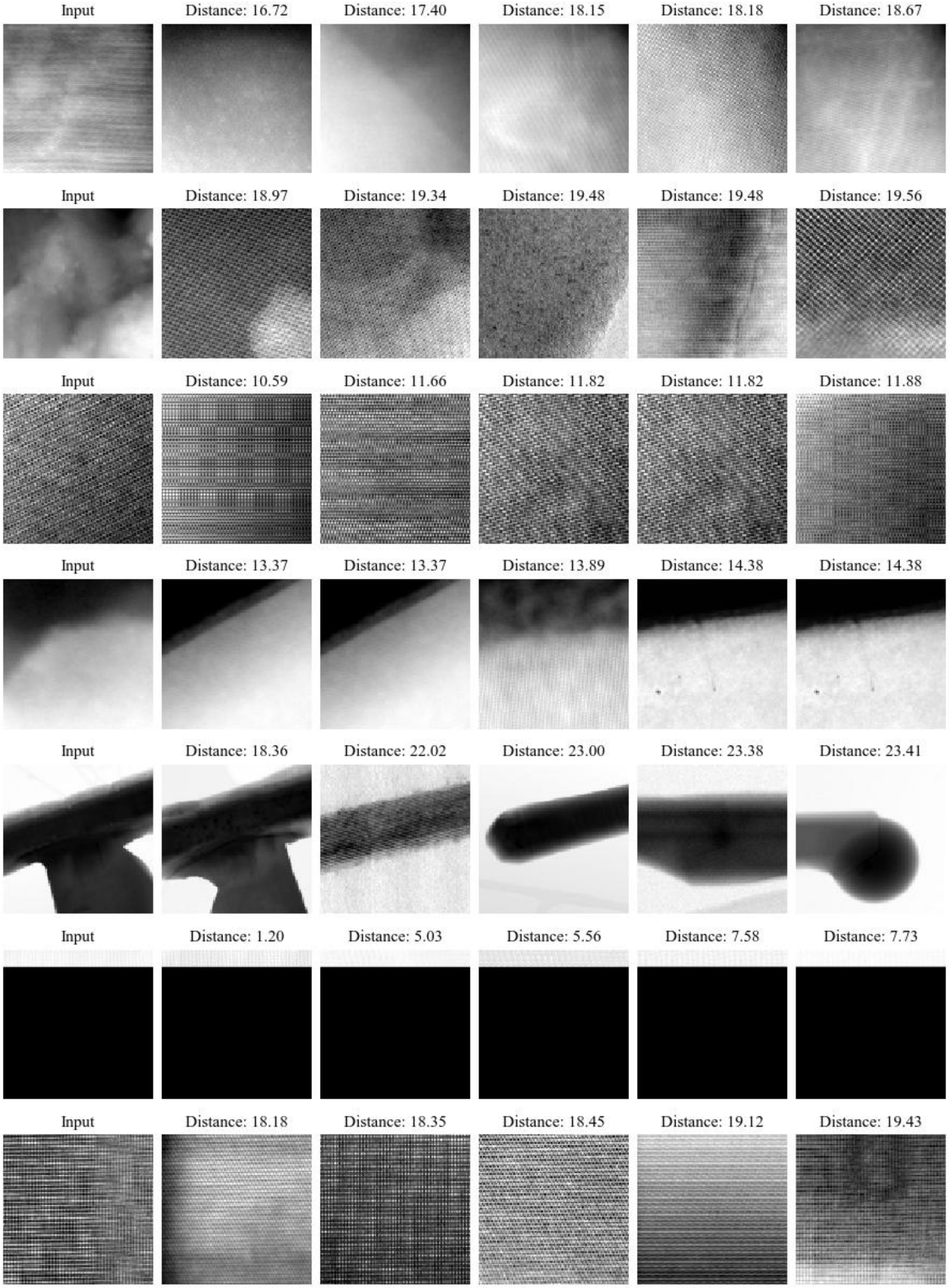


Figure S15. Examples of top-5 search results for 96×96 STEM images. Euclidean distances between μ encoded for search inputs and results are smaller for more similar images.

2.2 Amendments and Corrections

There are amendments or corrections to the paper[2] covered by this chapter.

Location: Page 4, caption of fig. 2.
Change: "...at 500 randomly selected images..." should say "...at 500 randomly selected data points...".

2.3 Reflection

This ancillary chapter covers my paper titled "Warwick Electron Microscopy Datasets"[2] and associated research outputs[9,13–15]. My paper presents visualizations for large new electron microscopy datasets published with our earlier papers. There are 17266 TEM images curated to train our denoiser[6] (ch. 6), 98340 STEM images curated to train generative adversarial networks (GANs) for compressed sensing[4,19] (ch. 4), and 98340 TEM exit wavefunctions simulated to investigate EWR[7] (ch. 7), and derived datasets containing smaller TEM and STEM images that I created to rapidly prototype of ANNs for adaptive partial STEM[5] (ch. 5). To improve visualizations, I developed new regularization mechanisms for variational autoencoders[107–109] (VAEs), which were trained to embed high-dimensional electron micrographs in low-dimensional latent spaces. In addition, I demonstrate that VAEs can be used as the basis of electron micrograph search engines. Finally, I provide extensions to t-distributed stochastic neighbour embedding[110–114] (tSNE) and interactive dataset visualizations.

Making our large machine learning datasets openly accessible enables our research to be reproduced[115], standardization of performance comparisons, and dataset reuse in future research. Dissemination of large datasets is enabled by the internet[116,117], for example, through fibre optic[118] broadband[119,120] or satellite[121,122] connections. Subsequently, there are millions of open access datasets[123,124] that can be used for machine learning[125,126]. Performance of ANNs usually increases with increasing training dataset size[125], so some machine learning datasets have millions of examples. Examples of datasets with millions of examples include DeepMind Kinetics[127], ImageNet[128], and YouTube 8M[129]. Nevertheless, our datasets containing tens of thousands of examples are more than sufficient for initial exploration of deep learning in electron microscopy. For reference, some datasets used for initial explorations of deep learning for Coronavirus Disease 2019[130–132] (COVID-19) diagnosis are 10× smaller[133] than WEMD.

There are many data clustering algorithms[134–140] that can group data for visualization. However, tSNE is a *de facto* default as it often outperforms other algorithms[110]. For context, tSNE is a variant of stochastic neighbour embedding[141] (SNE) where a heavy-tailed Student's t-distribution is used to measure distances between embedded data points. Applications of tSNE include bioinformatics[142,143], forensic science[144,145], medical signal processing[146–148], particle physics[149,150], smart electricity metering[151], and sound synthesis[152]. Before tSNE, data is often embedded in a low-dimensional space to reduce computation, suppress noise, and prevent Euclidean distances used in tSNE optimization being afflicted by the curse of dimensionality[153]. For example, the original tSNE paper suggests using principal component analysis[154–157] (PCA) to reduce data dimensionality to 30 before applying tSNE[110].

Extensions of tSNE can improve clustering. For example, graphical processing unit accelerated implementations of tSNE[158,159] can speedup clustering 50-700×. Alternatively, approximate tSNE[160] (A-tSNE) can trade accuracy for decreased computation time. Our tSNE visualizations took a couple of hours to optimize on an Intel i7-6700 central processing unit (CPU) as we used 10000 iterations to ensure that clusters stabilized. It follows that accelerated tSNE implementations may be preferable to reduce computation time. Another extension is to adjust distances used for tSNE optimization with a power transform based on the intrinsic dimension of each point. This can alleviate the curse of dimensionality for high-dimensional data[153]; however, it was not necessary for our data as I used VAEs to reduce image dimensionality to 64 before tSNE. Finally, tSNE early exaggeration (EE), where probabilities modelling distances in a high-dimensional space are increased, and number of iterations can be automatically tuned with opt-tSNE[161]. Tuning can significantly improve visualizations, especially for large datasets with millions of examples. However, I doubt that opt-tSNE would result in large improvements to clustering as our datasets contain tens of thousands of examples, where tSNE is effective. Nevertheless, I expect that opt-tSNE could have improved clustering if I had been aware of it.

Further extensions to tSNE are proposed in my paper[2,9]. I think that the most useful extension uniformly separates clustered points based clustering density. Uniformly separated tSNE (US-tSNE) can often double whitespace utilization, which could make tSNE visualizations more suitable for journals, websites, and other media where space is limited. However, the increased whitespace utilization comes at the cost of removing information about the structure of clusters. Further, my preliminary implementation of US-tSNE is limited insofar that Clough-Tocher cubic Bezier interpolation[162] used to map tSNE points to a uniform map is only applied to points within their convex hull. I also proposed a tSNE extension that uses standard deviations encoded by VAEs to inform clustering as this appeared to slightly improve clustering. However, I later found that using standard deviations appears to decrease similarity of nearest neighbours in tSNE visualizations. As a result, I think that how extra information encoded in standard deviations is used to inform clustering may merit further investigation.

To improve VAE encodings for tSNE, I applied a variant of batch normalization to their latent spaces. This avoids needing to tune a hyperparameter to balance VAE decoder and Kullback-Leibler (KL) losses, which is architecture-specific and can be complicated by relative sizes of their gradients varying throughout training. I also considered adaptive gradient balancing[163,164] of losses; however, that would require separate backpropagation through the VAE generator for each loss, increasing computation. To increase image realism, I added Sobel losses to mean squared errors (MSEs). Sobel losses often improve realism as human vision is sensitive to edges[165]. In addition, Sobel losses require less computation than VAE training with GAN[166] or perceptual[167] losses. Another computationally inexpensive approach to improve generated image realism is to train with structural similarity index measures[168] (SSIMs) instead of MSEs[169].

My VAEs are used as the basis of my openly accessible electron microscopy search engines. I observe that top-5 search results are usually successful insofar that they contain images that are similar to input images. However, they often contain some images that are not similar, possibly due to there not being many similar images in our datasets. Thus, I expect that search results could be improved by increasing dataset size. Increasing input image size from 96×96 to a couple of hundred pixels and increasing training iterations could also improve performance. Further, training could be modified to encode binary latent variables for efficient hashing[170–175]. Finally, I think that an interesting research direction is to create a web interface for an electron microscopy search engine that indexes institutional electron microscopy data servers. Such a search engine could enhance collaboration by making it easier to find electron microscopists working on interesting projects.

An application of my VAEs that is omitted from my paper is that VAE generators could function as portable electron microscopy image generators. For example, to create training data for machine learning. For comparison, my VAE generators require roughly 0.1% of the storage space needed for my image datasets to store their trainable parameters. However, I was concerned that a distribution of generated images might be biased by catastrophic forgetting[176]. Further, a distribution of generated images could be sensitive to ANN architecture and learning policy, including when training is stopped[177,178]. Nevertheless, I expect that data generated from by VAEs could be used for pretraining to improve ANN robustness[179]. Overall, I think it will become increasingly practical to use VAEs or GANs as high-quality data generators as ANN architectures and learning policies are improved.

Perhaps the main limitation of my paper is that I did not introduce my preferred abbreviation, "WEMD", for "Warwick Electron Microscopy Datasets". Further, I did not define "WEMD" in my WEMD preprint[13]. Subsequently, I introduced my preferred abbreviation in my review of deep learning in electron microscopy[1] (ch. 1). I also defined an abbreviation, "WLEMD", for "Warwick Large Electron Microscopy Datasets" in the first version of the partial STEM preprint[18] (ch. 4). Another limitation is that my paper only details datasets that had already been published, or that were derived from the published datasets. For example, Richard Beanland and I successfully co-authored an application for funding to simulate tens of thousands of CBED patterns with Felix[180], which are not detailed in my paper. The CBED dataset requires a couple of terabytes of storage and has not been processed for dissemination. Nevertheless, Richard Beanland[1] may be able to provide the CBED dataset upon request.

---

[1] Email: r.beanland@warwick.ac.uk

Chapter 3

Adaptive Learning Rate Clipping Stabilizes Learning

3.1 Scientific Paper

This chapter covers the following paper[3].

> J. M. Ede and R. Beanland. Adaptive Learning Rate Clipping Stabilizes Learning. *Machine Learning: Science and Technology*, 1:015011, 2020

IOP Publishing *Mach. Learn.: Sci. Technol.* **1** (2020) 015011 https://doi.org/10.1088/2632-2153/ab81e2

MACHINE
LEARNING
Science and Technology

PAPER

Adaptive learning rate clipping stabilizes learning

Jeffrey M Ede[1] **and Richard Beanland**

Department of Physics, University of Warwick, Coventry CV4 7AL United Kingdom

E-mail: j.m.ede@warwick.ac.uk and r.beanland@warwick.ac.uk

Keywords: machine learning, optimization, electron microscopy, learning stability

Supplementary material for this article is available online

CrossMark

OPEN ACCESS

RECEIVED
20 December 2019

REVISED
26 February 2020

ACCEPTED FOR PUBLICATION
20 March 2020

PUBLISHED
28 April 2020

Original Content from this work may be used under the terms of the Creative Commons Attribution 4.0 licence.

Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

Abstract

Artificial neural network training with gradient descent can be destabilized by 'bad batches' with high losses. This is often problematic for training with small batch sizes, high order loss functions or unstably high learning rates. To stabilize learning, we have developed adaptive learning rate clipping (ALRC) to limit backpropagated losses to a number of standard deviations above their running means. ALRC is designed to complement existing learning algorithms: Our algorithm is computationally inexpensive, can be applied to any loss function or batch size, is robust to hyperparameter choices and does not affect backpropagated gradient distributions. Experiments with CIFAR-10 supersampling show that ALCR decreases errors for unstable mean quartic error training while stable mean squared error training is unaffected. We also show that ALRC decreases unstable mean squared errors for scanning transmission electron microscopy supersampling and partial scan completion. Our source code is available at https://github.com/Jeffrey-Ede/ALRC.

1. Introduction

Loss spikes arise when artificial neural networks (ANNs) encounter difficult examples and can destabilize training with gradient descent[1, 2]. Examples may be difficult because an ANN needs more training to generalize, catastrophically forgot previous learning [3] or because an example is complex or unusual. Whatever the cause, applying gradients backpropagated [4] from high losses results in large perturbations to trainable parameters.

When a trainable parameter perturbation is much larger than others, learning can be destabilized while parameters adapt. This behaviour is common for ANN training with gradient descent where a large portion of parameters is perturbed at each optimization step. In contrast, biological networks often perturb small portions of neurons to combine new learning with previous learning. Similar to biological networks, ANN layers can become more specialized throughout training [5] and specialized capsule networks [6] are being developed. Nevertheless, ANN loss spikes during optimization are still a common reason for learning instability. Loss spikes are common for training with small batch sizes, high order loss functions, and unstably high learning rates.

During ANN training by stochastic gradient descent [1] (SGD), a trainable parameter, θ_t, from step t is updated to θ_{t+1} in step $t+1$. The size of the update is given by the product of a learning rate, η, and the backpropagated gradient of a loss function with respect to the trainable parameter

$$\theta_{t+1} \leftarrow \theta_t - \eta \frac{\partial L}{\partial \theta}. \quad (1)$$

Without modification, trainable parameter perturbations are proportional to the scale of the loss function. Following gradient backpropagation, a high loss spike can cause a large perturbation to a learned parameter distribution. Learning will then be destabilized while subsequent iterations update trainable parameters back to an intelligent distribution.

---

[1] Author to whom any correspondence should be addressed.

Trainable parameter perturbations are often limited by clipping gradients to a multiple of their global L2 norm [7]. For large batch sizes, this can limit perturbations by loss spikes as their gradients will be larger than other gradients in the batch. However, global L2 norm clipping alters the distribution of gradients backpropagated from high losses and is unable to identify and clip high losses if the batch size is small. Clipping gradients of individual layers by their L2 norms has the same limitations.

Gradient clipping to a user-provided threshold [8] can also be applied globally or to individual layers. This can limit loss spike perturbations for any batch size. However, the clipping threshold is an extra hyperparameter to determine and may need to be changed throughout training. Further, it does not preserve distributions of gradients for high losses.

More commonly, destabilizing perturbations are reduced by selecting a low order loss function and stable learning rate. Low order loss functions, such as absolute and squared distances, are effective because they are less susceptible to destabilizingly high errors than higher-order loss functions. Indeed, loss function modifications used to stabilize learning often lower loss function order. For instance, Huberization [9, 10] reduces perturbations by losses, L, larger than h by applying the mapping $L \to \min(L, (hL)^{1/2})$.

2. Algorithm

Adaptive learning rate clipping (ALRC, algorithm 1) is designed to address the limitations of gradient clipping. Namely, to be computationally inexpensive, effective for any batch size, robust to hyperparameter choices and to preserve backpropagated gradient distributions. Like gradient clipping, ALRC also has to be applicable to arbitrary loss functions and neural network architectures.

Rather than allowing loss spikes to destabilize learning, ALRC applies the mapping $\eta L \to \text{stop\_gradient}(L_\text{max}/L)\eta L$ if $L > L_\text{max}$. The function stop$_\text{g}$radient leaves its operand unchanged in the forward pass and blocks gradients in the backwards pass. ALRC adapts the learning rate to limit the effective loss being backpropagated to L_max. The value of L_max is non-trivial for ALRC to complement existing learning algorithms. In addition to training stability and robustness to hyperparameter choices, L_max needs to adapt to losses and learning rates as they vary.

In our implementation, L_max and L_min are numbers of standard deviations of the loss above and below its mean, respectively. ALRC has six hyperparameters; however, it is robust to their values. There are two decay rates, β_1 and β_2, for exponential moving averages used to estimate the mean and standard deviation of the loss and a number, n, of standard deviations. Similar to batch normalization [11], any decay rate close to 1 is effective e.g. $\beta_1 = \beta_2 = 0.999$. Performance does vary slightly with n_max; however, we found that any $n_\text{max} \approx 3$ is effective. Varying n_min is an optional extension and we default to one-sided ALRC above i.e. $n_\text{min} = \infty$. Initial values for the running means, μ_1 and μ_2, where $\mu_1^2 < \mu_2$ also have to be provided. However, any sensible initial estimates larger than their true values are fine as μ_1 and μ_2 will decay to their correct values.

ALRC can be extended to any loss function or batch size. For batch sizes above 1, we apply ALRC to individual losses, while μ_1 and μ_2 are updated with mean losses. ARLC can also be applied to loss summands, such as per pixel errors between generated and reference images, while μ_1 and μ_2 are updated with the mean errors.

Algorithm 1 Two-sided adaptive learning rate clipping (ALRC) of loss spikes. Sensible parameters are $\beta_1 = \beta_2 = 0.999$, $n_\text{min} = \infty$, $n_\text{max} = 3$, and $\mu_1^2 < \mu_2$.

```
Initialize running means, $\mu_1$ and $\mu_2$, with decay rates, $\beta_1$ and $\beta_2$.
Choose number, $n$, of standard deviations to clip to.
While Training is not finished do
    Infer forward-propagation loss, $L$.
    $\sigma \leftarrow (\mu_2 - \mu_1^2)^{1/2}$
    $L_\text{min} \leftarrow \mu_1 - n_\text{min}\sigma$
    $L_\text{max} \leftarrow \mu_1 + n_\text{max}\sigma$
    If $L < L_\text{min}$ then
        $L_\text{dyn} \leftarrow \text{stop\_gradient}(L_\text{min}/L)L$
    else if $L > L_\text{max}$ then
        $L_\text{dyn} \leftarrow \text{stop\_gradient}(L_\text{max}/L)L$
    else
        $L_\text{dyn} \leftarrow L$
    end if
    Optimize network by back-propagating $L_\text{dyn}$.
    $\mu_1 \leftarrow \beta_1\mu_1 + (1 - \beta_1)L$
    $\mu_2 \leftarrow \beta_2\mu_2 + (1 - \beta_2)L^2$
end while
```

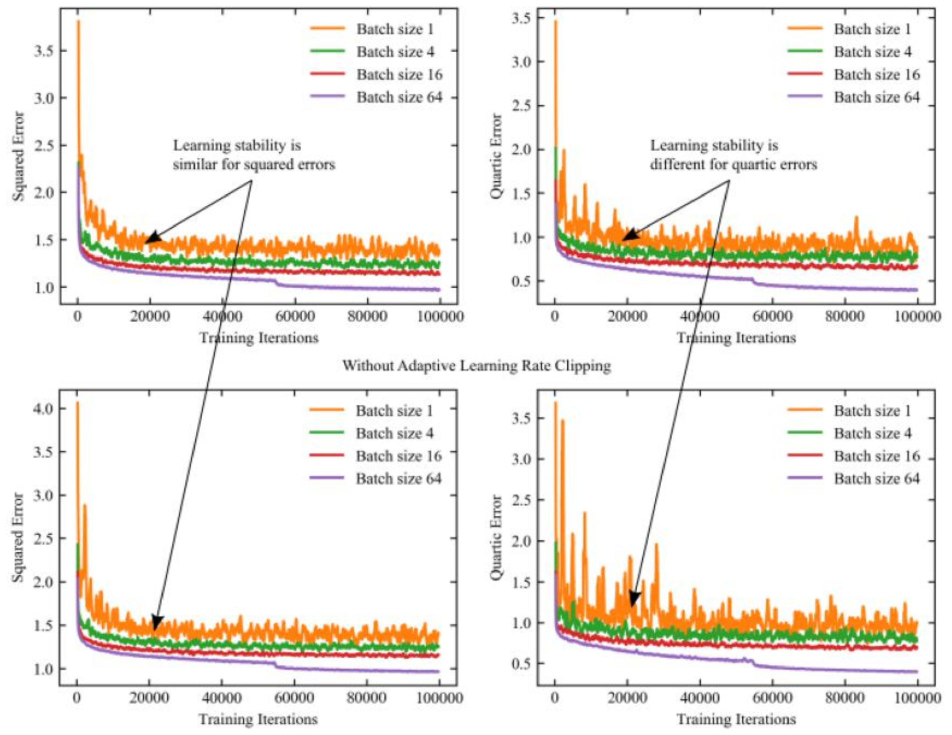


Figure 1. Unclipped learning curves for 2× CIFAR-10 supersampling with batch sizes 1, 4, 16 and 64 with and without adaptive learning rate clipping of losses to 3 standard deviations above their running means. Training is more stable for squared errors than quartic errors. Learning curves are 500 iteration boxcar averaged.

Table 1. Adaptive learning rate clipping (ALRC) for losses 2, 3, 4 and ∞ running standard deviations above their running means for batch sizes 1, 4, 16 and 64. ARLC was not applied for clipping at ∞. Each squared and quartic error mean and standard deviation is for the means of the final 5000 training errors of 10 experiments. ALRC lowers errors for unstable quartic error training at low batch sizes and otherwise has little effect. Means and standard deviations are multiplied by 100.

Squared Errors

	Batch Size 1		Batch Size 4		Batch Size 16		Batch Size 64	
Threshold	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
2	5.55	0.048	4.96	0.016	4.58	0.010	–	–
3	5.52	0.054	4.96	0.029	4.58	0.004	3.90	0.013
4	5.56	0.048	4.97	0.017	4.58	0.007	3.89	0.016
∞	5.55	0.041	4.98	0.017	4.59	0.006	3.89	0.014

Quartic Errors

	Batch Size 1		Batch Size 4		Batch Size 16		Batch Size 64	
Threshold	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
2	3.54	0.084	3.02	0.023	2.60	0.012	1.65	0.011
3	3.59	0.055	3.08	0.024	2.61	0.014	1.58	0.016
4	3.61	0.054	3.13	0.023	2.64	0.016	1.57	0.016
∞	3.88	0.108	3.32	0.037	2.74	0.020	1.61	0.008

3. Experiments: CIFAR-10 supersampling

To investigate the ability of ALRC to stabilize learning and its robustness to hyperparameter choices, we performed a series of toy experiments with networks trained to supersample CIFAR-10 [12, 13] images to 32×32×3 after downsampling to 16×16×3.

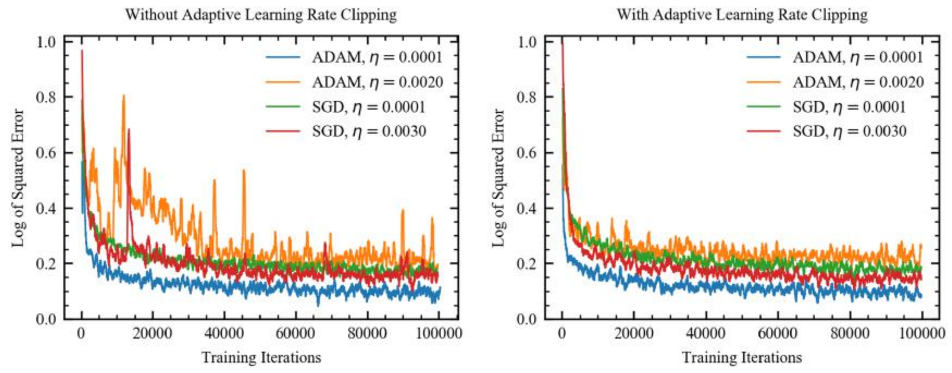


Figure 2. Unclipped learning curves for 2× CIFAR-10 supersampling with ADAM and SGD optimizers at stable and unstably high learning rates, η. Adaptive learning rate clipping prevents loss spikes and decreases errors at unstably high learning rates. Learning curves are 500 iteration boxcar averaged.

Data pipeline: In order, images were randomly flipped left or right, had their brightness altered, had their contrast altered, were linearly transformed to have zero mean and unit variance and bilinearly downsampled to $16\times16\times3$.
Architecture: Images were upsampled and passed through a convolutional neural network [14, 15] shown in figure 5. Each convolutional layer is followed by ReLU [16] activation, except the last.
Initialization: All weights were Xavier [17] initialized. Biases were zero initialized.
Learning policy: ADAM optimization was used with the hyperparameters recommended in [18] and a base learning rate of 1/1280 for 100 000 iterations. The learning rate was constant in batch size 1, 4, 16 experiments and decreased to 1/12 800 after 54 687 iterations in batch size 64 experiments. Networks were trained to minimize mean squared or quartic errors between restored and ground truth images. ALRC was applied to limit the magnitudes of losses to either 2, 3, 4 or ∞ standard deviations above their running means. For batch sizes above 1, ALRC was applied to each loss individually.
Results: Example learning curves for mean squared and quartic error training are shown in figure 1. Training is more stable and converges to lower losses for larger batch sizes. However, learning is less stable for quartic errors than squared errors, allowing ALRC to be examined for loss functions with different stability. Training was repeated 10 times for each combination of ALRC threshold and batch size. Means and standard deviations of the means of the last 5000 training losses for each experiment are tabulated in table 1. ALRC has no effect on mean squared error (MSE) training, even for batch size 1. However, it decreases errors for batch sizes 1, 4 and 16 for mean quartic error training.

Additional learning curves are shown in figure 2 for both ADAM and SGD optimizers to showcase the effect of ALRC on unstably high learning rates. Experiments are for a batch size of 1. ALRC has no effect at stable learning rates where learning is unaffected by loss spikes. However, ALRC prevents loss spikes and decreases errors at unstably high learning rates. In addition, these experiments show that ALRC is effective for different optimizers.

4. Experiments: partial STEM

To test ALRC in practice, we applied our algorithm to neural networks learning to complete 512×512 scanning transmission electron microscopy (STEM) images [19] from partial scans [20] with 1/20 coverage. Example completions are shown in figure 3.
Data pipeline: In order, each image was subject to a random combination of flips and 90° rotations to augment the dataset by a factor of 8. Next, each STEM image was blurred, and a path described by a 1/20 coverage spiral was selected. Finally, artificial noise was added to scans to make them more difficult to complete.
Architecture: Our network can be divided into three subnetworks shown in figure 6: an inner generator, outer generator and an auxiliary inner generator trainer. The auxiliary trainer [21, 22] is introduced to provide a more direct path for gradients to backpropagate to the inner generator. Each convolutional layer is followed by ReLU activation, except the last.

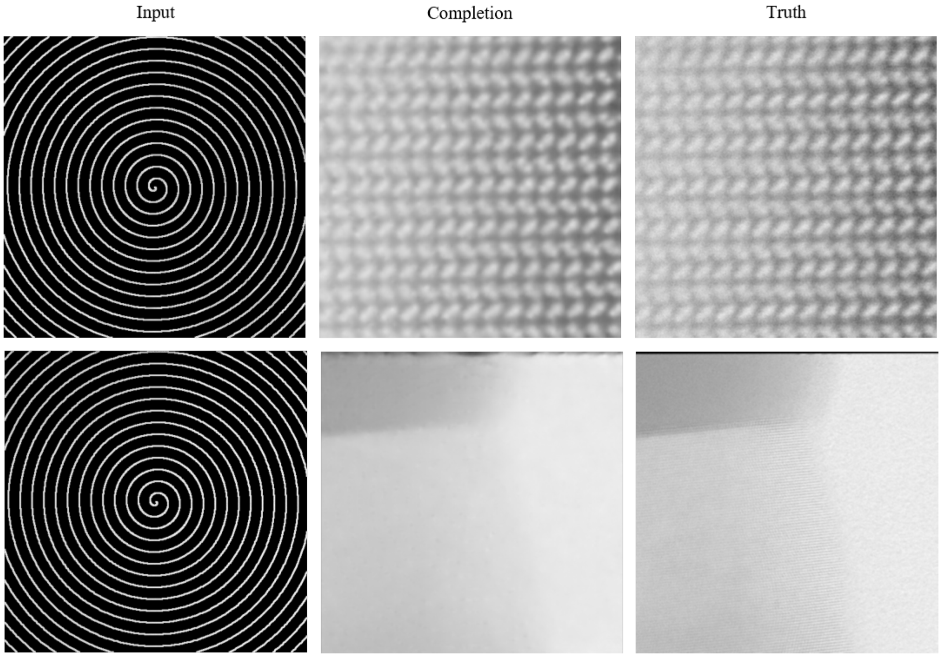


Figure 3. Neural network completions of 512× 512 scanning transmission electron microscopy images from 1/20 coverage blurred spiral scans.

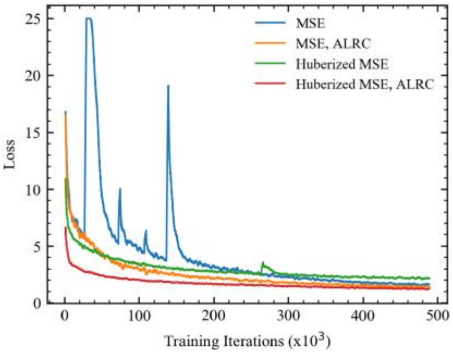


Figure 4. Outer generator losses show that ALRC and Huberization stabilize learning. ALRC lowers final mean squared error (MSE) and Huberized MSE losses and accelerates convergence. Learning curves are 2500 iteration boxcar averaged.

Initialization: Weights were initialized from a normal distribution with mean 0.00 and standard deviation 0.05. There are no biases.

Weight normalization: All generator weights are weight normalized [23] and a weight normalization initialization pass was performed after weight initialization. Following [23, 24], running mean-only batch normalization was applied to the output channels of every convolutional layer except the last. Channel means were tracked by exponential moving averages with decay rates of 0.99. Similar to [25], running mean-only batch normalization was frozen in the second half of training to improve stability.

Loss functions: The auxiliary inner generator trainer learns to generate half-size completions that minimize MSEs from half-size blurred ground truth STEM images. Meanwhile, the outer generator learns to produce full-size completions that minimize MSEs from blurred STEM images. All MSEs were multiplied by 200. The inner generator cooperates with the auxiliary inner generator trainer and outer generator.

Table 2. Means and standard deviations of 20 000 unclipped test set MSEs for STEM supersampling networks trained with various learning rate clipping algorithms and clipping hyperparameters, $n^{\uparrow}$ and $n^{\downarrow}$, above and below, respectively.

Algorithm	$n^{\downarrow}$	$n^{\uparrow}$	Mean	Std
Unchanged	∞	∞	0.95	1.33
ALRC	∞	3	0.89	1.68
ALRC	3	3	0.92	1.77
CLRC$^{(\downarrow)}$, ALRC$^{(\uparrow)}$	1	3	0.95	2.30
DALRC	3	3	0.93	1.57
DALRC	∞	2	0.89	1.51
DALRC	2	2	0.91	1.34
DALRC	1	2	0.91	1.54

To benchmark ALRC, we investigated training with MSEs, Huberized ($h = 1$) MSEs, MSEs with ALRC and Huberized ($h = 1$) MSEs with ALRC before Huberization. Training with both ALRC and Hubarization showcases the ability of ALRC to complement another loss function modification.

Learning policy: ADAM optimization [18] was used with a constant generator learning rate of 0.0003 and a first moment of the momentum decay rate, $\beta_1 = 0.9$, for 250 000 iterations. In the next 250 000 iterations, the learning rate and β_1 were linearly decayed in eight steps to zero and 0.5, respectively. The learning rate for the auxiliary inner generator trainer was two times the generator learning rate; β_1 were the same. All training was performed with batch size 1 due to the large model size needed to complete 512×512 scans.

Results: Outer generator losses in figure 4 show that ALRC and Huberization stabilize learning. Further, ALRC accelerates MSE and Huberized MSE convergence to lower losses. To be clear, learning policy was optimized for MSE training so direct loss comparison is uncharitable to ALRC.

Algorithm 2 Two-sided constant learning rate clipping (CLRC) to effective losses in $[L_{\text{min}}, L_{\text{max}}]$.

```
  Choose effective loss bounds, $L_{\text{min}}$ and $L_{\text{max}}$.
  While Training is not finished do
    If $L < L_{\text{min}}$ then
      $L_{\text{dyn}} \leftarrow \text{stop\_gradient}(L_{\text{min}}/L)L$
    else if $L > L_{\text{max}}$ then
      $L_{\text{dyn}} \leftarrow \text{stop\_gradient}(L_{\text{max}}/L)L$
    else
      $L_{\text{dyn}} \leftarrow L$
    end if
    Optimize network by back-propagating $L_{\text{dyn}}$.
end While
```

Algorithm 3 Two-sided doubly adaptive learning rate clipping (DALRC) of loss spikes. Sensible parameters are $\beta_1 = \beta^{\downarrow} = \beta^{\uparrow} = 0.999$, and $n^{\downarrow} = n^{\uparrow} = 2$.

```
  Initialize running means, $\mu_1$, $\mu^{\downarrow}$ and $\mu^{\uparrow}$, with decay rates, $\beta_1$, $\beta^{\downarrow}$ and $\beta^{\uparrow}$.
  Choose numbers, $n$, of standard deviations to clip to.
  While Training is not finisheddo
    Infer forward-propagation loss, $L$.
    $L_{\text{min}} \leftarrow \mu_1 - n^{\downarrow}\mu^{\downarrow}$
    $L_{\text{max}} \leftarrow \mu_1 + n^{\uparrow}\mu^{\uparrow}$
    if $L < L_{\text{min}}$ then
      $L_{\text{dyn}} \leftarrow \text{stop\_gradient}(L_{\text{min}}/L)L$
    else if $L > L_{\text{max}}$ then
      $L_{\text{dyn}} \leftarrow \text{stop\_gradient}(L_{\text{max}}/L)L$
    else
      $L_{\text{dyn}} \leftarrow L$
    end if
    Optimize network by back-propagating $L_{\text{dyn}}$.
    if $L > \mu_1$ then
      $\mu^{\uparrow} \leftarrow \beta^{\uparrow}\mu^{\uparrow} + (1 - \beta^{\uparrow})(L - \mu_1)$
    else if $L < \mu_1$ then
      $\mu^{\downarrow} \leftarrow \beta^{\downarrow}\mu^{\downarrow} + (1 - \beta^{\downarrow})(\mu_1 - L)$
    end if
    $\mu_1 \leftarrow \beta_1\mu_1 + (1 - \beta_1)L$
end While
```

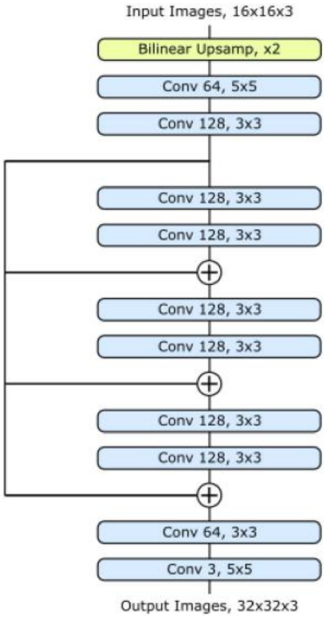


Figure 5. Convolutional image 2× supersampling network with three skip-2 residual blocks.

5. Experiments: ALRC variants

ALRC was developed to limit perturbations by loss spikes. Nevertheless, ALRC can also increase parameter perturbations for low losses, possibly improving performance on examples that an ANN is already good at. To investigate ALRC variants, we trained a generator to supersample STEM images to 512× 512 after nearest neighbour downsampling to 103× 103. Network architecture and learning protocols are the same as those for partial STEM in section 4, except training iterations are increased from 5×10^5 to 10^6.

Means and standard deviations of 20 000 unclipped test set MSEs for possible ALRC variants are tabulated in table 2. Variants include constant learning rate clipping (CLRC) in algorithm 2; where the effective loss is kept between constant values, and doubly adaptive learning rate clipping (DALRC) in algorithm 3; where moments above and below a running mean are tracked separately. ALRC has the lowest test set MSEs whereas DALRC has lower variance. Both ALRC and DLRC outperform no learning rate clipping for all tabulated hyperparameters and may be a promising starting point for future research on learning rate clipping.

6. Discussion

Taken together, our CIFAR-10 supersampling results show that ALRC improves stability and lowers losses for learning that would be destabilized by loss spikes and otherwise has little effect. Loss spikes are often encountered when training with high learning rates, high order loss functions or small batch sizes. For example, a moderate learning rate was used in MSE experiments so that losses did not spike enough to destabilize learning. In contrast, training at the same learning rate with quartic errors is unstable so ALRC stabilizes learning and lowers losses. Similar results are confirmed at unstably high learning rates, for partial STEM and for STEM supersampling, where ALRC stabilizes learning and lowers losses.

ALRC is designed to complement existing learning algorithms with new functionality. It is effective for any loss function or batch size and can be applied to any neural network trained with gradient descent. Our algorithm is also computationally inexpensive, requiring orders of magnitude fewer operations than other layers typically used in neural networks. As ALRC either stabilizes learning or has little effect, this means that it is suitable for routine application to arbitrary neural network training with gradient descent. In addition, we note that ALRC is a simple algorithm that has a clear effect on learning.

Nevertheless, ALRC can replace other learning algorithms in some situations. For instance, ALRC is a computationally inexpensive alternative to gradient clipping in high batch size training where gradient

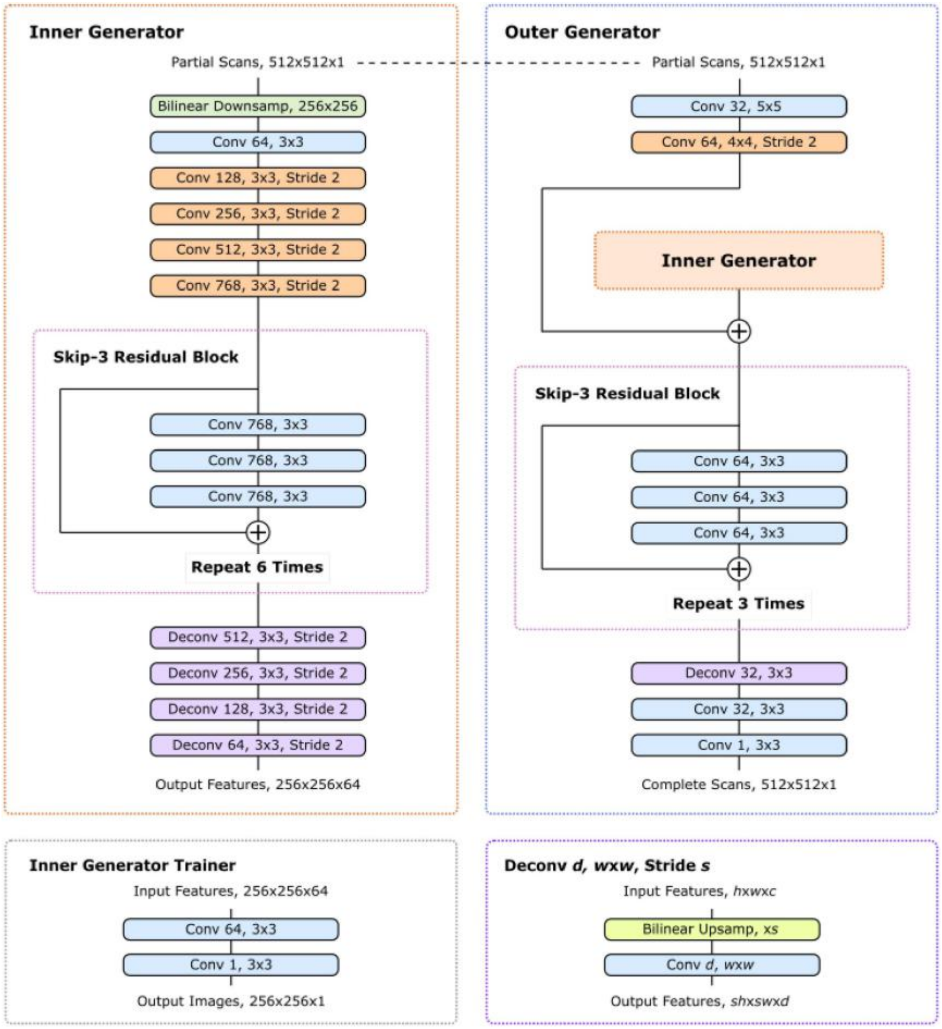


Figure 6. Two-stage generator that completes 512 × 512 micrographs from partial scans. A dashed line indicates that the same image is input to the inner and outer generator. Large scale features developed by the inner generator are locally enhanced by the outer generator and turned into images. An auxiliary inner generator trainer restores images from inner generator features to provide direct feedback.

clipping is being used to limit perturbations by loss spikes. However, it is not a direct replacement as ALRC preserves the distribution of backpropagated gradients whereas gradient clipping reduces large gradients. Instead, ALRC is designed to complement gradient clipping by limiting perturbations by large losses while gradient clipping modifies gradient distributions.

The implementation of ALRC in algorithm 1 is for positive losses. This avoids the need to introduce small constants to prevent divide-by-zero errors. Nevertheless, ALRC can support negative losses by using standard methods to prevent divide-by-zero errors. Alternatively, a constant can be added to losses to make them positive without affecting learning.

ALRC can also be extended to limit losses more than a number of standard deviations below their mean. This had no effect in our experiments. However, preemptively reducing loss spikes by clipping rewards between user-provided upper and lower bounds can improve reinforcement learning [26]. Subsequently, we suggest that clipping losses below their means did not improve learning because losses mainly spiked above their means; not below. Some partial STEM losses did spike below; however, they were mainly for blank or otherwise trivial completions.

7. Conclusions

We have developed ALRC to stabilize the training of ANNs by limiting backpropagated loss perturbations. Our experiments show that ALRC accelerates convergence and lowers losses for learning that would be destabilized by loss spikes and otherwise has little effect. Further, ALRC is computationally inexpensive, can be applied to any loss function or batch size, does not affect the distribution of backpropagated gradients and has a clear effect on learning. Overall, ALRC complements existing learning algorithms and can be routinely applied to arbitrary neural network training with gradient descent.

Data Availability

The data that support the findings of this study are openly available. Source code based on TensorFlow[27] is provided for CIFAR-10 supersampling[28] and partial STEM[29], and both CIFAR-10[12] and STEM[19] datasets are available. For additional information contact the corresponding author (J M E).

8. Network architecture

ANN architecture for CIFAR-10 experiments is shown in figure 5, and architecture for STEM partial scan and supersampling experiments is shown in figure 6. The components in our networks are

Bilinear Downsamp, *wxw*: This is an extension of linear interpolation in one dimension to two dimensions. It is used to downsample images to $w \times w$.

Bilinear Upsamp, x*s*: This is an extension of linear interpolation in one dimension to two dimensions. It is used to upsample images by a factor of s.

Conv *d*, *wxw*, Stride, *x*: Convolutional layer with a square kernel of width, w, that outputs d feature channels. If the stride is specified, convolutions are only applied to every xth spatial element of their input, rather than to every element. Striding is not applied depthwise.

$\oplus$: Circled plus signs indicate residual connections[30] where tensors are added together. Residual connections help reduce signal attenuation and allow networks to learn perturbative transformations more easily.

Acknowledgment

J M E and R B acknowledge EPSRC grant EP/N035437/1 for financial support. In addition, J M E acknowledges EPSRC Studentship 1917382.

ORCID iD

Jeffrey M Ede https://orcid.org/0000-0002-9358-5364

References

[1] Ruder S 2016 An overview of gradient descent optimization algorithms arXiv:1609.04747

[2] Zou D, Cao Y, Zhou D and Gu Q 2018 Stochastic gradient descent optimizes over-parameterized deep ReLU networks arXiv:1811.08888

[3] Pfülb B, Gepperth A, Abdullah S and Kilian A 2018 Catastrophic forgetting: still a problem for DNNs *Int. Conf. on Artificial Neural Networks* pp 487–97 Springer

[4] Boué L 2018 Deep learning for pedestrians: backpropagation in CNNs arXiv:1811.11987

[5] Qin Z, Yu F, Liu C and Chen X 2018 How convolutional neural network see the world-A survey of convolutional neural network visualization methods arXiv:1804.11191

[6] Sabour S, Frosst N and Hinton G E 2017 Dynamic routing between capsules *Advances in Neural Information Processing Systems* pp 3856–66

[7] Bengio Y and Pascanu R 2012 On the difficulty of training recurrent neural networks arXiv:1211.5063

[8] Mikolov T 2012 Statistical language models based on neural networks *PhD thesis* Brno University of Technology

[9] Huber P J 1964 Robust estimation of a location parameter *The Annals of Mathematical Statistics* pp 73–101

[10] Meyer G P 2019 An alternative probabilistic interpretation of the Huber loss arXiv:1911.02088

[11] Ioffe S and Szegedy C 2015 Batch normalization accelerating deep network training by reducing internal covariate shift arXiv:1502.03167

[12] Krizhevsky A, Nair V and Hinton G 2014 The CIFAR-10 dataset Online (www.cs.toronto.edu/ Kriz/Cifar.html) vol 55

[13] Krizhevsky A and Hinton G 2009 Learning multiple layers of features from tiny images *Technical Report TR-2009* University of Toronto

[14] McCann M T, Jin K H and Unser M 2017 Convolutional neural networks for inverse problems in imaging: A review *IEEE Signal Process. Mag.* **34** 85–95

[15] Krizhevsky A, Sutskever I and Hinton G E 2012 ImageNet classification with deep convolutional neural networks *Advances in Neural Information Processing Systems* pp 1097–105

[16] Nair V and Hinton G E 2010 Rectified linear units improve restricted Boltzmann machines *in Proc. of the 27th Int. Conf. on Machine Learning (ICML-10)* pp 807–14
[17] Glorot X and Bengio Y 2010 Understanding the difficulty of training deep feedforward neural networks *Proc. of the Thirteenth Int. Conf. on Artificial Intelligence and Statistics* pp 249–56
[18] Kingma D P and Ba J 2014 ADAM: A method for stochastic optimization arXiv:1412.6980
[19] Ede J M and STEM Crops Dataset 2019 Online (https://warwick.ac.uk/fac/sci/physics/research/condensedmatt/microscopy/research/machinelearning)
[20] Ede J M and Beanland R 2020 Partial scanning transmission electron microscopy with deep learning arXiv:1910.10467
[21] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V and Rabinovich A 2015 Going deeper with convolutions *Proc. of the Conf. on Computer Vision and Pattern Recognition* pp 1–9
[22] Szegedy C, Vanhoucke V, Ioffe S, Shlens J and Wojna Z 2016 Rethinking the inception architecture for computer vision *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition* pp 2818–26
[23] Salimans T and Kingma D P 2016 Weight normalization: A simple reparameterization to accelerate training of deep neural networks *Advances in Neural Information Processing Systems* pp 901–9
[24] Hoffer E, Banner R, Golan I and Soudry D 2018 Norm matters: efficient and accurate normalization schemes in deep networks *Advances in Neural Information Processing Systems* pp 2160–70
[25] Chen L-C, Papandreou G, Schroff F and Adam H 2017 Rethinking atrous convolution for semantic image segmentation arXiv:1706.05587
[26] Mnih V *et al et al* 2015 Human-Level control through deep reinforcement learning *Nature* **518** 529
[27] Abadi M *et al et al* 2016 Tensor flow: A system for large-scale machine learning. *OSDI* **16** 265–83
[28] Ede J M and ALRC 2020 Online: (https://github.com/Jeffrey-Ede/ALRC)
[29] Ede J M and Partial STEM 2020 Online: (https://github.com/Jeffrey-Ede/partial-STEM)
[30] He K, Zhang X, Ren S and Sun J 2016 Deep residual learning for image recognition *Proc. of the Conf. on Computer Vision and Pattern Recognition* pp 770–8

3.2 Amendments and Corrections

There are amendments or corrections to the paper[3] covered by this chapter.

Location: Page 3, image in fig. 1.
Change: A title above the top two graphs is cut off. The missing title said "With Adaptive Learning Rate Clipping", and is visible in our preprint[16].

Location: Last paragraph starting on page 7.
Change: "...inexpensive alternative to gradient clipping in high batch size training where..." should say "...inexpensive alternative to gradient clipping where...".

3.3 Reflection

This ancillary chapter covers my paper titled "Adaptive Learning Rate Clipping Stabilizes Learning"[3] and associated research outputs[16,17]. The ALRC algorithm was developed to prevent loss spikes destabilizing training of DNNs for partial STEM[4] (ch. 4). To fit the partial STEM ANN in GPU memory, it was trained with a batch size of 1. However, using a small batch size results in occasional loss spikes, which meant that it was sometimes necessary to repeat training to compare performance with earlier experiments where learning had not been destabilized by loss spikes. I expected that I could adjust training hyperparameters to stabilize learning; however, I had optimized the hyperparameters and training was usually fine. Thus, I developed ALRC to prevent loss spikes from destabilizing learning. Initially, ALRC was included as an appendix in the first version of the partial STEM preprint[18]. However, ALRC was so effective that I continued to investigate. Eventually, there were too many ALRC experiments to comfortably fit in an appendix of the partial STEM paper, so I separated ALRC into its own paper.

There are variety of alternatives to ALRC that can stabilize learning. A popular alternative is training with Huberized losses[181,182],

$$\text{Huber}(L) = \min(L, (\lambda L)^{1/2}) \,, \tag{3.1}$$

where L is a loss and λ is a training hyperparameter. However, I found that Huberized learning continued to be destabilized by loss spikes. I also considered gradient clipping[183–185]. However, my DNNs for partial STEM have many millions of trainable parameters, so computational requirements for gradient clipping are millions of times higher than applying ALRC to losses. Similarly, rectified ADAM[186] (RADAM), can stabilize learning by decreasing trainable parameter learning rates if adaptive learning rates of an ADAM[187] optimizer have high variance. However, computational requirements of RADAM are also often millions of times higher than ALRC as RADAM adapts adaptive learning rates for every trainable parameter.

Overall, I think that ALRC merits further investigation. ALRC is computationally inexpensive, can be applied to any loss function, and appears to either stabilize learning or have no significant effect. Further, ALRC can often readily improve ANN training that would otherwise be destabilized loss spikes. However, I suspect that ALRC may slightly decrease performance where learning is not destabilized by loss spikes as ALRC modifies training losses. In addition, I have only investigated applications of ALRC to mean square and quartic errors per training example of deep convolutional neural networks (CNNs). Applying ALRC to losses for individual pixels of CNN outputs or to losses at each step of a recurrent neural network (RNN) may further improve performance. Encouragingly, my initial experiments with ALRC variants[3] show that a variety approaches improve training that would otherwise be destabilized by loss spikes.

Chapter 4

Partial Scanning Transmission Electron Microscopy with Deep Learning

4.1 Scientific Paper

This chapter covers the following paper[4] and its supplementary information[10].

> J. M. Ede and R. Beanland. Partial Scanning transmission Electron Microscopy with Deep Learning. *Scientific Reports*, 10(1):1–10, 2020
>
> J. M. Ede. Supplementary Information: Partial Scanning Transmission Electron Microscopy with Deep Learning. Online: `https://static-content.springer.com/esm/art%3A10.1038%2Fs41598-020-65261-0/MediaObjects/41598_2020_65261_MOESM1_ESM.pdf`, 2020

SCIENTIFIC REPORTS
natureresearch

Check for updates

OPEN

Partial Scanning Transmission Electron Microscopy with Deep Learning

Jeffrey M. Ede✉ & Richard Beanland

Compressed sensing algorithms are used to decrease electron microscope scan time and electron beam exposure with minimal information loss. Following successful applications of deep learning to compressed sensing, we have developed a two-stage multiscale generative adversarial neural network to complete realistic 512 × 512 scanning transmission electron micrographs from spiral, jittered gridlike, and other partial scans. For spiral scans and mean squared error based pre-training, this enables electron beam coverage to be decreased by 17.9× with a 3.8% test set root mean squared intensity error, and by 87.0× with a 6.2% error. Our generator networks are trained on partial scans created from a new dataset of 16227 scanning transmission electron micrographs. High performance is achieved with adaptive learning rate clipping of loss spikes and an auxiliary trainer network. Our source code, new dataset, and pre-trained models are publicly available.

Aberration corrected scanning transmission electron microscopy (STEM) can achieve imaging resolutions below 0.1 nm, and locate atom columns with pm precision[1,2]. Nonetheless, the high current density of electron probes produces radiation damage in many materials, limiting the range and type of investigations that can be performed[3,4]. A number of strategies to minimize beam damage have been proposed, including dose fractionation[5] and a variety of sparse data collection methods[6]. Perhaps the most intensively investigated approach to the latter is sampling a random subset of pixels, followed by reconstruction using an inpainting algorithm[3,6–10]. Poisson random sampling of pixels is optimal for reconstruction by compressed sensing algorithms[11]. However, random sampling exceeds the design parameters of standard electron beam deflection systems, and can only be performed by collecting data slowly[12,13], or with the addition of a fast deflection or blanking system[3,14].

Sparse data collection methods that are more compatible with conventional beam deflection systems have also been investigated. For example, maintaining a linear fast scan deflection whilst using a widely-spaced slow scan axis with some small random 'jitter'[9,12]. However, even small jumps in electron beam position can lead to a significant difference between nominal and actual beam positions in a fast scan. Such jumps can be avoided by driving functions with continuous derivatives, such as those for spiral and Lissajous scan paths[3,13,15,16]. Sang[13,16] considered a variety of scans including Archimedes and Fermat spirals, and scans with constant angular or linear displacements, by driving electron beam deflectors with a field-programmable gate array (FPGA) based system. Spirals with constant angular velocity place the least demand on electron beam deflectors. However, dwell times, and therefore electron dose, decreases with radius. Conversely, spirals created with constant spatial speeds are prone to systematic image distortions due to lags in deflector responses. In practice, fixed doses are preferable as they simplify visual inspection and limit the dose dependence of STEM noise[17].

Deep learning has a history of successful applications to image infilling, including image completion[18], irregular gap infilling[19] and supersampling[20]. This has motivated applications of deep learning to the completion of sparse, or 'partial', scans, including supersampling of scanning electron microscopy[21] (SEM) and STEM images[22,23]. Where pre-trained models are unavailable for transfer learning[24], artificial neural networks (ANNs) are typically trained, validated and tested with large, carefully partitioned machine learning datasets[25,26] so that they are robust to general use. In practice, this often requires at least a few thousand examples. Indeed, standard machine learning datasets such as CIFAR-10[27,28], MNIST[29], and ImageNet[30] contain tens of thousands or millions of examples. To train an ANN to complete STEM images from partial scans, an ideal dataset might consist of a large number of pairs of partial scans and corresponding high-quality, low noise images, taken with an aberration-corrected STEM. To our knowledge, such a dataset does not exist. As a result, we have collated a new dataset of STEM raster scans from which partial scans can be selected. Selecting partial scans from full scans is

University of Warwick, Department of Physics, Coventry, CV4 7AL, UK. ✉e-mail: j.m.ede@warwick.ac.uk

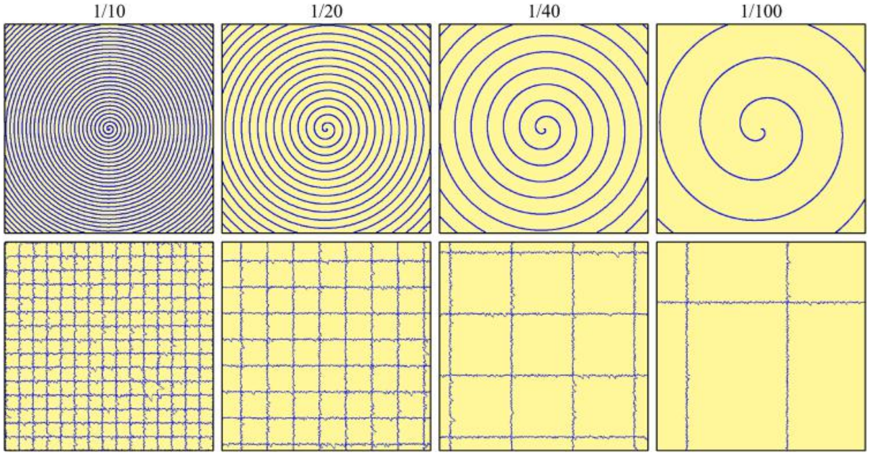


Figure 1. Examples of Archimedes spiral (top) and jittered gridlike (bottom) 512 × 512 partial scan paths for 1/10, 1/20, 1/40, and 1/100 px coverage.

less expensive than collecting image pairs, and individual pixels selected from experimental images have realistic noise characteristics.

Examples of spiral and jittered gridlike partial scans investigated in this paper are shown in Fig. 1. Continuous spiral scan paths that extend to image corners cannot be created by conventional scan systems without going over image edges. However, such a spiral can be cropped from a spiral with radius at least $2^{-1/2}$ times the minimum image side, at the cost of increased scan time and electron beam damage to the surrounding material. We use Archimedes spirals, where $r \propto \theta$, and r and θ are polar radius and angle coordinates, as these spirals have the most uniform spatial coverage. Jittered gridlike scans would also be difficult to produce with a conventional system, which would suffer variations in dose and distortions due to limited beam deflector response. Nevertheless, these idealized scan paths serve as useful inputs to demonstrate the capabilities of our approach. We expect that other scan paths could be used with similar results.

We fine-tune our ANNs as part of generative adversarial networks[31] (GANs) to complete realistic images from partial scans. A GAN consists of sets of generators and discriminators that play an adversarial game. Generators learn to produce outputs that look realistic to discriminators, while discriminators learn to distinguish between real and generated examples. Limitedly, discriminators only assess whether outputs look realistic; not if they are correct. This can result in a neural network only generating a subset of outputs, referred to as mode collapse[32]. To counter this issue, generator learning can be conditioned on an additional distance between generated and true images[33]. Meaningful distances can be hand-crafted or learned automatically by considering differences between features imagined by discriminators for real and generated images[34,35].

Training

In this section we introduce a new STEM images dataset for machine learning, describe how partial scans were selected from images in our data pipeline, and outline ANN architecture and learning policy. Detailed ANN architecture, learning policy, and experiments are provided as Supplementary Information, and source code is available[36].

Data pipeline.

To create partial scan examples, we collated a new dataset containing 16227 32-bit floating point STEM images collected with a JEOL ARM200F atomic resolution electron microscope. Individual micrographs were saved to University of Warwick data servers by dozens of scientists working on hundreds of projects as Gatan Microscopy Suite[37] generated dm3 or dm4 files. As a result, our dataset has a diverse constitution. Atom columns are visible in two-thirds of STEM images, with most signals imaged at several times their Nyquist rates[38], and similar proportions of images are bright and dark field. The other third of images are at magnifications too low for atomic resolution, or are of amorphous materials. Importantly, our dataset contains noisy images, incomplete scans and other low-quality images that would not normally be published. This ensures that ANNs trained on our dataset are robust to general use. The Digital Micrograph image format is rarely used outside the microscopy community. As a result, data has been transferred to the widely supported TIFF[39] file format in our publicly available dataset[40,41].

Micrographs were split into 12170 training, 1622 validation, and 2435 test set examples. Each subset was collected by a different subset of scientists and has different characteristics. As a result, unseen validation and test sets can be used to quantify the ability of a trained network to generalize. To reduce data read times, each micrograph was split into non-overlapping 512 × 512 sub-images, referred to as 'crops', producing 110933 training, 21259 validation and 28877 test set crops. For convenience, our crops dataset is also available[40,41]. Each crop, I, was processed in our data pipeline by replacing non-finite electron counts, i.e. NaN and $\pm\infty$, with zeros. Crops were then linearly transformed to have intensities $I_\text{N} \in [-1, 1]$, except for uniform crops satisfying

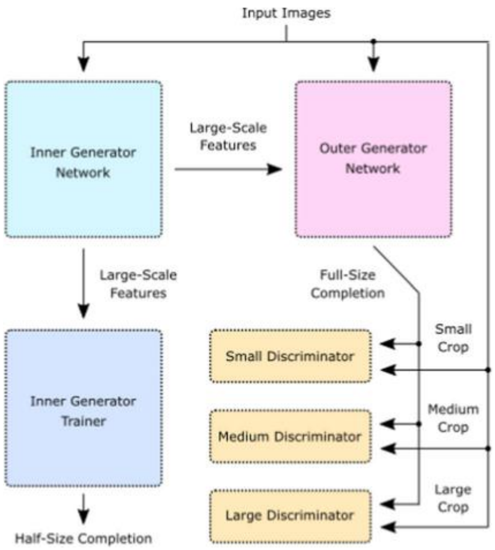


Figure 2. Simplified multiscale generative adversarial network. An inner generator produces large-scale features from inputs. These are mapped to half-size completions by a trainer network and recombined with the input to generate full-size completions by an outer generator. Multiple discriminators assess multiscale crops from input images and full-size completions. This figure was created with Inkscape[83].

$\max(I) - \min(I) < 10^{-6}$ where we set $I_\text{N} = 0$ everywhere. Finally, each crop was subject to a random combination of flips and 90° rotations to augment the dataset by a factor of eight.

Partial scans, I_scan, were selected from raster scan crops, I_N, by multiplication with a binary mask Φ_path,

$$I_\text{scan} = \Phi_\text{path} I_\text{N}, \tag{1}$$

where $\Phi_\text{path} = 1$ on a scan path, and $\Phi_\text{path} = 0$ otherwise. Raster scans are sampled at a rectangular lattice of discrete locations, so a subset of raster scan pixels are experimental measurements. In addition, although electron probe position error characteristics may differ for partial and raster scans, typical position errors are small[42,43]. As a result, we expect that partial scans selected from raster scans with binary masks are realistic.

We also selected partial scans with blurred masks to simulate varying dwell times and noise characteristics. These difficulties are encountered in incoherent STEM[44,45], where STEM illumination is detected by a transmission electron microscopy (TEM) camera. For simplicity, we created non-physical noise by multiplying I_scan with $\eta(\Phi_\text{path}) = \Phi_\text{path} + (1 - \Phi_\text{path})U$, where U is a uniform random variate distributed in $[0, 2)$. ANNs are able to generalize[46,47], so we expect similar results for other noise characteristics. A binary mask, with values in $\{0, 1\}$, is a special case where no noise is applied i.e. $\eta(1) = 1$, and $\Phi_\text{path} = 0$ is not traversed. Performance is reported for both binary and blurred masks.

The noise characteristics in our new STEM images dataset vary. This is problematic for mean squared error (MSE) based ANN training losses, as differences are higher for crops with higher noise. In effect, this would increase the importance of noisy images in the dataset, even if they are not more representative. Although adaptive ANN optimizers that divide parameter learning rates by gradient sizes[48] can partially mitigate weighting by varying noise levels, this restricts training to a batch size of 1 and limits momentum. Consequently, we low-passed filtered ground truth images, I_N, to I_blur by a 5×5 symmetric Gaussian kernel with a 2.5 px standard deviation, to calculate MSEs for ANN outputs.

Network architecture. To generate realistic images, we developed a multiscale conditional GAN with TensorFlow[49]. Our network can be partitioned into the six convolutional[50,51] subnetworks shown in Fig. 2: an inner generator, G_inner, outer generator, G_outer, inner generator trainer, T, and small, medium and large scale discriminators, D_1, D_2 and D_3. We refer to the compound network $G(I_\text{scan}) = G_\text{outer}(G_\text{inner}(I_\text{scan}), I_\text{scan})$ as the generator, and to $D = \{D_1, D_2, D_3\}$ as the multiscale discriminator. The generator is the only network needed for inference.

Following recent work on high-resolution conditional GANs[34], we use two generator subnetworks. The inner generator produces large scale features from partial scans bilinearly downsampled from 512×512 to 256×256. These features are then combined with inputs embedded by the outer generator to output full-size completions. Following Inception[52,53], we introduce an auxiliary trainer network that cooperates with the inner generator to output 256×256 completions. This acts as a regularization mechanism, and provides a more direct path for gradients to backpropagate to the inner generator. To more efficiently utilize initial generator convolutions, partial scans selected with a binary mask are nearest neighbour infilled before being input to the generator.

Multiscale discriminators examine real and generated STEM images to predict whether they are real or generated, adapting to the generator as it learns. Each discriminator assesses different-sized crops selected from 512 × 512 images, with sizes 70 × 70, 140 × 140 or 280 × 280. After selection, crops are bilinearly downsampled to 70 × 70 before discriminator convolutions. Typically, discriminators are applied at fractions of the full image size[34] e.g. $512/2^2$, $512/2^1$ and $512/2^0$. However, we found that discriminators that downsample large fields of view to 70 × 70 are less sensitive to high-frequency STEM noise characteristics. Processing fixed size image regions with multiple discriminators has been proposed[54] to decrease computation for large images, and extended to multiple region sizes[34]. However, applying discriminators to arrays of non-overlapping image patches[55] results in periodic artefacts[34] that are often corrected by larger-scale discriminators. To avoid these artefacts and reduce computation, we apply discriminators to randomly selected regions at each spatial scale.

Learning policy. Training has two halves. In the non-adversarial first half, the generator and auxiliary trainer cooperate to minimize mean squared errors (MSEs). This is followed by an optional second half of training, where the generator is fine-tuned as part of a GAN to produce realistic images. Our ANNs are trained by ADAM[56] optimized stochastic gradient descent[48,57] for up to 2×10^6 iterations, which takes a few days with an Nvidia GTX 1080 Ti GPU and an i7-6700 CPU. The objectives of each ANN are codified by their loss functions.

In the non-adversarial first half of training, the generator, G, learns to minimize the MSE based loss

$$L_{\text{MSE}} = \text{ALRC}(\lambda_{\text{cond}}\text{MSE}(G(I_{\text{scan}}), I_{\text{blur}})), \tag{2}$$

where $\lambda_{\text{cond}} = 200$, and adaptive learning rate clipping[58] (ALRC) is important to prevent high loss spikes from destabilizing learning. Experiments with and without ALRC are in Supplementary Information. To compensate for varying noise levels, ground truth images were blurred by a 5 × 5 symmetric Gaussian kernel with a 2.5 px standard deviation. In addition, the inner generator, G_{inner}, cooperates with the auxiliary trainer, T, to minimize

$$L_{\text{aux}} = \text{ALRC}(\lambda_{\text{trainer}}\text{MSE}(T(G_{\text{inner}}(I_{\text{scan}}^{\text{half}})), I_{\text{blur}}^{\text{half}}), \tag{3}$$

where $\lambda_{\text{trainer}} = 200$, and $I_{\text{scan}}^{\text{half}}$ and $I_{\text{blur}}^{\text{half}}$ are 256 × 256 inputs bilinearly downsampled from I_{scan} and I_{blur}, respectively.

In the optional adversarial second half of training, we use $N = 3$ discriminator scales with numbers, N_1, N_2 and N_3, of discriminators, D_1, D_2 and D_3, respectively. There many popular GAN loss functions and regularization mechanisms[59,60]. In this paper, we use spectral normalization[61] with squared difference losses[62] for the discriminators,

$$L_D = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{N_i}[D_i(G(I_{\text{scan}}))^2 + (D_i(I_N) - 1)^2], \tag{4}$$

where discriminators try to predict 1 for real images and 0 for generated images. We found that $N_1 = N_2 = N_3 = 1$ is sufficient to train the generator to produce realistic images. However, higher performance might be achieved with more discriminators e.g. 2 large, 8 medium and 32 small discriminators. The generator learns to minimize the adversarial squared difference loss,

$$L_{\text{adv}} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{N_i}D_i(G(I_{\text{scan}}) - 1)^2, \tag{5}$$

by outputting completions that look realistic to discriminators.

Discriminators only assess the realism of generated images; not if they are correct. To the lift degeneracy and prevent mode collapse, we condition adversarial training on non-adversarial losses. The total generator loss is

$$L_G = \lambda_{\text{adv}}L_{\text{adv}} + L_{\text{MSE}} + \lambda_{\text{aux}}L_{\text{aux}}, \tag{6}$$

where we found that $\lambda_{\text{aux}} = 1$ and $\lambda_{\text{adv}} = 5$ is effective. We also tried conditioning the second half of training on differences between discriminator imagination[34,35]. However, we found that MSE guidance converges to slightly lower MSEs and similar structural similarity indexes[63] for STEM images.

Performance

To showcase ANN performance, example applications of adversarial and non-adversarial generators to 1/20 px coverage partial STEM completion are shown in Fig. 3. Adversarial completions have more realistic high-frequency spatial information and structure, and are less blurry than non-adversarial completions. Systematic spatial variation is also less noticeable for adversarial completions. For example, higher detail along spiral paths, where errors are lower, can be seen in the bottom two rows of Fig. 3 for non-adversarial completions. Inference only requires a generator, so inference times are the same for adversarial and non-adversarial completions. Single image inference time during training is 45 ms with an Nvidia GTX 1080 Ti GPU, which is fast enough for live partial scan completion.

In practice, 1/20 px scan coverage is sufficient to complete most spiral scans. However, generators cannot reliably complete micrographs with unpredictable structure in regions where there is no coverage. This is demonstrated by example applications of non-adversarial generators to 1/20 px coverage spiral and gridlike partial scans

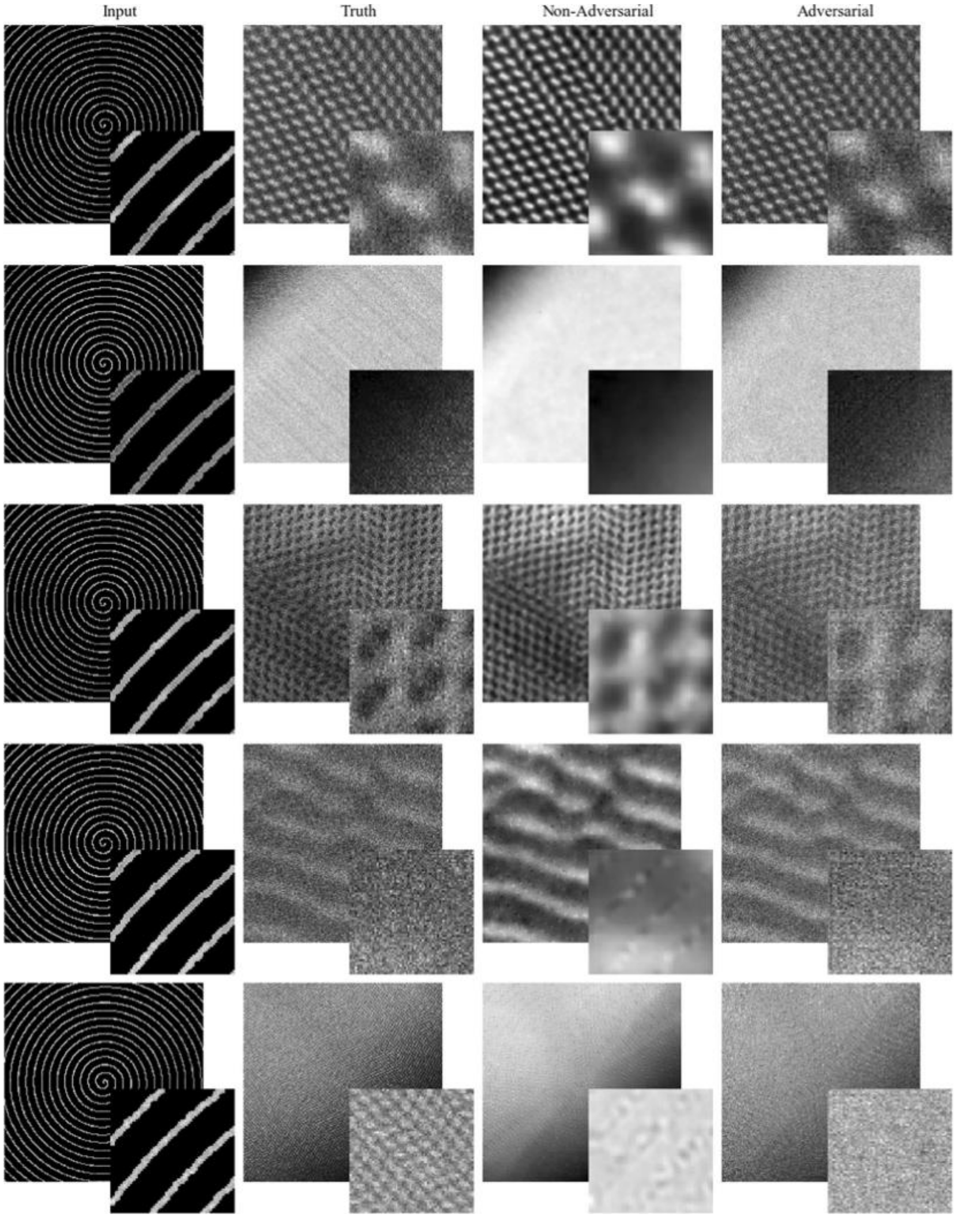


Figure 3. Adversarial and non-adversarial completions for 512 × 512 test set 1/20 px coverage blurred spiral scan inputs. Adversarial completions have realistic noise characteristics and structure whereas non-adversarial completions are blurry. The bottom row shows a failure case where detail is too fine for the generator to resolve. Enlarged 64 × 64 regions from the top left of each image are inset to ease comparison, and the bottom two rows show non-adversarial generators outputting more detailed features nearer scan paths.

in Fig. 4. Most noticeably, a generator invents a missing atom at a gap in gridlike scan coverage. Spiral scans have lower errors than gridlike scans as spirals have smaller gaps between coverage. Additional sheets of examples for spiral scans selected with binary masks are provided for scan coverages between 1/17.9 px and 1/87.0 px as Supplementary Information.

To characterize generator performance, MSEs for output pixels are shown in Fig. 5. Errors were calculated for 20000 test set 1/20 px coverage spiral scans selected with blurred masks. Errors systematically increase with increasing distance from paths for non-adversarial training, and are less structured for adversarial training. Similar to other generators[23,64], errors are also higher near the edges of non-adversarial outputs where there is less information. We tried various approaches to decrease non-adversarial systematic error variation by modifying loss functions. For examples: by ALRC; multiplying pixel losses by their running means; by ALRC and

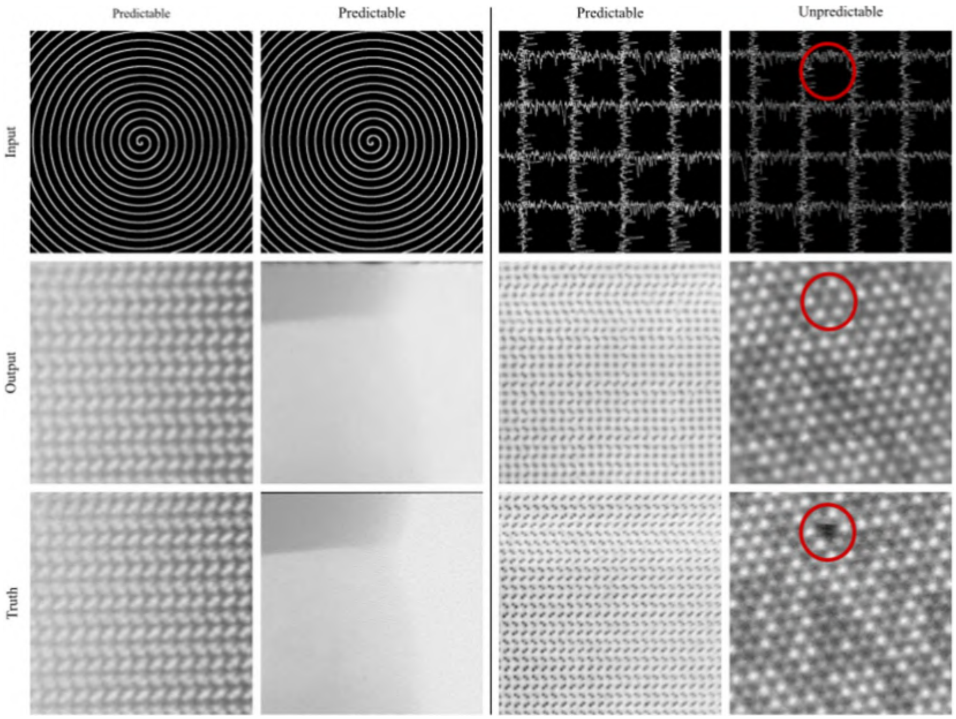


Figure 4. Non-adversarial generator outputs for 512 × 512 1/20 px coverage blurred spiral and gridlike scan inputs. Images with predictable patterns or structure are accurately completed. Circles accentuate that generators cannot reliably complete unpredictable images where there is no information. This figure was created with Inkscape[83].

multiplying pixel losses by their running means; and by ALRC and multiplying pixel losses by final mean losses of a trained network. However, we found that systematic errors are similar for all variants. This is a limitation of partial STEM as information decreases with increasing distance from scan paths. Adversarial completions also exhibit systematic errors that vary with distance from spiral paths. However, spiral variation is dominated by other, less structured, spatial error variation. Errors are higher for adversarial training than for non-adversarial training as GANs complete images with realistic noise characteristics.

Spiral path test set intensity errors are shown in Fig. 6a, and decrease with increasing coverage for binary masks. Test set errors are also presented for deep learning supersampling[23] (DLSS) as they are the only results that are directly comparable. DLSS is an alternative approach to compressed sensing where STEM images are completed from a sublattice of probing locations. Both DLSS and partial STEM results are for the same neural network architecture, learning policy and training dataset. Results depend on datasets, so using the same dataset is essential for quantitative comparison. We find that DLSS errors are lower than spiral errors at all coverages. In addition, spiral errors exponentially increase above DLSS errors at low coverages where minimum distances from spiral paths increase. Although this comparison may appear unfavourable for partial STEM, we expect that this is a limitation of training signals being imaged at several times their Nyquist rates.

Distributions of 20000 spiral path test set root mean squared (RMS) intensity errors for spiral data in Fig. 6a are shown in Fig. 6b. The coverages listed in Fig. 6 are for infinite spiral paths with 1/16, 1/25, 1/36, 1/49, 1/64, 1/81, and 1/100 px coverage after paths are cut by image boundaries; changing coverage. All distributions have a similar peak near an RMS error of 0.04, suggesting that generator performance remains similar for a portion of images as coverage is varied. As coverage decreases, the portion of errors above the peak increases as generators have difficulty with more images. In addition, there is a small peak close to zero for blank or otherwise trivial completions.

Discussion

Partial STEM can decrease scan coverage and total electron electron dose by 10–100× with 3–6% test set RMS errors. These errors are small compared to typical STEM noise. Decreased electron dose will enable new STEM applications to beam-sensitive materials, including organic crystals[65], metal-organic frameworks[66], nanotubes[67], and nanoparticle dispersions[68]. Partial STEM can also decrease scan times in proportion to decreased coverage. This will enable increased temporal resolution of dynamic materials, including polar nanoregions in relaxor ferroelectrics[69,70], atom motion[71], nanoparticle nucleation[72], and material interface dynamics[73]. In addition, faster scans can reduce delay for experimenters, decreasing microscope time. Partial STEM can also be a starting point for algorithms that process STEM images e.g. to find and interpret atomic positions[74].

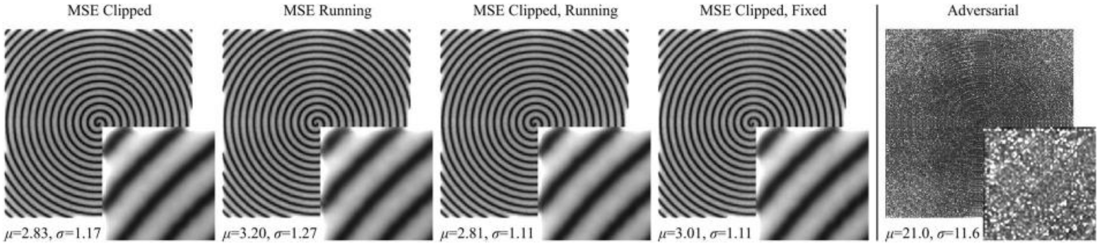


Figure 5. Generator mean squared errors (MSEs) at each output pixel for 20000 512 × 512 1/20 px coverage test set images. Systematic errors are lower near spiral paths for variants of MSE training, and are less structured for adversarial training. Means, μ, and standard deviations, σ, of all pixels in each image are much higher for adversarial outputs. Enlarged 64 × 64 regions from the top left of each image are inset to ease comparison, and to show that systematic errors for MSE training are higher near output edges.

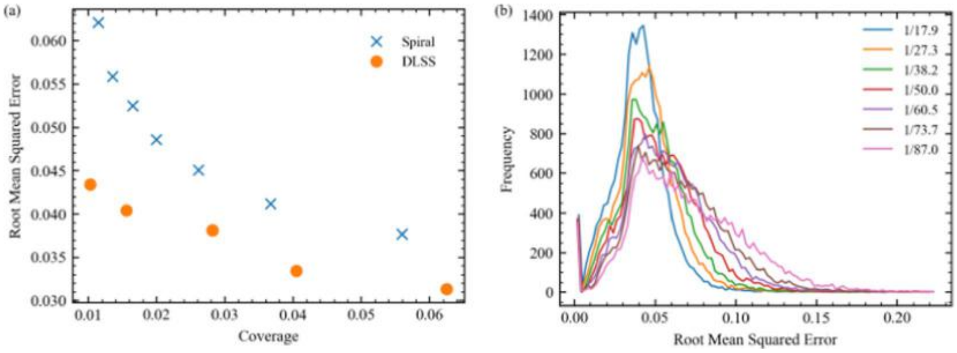


Figure 6. Test set root mean squared (RMS) intensity errors for spiral scans in $[0,\ 1]$ selected with binary masks. (**a**) RMS errors decrease with increasing electron probe coverage, and are higher than deep learning supersampling[23] (DLSS) errors. (**b**) Frequency distributions of 20000 test set RMS errors for 100 bins in $[0,\ 0.224]$ and scan coverages in the legend.

Our generators are trained for fixed coverages and 512 × 512 inputs. However, recent research has introduced loss function modifications that can be used to train a single generator for multiple coverages with minimal performance loss[23]. Using a single GAN improves portability as each of our GANs requires 1.3 GB of storage space with 32 bit model parameters, and limits technical debt that may accompany a large number of models. Although our generator input sizes are fixed, they can be tiled across larger images; potentially processing tiles in a single batch for computational efficiency. To reduce higher errors at the edge of generator outputs, tiles can be overlapped so that edges may be discarded[64]. Smaller images could be padded. Alternatively, dedicated generators can be trained for other output sizes.

There is an effectively infinite number of possible partial scan paths for 512 × 512 STEM images. In this paper, we focus on spiral and gridlike partial scans. For a fixed coverage, we find that the most effective method to decrease errors is to minimize maximum distances from input information. The less information there is about an output region, the more information that needs to be extrapolated, and the higher the error. For example, we find that errors are lower for spiral scans than gridlike scans as maximum distances from input information are lower. Really, the optimal scan shape is not static: It is specific to a given image and generator architecture. As a result, we are actively developing an intelligent partial scan system that adapts to inputs as they are scanned.

Partial STEM has a number of limitations relative to DLSS. For a start, partial STEM may require a custom scan system. Even if a scan system supports or can be reprogrammed to support custom scan paths, it may be insufficiently responsive. In contrast, DLSS can be applied as a postprocessing step without hardware modification. Another limitation of partial STEM is that errors increase with increasing distance from scan paths. Distances from continuous scan paths cannot be decreased without increasing coverage. Finally, most features in our new STEM crops dataset are sampled at several times their Nyquist rates. Electron microscopists often record images above minimum sufficient resolutions and intensities to ease visual inspection and limit the effects of drift[75], shot[17], and other noise. This means that a DLSS lattice can still access most high frequency information in our dataset.

Test set DLSS errors are lower than partial STEM errors for the same architecture and learning policy. However, this is not conclusive as generators were trained for a few days; rather than until validation errors diverged from training errors. For example, we expect that spirals need more training iterations than DLSS as nearest neighbour infilled spiral regions have varying shapes, whereas infilled regions of DLSS grids are square. In addition, limited high frequency information in training data limits one of the key strengths of partial STEM that DLSS lacks: access to high-frequency information from neighbouring pixels. As a result, we expect that partial STEM performance would be higher for signals imaged closer to their Nyquist rates.

To generate realistic images, we fine-tuned partial STEM generators as part of GANs. GANs generate images with more realistic high-frequency spatial components and structure than MSE training. However, GANs focus on semantics; rather than intensity differences. This means that although adversarial completions have realistic characteristics, such as high-frequency noise, individual pixel values differ from true values. GANs can also be difficult to train[76,77], and training requires additional computation. Nevertheless, inference time is the same for adversarial and non-adversarial generators after training.

Encouragingly, ANNs are universal approximators[78] that can represent[79] the optimal mapping from partial scans with arbitrary accuracy. This overcomes the limitations of traditional algorithms where performance is fixed. If ANN performance is insufficient or surpassed by another method, training or development can be continued to achieve higher performance. Indeed, validation errors did not diverge from training errors during our experiments, so we are presenting lower bounds for performance. In this paper, we compare spiral STEM performance against DLSS. It is the only method that we can rigorously and quantitatively compare against as it used the same test set data. This yielded a new insight into how signals being imaged above their Nyquist rates may affect performance discussed two paragraphs earlier, and highlights the importance of standardized datasets like our new STEM images dataset. As machine learning becomes more established in the electron microscopy community, we hope that standardized datasets will also become established to standardize performance benchmarks.

Detailed neural network architecture, learning policy, experiments, and additional sheets of examples are provided as Supplementary Information. Further improvements might be made with AdaNet[80], Ludwig[81], or other automatic machine learning[82] algorithms, and we encourage further development. In this spirit, we have made our source code[36], a new dataset containing 16227 STEM images[40,41], and pre-trained models publicly available. For convenience, new datasets containing 161069 non-overlapping 512 × 512 crops from STEM images used for training, and 19769 antialiased 96 × 96 area downsampled STEM images created for faster ANN development, are also available.

Conclusions

Partial STEM with deep learning can decrease electron dose and scan time by over an order of magnitude with minimal information loss. In addition, realistic STEM images can be completed by fine-tuning generators as part of a GAN. Detailed MSE characteristics are provided for multiple coverages, including MSEs per output pixel for 1/20 px coverage spiral scans. Partial STEM will enable new beam sensitive applications, so we have made our source code, new STEM dataset, pre-trained models, and details of experiments available to encourage further investigation. High performance is achieved by the introduction of an auxiliary trainer network, and adaptive learning rate clipping of high losses. We expect our results to be generalizable to SEM and other scan systems.

Data availability

New STEM datasets are available on our publicly accessible dataserver[40,41]. Source code for ANNs and to create images is in a GitHub repository with links to pre-trained models[36]. For additional information contact the corresponding author (J.M.E.).

Received: 12 February 2020; Accepted: 28 April 2020;
Published online: 20 May 2020

References

1. Yankovich, A. B., Berkels, B., Dahmen, W., Binev, P. & Voyles, P. M. High-Precision Scanning Transmission Electron Microscopy at Coarse Pixel Sampling for Reduced Electron Dose. *Adv. Struct. Chem. Imaging* **1**, 2 (2015).
2. Peters, J. J. P., Apachitei, G., Beanland, R., Alexe, M. & Sanchez, A. M. Polarization Curling and Flux Closures in Multiferroic Tunnel Junctions. *Nat. Commun.* **7**, 13484 (2016).
3. Hujsak, K., Myers, B. D., Roth, E., Li, Y. & Dravid, V. P. Suppressing Electron Exposure Artifacts: An Electron Scanning Paradigm with Bayesian Machine Learning. *Microsc. Microanal.* **22**, 778–788 (2016).
4. Egerton, R. F., Li, P. & Malac, M. Radiation Damage in the TEM and SEM. *Micron* **35**, 399–409 (2004).
5. Jones, L. *et al.* Managing Dose-, Damage- and Data-Rates in Multi-Frame Spectrum-Imaging. *Microscopy* **67**, i98–i113 (2018).
6. Trampert, P. *et al.* How Should a Fixed Budget of Dwell Time be Spent in Scanning Electron Microscopy to Optimize Image Quality? *Ultramicroscopy* **191**, 11–17 (2018).
7. Anderson, H. S., Ilic-Helms, J., Rohrer, B., Wheeler, J. & Larson, K. Sparse Imaging for Fast Electron Microscopy. In *Computational Imaging XI*, vol. 8657, 86570C (International Society for Optics and Photonics, 2013).
8. Stevens, A., Yang, H., Carin, L., Arslan, I. & Browning, N. D. The Potential for Bayesian Compressive Sensing to Significantly Reduce Electron Dose in High-Resolution STEM Images. *Microscopy* **63**, 41–51 (2013).
9. Stevens, A. *et al.* A Sub-Sampled Approach to Extremely Low-Dose STEM. *Appl. Phys. Lett.* **112**, 043104 (2018).
10. Hwang, S., Han, C. W., Venkatakrishnan, S. V., Bouman, C. A. & Ortalan, V. Towards the Low-Dose Characterization of Beam Sensitive Nanostructures via Implementation of Sparse Image Acquisition in Scanning Transmission Electron Microscopy. *Meas. Sci. Technol.* **28**, 045402 (2017).
11. Candes, E. & Romberg, J. Sparsity and Incoherence in Compressive Sampling. *Inverse Probl.* **23**, 969 (2007).
12. Kovarik, L., Stevens, A., Liyu, A. & Browning, N. D. Implementing an Accurate and Rapid Sparse Sampling Approach for Low-Dose Atomic Resolution STEM Imaging. *Appl. Phys. Lett.* **109**, 164102 (2016).
13. Sang, X. *et al.* Dynamic Scan Control in STEM: Spiral Scans. *Adv. Struct. Chem. Imaging* **2**, 6 (2017).
14. Béché, A., Goris, B., Freitag, B. & Verbeeck, J. Development of a Fast Electromagnetic Beam Blanker for Compressed Sensing in Scanning Transmission Electron Microscopy. *Appl. Phys. Lett.* **108**, 093103 (2016).
15. Li, X., Dyck, O., Kalinin, S. V. & Jesse, S. Compressed Sensing of Scanning Transmission Electron Microscopy (STEM) with Nonrectangular Scans. *Microsc. Microanal.* **24**, 623–633 (2018).

16. Sang, X. *et al*. Precision Controlled Atomic Resolution Scanning Transmission Electron Microscopy using Spiral Scan Pathways. *Sci. Reports* **7**, 43585 (2017).
17. Seki, T., Ikuhara, Y. & Shibata, N. Theoretical Framework of Statistical Noise in Scanning Transmission Electron Microscopy. *Ultramicroscopy* **193**, 118–125 (2018).
18. Wu, X. *et al*. Deep Portrait Image Completion and Extrapolation. *IEEE Transactions on Image Process*. (2019).
19. Liu, G. *et al*. Image Inpainting for Irregular Holes using Partial Convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 85–100 (2018).
20. Yang, W. *et al*. Deep Learning for Single Image Super-Resolution: A Brief Review. *IEEE Transactions on Multimed*. (2019).
21. Fang, L. *et al*. Deep Learning-Based Point-Scanning Super-Resolution Imaging. *bioRxiv* 740548 (2019).
22. de Haan, K., Ballard, Z. S., Rivenson, Y., Wu, Y. & Ozcan, A. Resolution Enhancement in Scanning Electron Microscopy using Deep Learning. *Sci. Reports* **9**, 12050, https://doi.org/10.1038/s41598-019-48444-2 (2019).
23. Ede, J. M. Deep Learning Supersampled Scanning Transmission Electron Microscopy. *arXiv preprint arXiv:1910.10467* (2019).
24. Tan, C. *et al*. *A Survey on Deep Transfer Learning. In International Conference on Artificial Neural Networks*, 270–279 (Springer, 2018).
25. Raschka, S. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. *arXiv preprint arXiv:1811.12808* (2018).
26. Roh, Y., Heo, G. & Whang, S. E. A Survey on Data Collection for Machine Learning: A Big Data-AI Integration Perspective. *IEEE Transactions on Knowl. Data Eng*. (2019).
27. Krizhevsky, A., Nair, V. & Hinton, G. The CIFAR-10 Dataset. Online: http://www.cs.toronto.edu/~kriz/cifar.html (2014).
28. Krizhevsky, A. & Hinton, G. Learning Multiple Layers of Features from Tiny Images. Tech. Rep., Citeseer (2009).
29. LeCun, Y., Cortes, C. & Burges, C. MNIST Handwritten Digit Database. AT&T Labs, online: http://yann.lecun.com/exdb/mnist (2010).
30. Russakovsky, O. *et al*. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **115**, 211–252 (2015).
31. Goodfellow, I. *et al*. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, 2672–2680 (2014).
32. Bang, D. & Shim, H. MGGAN: Solving Mode Collapse using Manifold Guided Training. *arXiv preprint arXiv:1804.04391* (2018).
33. Mirza, M. & Osindero, S. Conditional Generative Adversarial Nets. *arXiv preprint arXiv:1411.1784* (2014).
34. Wang, T.-C. *et al*. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8798–8807 (2018).
35. Larsen, A. B. L., Sønderby, S. K., Larochelle, H. & Winther, O. Autoencoding Beyond Pixels using a Learned Similarity Metric. *arXiv preprint arXiv:1512.09300* (2015).
36. Ede, J. M. Partial STEM Repository. Online: https://github.com/Jeffrey-Ede/partial-STEM, https://doi.org/10.5281/zenodo.3662481 (2019).
37. Gatan. Gatan Microscopy Suite. Online: www.gatan.com/products/tem-analysis/gatan-microscopy-suite-software (2019).
38. Landau, H. Sampling, Data Transmission, and the Nyquist Rate. *Proc. IEEE* **55**, 1701–1706 (1967).
39. Adobe Developers Association *et al*. TIFF Revision 6.0. Online: www.adobe.io/content/dam/udp/en/open/standards/tiff/TIFF6.pdf (1992).
40. Ede, J. M. STEM Datasets. Online: https://github.com/Jeffrey-Ede/datasets/wiki (2019).
41. Ede, J. M. Warwick Electron Microscopy Datasets. *arXiv preprint arXiv:2003.01113* (2020).
42. Ophus, C., Ciston, J. & Nelson, C. T. Correcting Nonlinear Drift Distortion of Scanning Probe and Scanning Transmission Electron Microscopies from Image Pairs with Orthogonal Scan Directions. *Ultramicroscopy* **162**, 1–9 (2016).
43. Sang, X. & LeBeau, J. M. Revolving Scanning Transmission Electron Microscopy: Correcting Sample Drift Distortion Without Prior Knowledge. *Ultramicroscopy* **138**, 28–35 (2014).
44. Krause, F. F. *et al*. ISTEM: A Realisation of Incoherent Imaging for Ultra-High Resolution TEM Beyond the Classical Information Limit. In *European Microscopy Congress 2016: Proceedings*, 501–502 (Wiley Online Library, 2016).
45. Hartel, P., Rose, H. & Dinges, C. Conditions and Reasons for Incoherent Imaging in STEM. *Ultramicroscopy* **63**, 93–114 (1996).
46. Neyshabur, B., Bhojanapalli, S., McAllester, D. & Srebro, N. Exploring Generalization in Deep Learning. In *Advances in Neural Information Processing Systems*, 5947–5956 (2017).
47. Kawaguchi, K., Kaelbling, L. P. & Bengio, Y. Generalization in Deep Learning. *arXiv preprint arXiv:1710.05468* (2017).
48. Ruder, S. An Overview of Gradient Descent Optimization Algorithms. *arXiv preprint arXiv:1609.04747* (2016).
49. Abadi, M. *et al*. TensorFlow: A System for Large-Scale Machine Learning. In *OSDI*, vol. 16, 265–283 (2016).
50. McCann, M. T., Jin, K. H. & Unser, M. Convolutional Neural Networks for Inverse Problems in Imaging: A Review. *IEEE Signal Process. Mag.* **34**, 85–95 (2017).
51. Krizhevsky, A., Sutskever, I. & Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, 1097–1105 (2012).
52. Szegedy, C. *et al*. Going Deeper with Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–9 (2015).
53. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2818–2826 (2016).
54. Durugkar, I., Gemp, I. & Mahadevan, S. Generative Multi-Adversarial Networks. arXiv preprint arXiv:1611.01673 (2016).
55. Isola, P., Zhu, J.-Y., Zhou, T. & Efros, A. A. Image-to-Image Translation with Conditional Adversarial Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1125–1134 (2017).
56. Kingma, D. P. & Ba, J. ADAM: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* (2014).
57. Zou, D., Cao, Y., Zhou, D. & Gu, Q. Stochastic Gradient Descent Optimizes Over-Parameterized Deep ReLU Networks. *arXiv preprint arXiv:1811.08888* (2018).
58. Ede, J. M. & Beanland, R. Adaptive Learning Rate Clipping Stabilizes Learning. Mach. Learn. *Sci. Technol.* (2020).
59. Wang, Z., She, Q. & Ward, T. E. Generative Adversarial Networks: A Survey and Taxonomy. arXiv preprint arXiv:1906.01529 (2019).
60. Dong, H.-W. & Yang, Y.-H. Towards a Deeper Understanding of Adversarial Losses. *arXiv preprint arXiv:1901.08753* (2019).
61. Miyato, T., Kataoka, T., Koyama, M. & Yoshida, Y. Spectral Normalization for Generative Adversarial Networks. *arXiv preprint arXiv:1802.05957* (2018).
62. Mao, X. *et al*. Least Squares Generative Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2794–2802 (2017).
63. Wang, Z., Bovik, A. C., Sheikh, H. R. & Simoncelli, E. P. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Process.* **13**, 600–612 (2004).
64. Ede, J. M. & Beanland, R. Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder. *Ultramicroscopy* **202**, 18–25 (2019).
65. S'ari, M., Cattle, J., Hondow, N., Brydson, R. & Brown, A. Low Dose Scanning Transmission Electron Microscopy of Organic Crystals by Scanning Moiré Fringes. *Micron* **120**, 1–9 (2019).
66. Mayoral, A., Mahugo, R., Sánchez-Sánchez, M. & Díaz, I. Cs-Corrected STEM Imaging of Both Pure and Silver-Supported Metal-Organic Framework MIL-100 (Fe). *ChemCatChem* **9**, 3497–3502 (2017).
67. Gnanasekaran, K., de With, G. & Friedrich, H. Quantification and Optimization of ADF-STEM Image Contrast for Beam-Sensitive Materials. *Royal Soc. Open Sci.* **5**, 171838 (2018).

68. Ilett, M., Brydson, R., Brown, A. & Hondow, N. Cryo-Analytical STEM of Frozen, Aqueous Dispersions of Nanoparticles. *Micron* **120**, 35–42 (2019).
69. Kumar, A., Dhall, R. & LeBeau, J. M. *In Situ* Ferroelectric Domain Dynamics Probed with Differential Phase Contrast Imaging. *Microsc. Microanal.* **25**, 1838–1839 (2019).
70. Xie, L. *et al.* Static and Dynamic Polar Nanoregions in Relaxor Ferroelectric $Ba(Ti_{1-x}Sn_x)O_3$ System at High Temperature. *Phys. Rev. B* **85**, 014118 (2012).
71. Aydin, C. *et al.* Tracking Iridium Atoms with Electron Microscopy: First Steps of Metal Nanocluster Formation in One-Dimensional Zeolite Channels. *Nano Lett.* **11**, 5537–5541 (2011).
72. Hussein, H. E. *et al.* Tracking Metal Electrodeposition Dynamics from Nucleation and Growth of a Single Atom to a Crystalline Nanoparticle. *ACS Nano* **12**, 7388–7396 (2018).
73. Chen, S. *et al.* Atomic Structure and Migration Dynamics of MoS_2/Li_xMoS_2 Interface. *Nano Energy* **48**, 560–568 (2018).
74. Ziatdinov, M. *et al.* Deep Learning of Atomically Resolved Scanning Transmission Electron Microscopy Images: Chemical Identification and Tracking Local Transformations. *ACS Nano* **11**, 12742–12752 (2017).
75. Jones, L. & Nellist, P. D. Identifying and Correcting Scan Noise and Drift in the Scanning Transmission Electron Microscope. *Microsc. Microanal.* **19**, 1050–1060 (2013).
76. Salimans, T. *et al.* Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems*, 2234–2242 (2016).
77. Liang, K. J., Li, C., Wang, G. & Carin, L. Generative Adversarial Network Training is a Continual Learning Problem. *arXiv preprint arXiv:1811.11083* (2018).
78. Hornik, K., Stinchcombe, M. & White, H. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks* **2**, 359–366 (1989).
79. Lin, H. W., Tegmark, M. & Rolnick, D. Why does Deep and Cheap Learning Work so Well? *J. Stat. Phys.* **168**, 1223–1247 (2017).
80. Weill, C. *et al.* AdaNet: A Scalable and Flexible Framework for Automatically Learning Ensembles. *arXiv preprint arXiv:1905.00080* (2019).
81. Molino, P., Dudin, Y. & Miryala, S. S. Ludwig: A Type-Based Declarative Deep Learning Toolbox. *arXiv preprint arXiv:1909.07930* (2019).
82. He, X., Zhao, K. & Chu, X. AutoML: A Survey of the State-of-the-Art. *arXiv preprint arXiv:1908.00709* (2019).
83. Harrington, B. *et al.* Inkscape 0.92, Online: http://www.inkscape.org/ (2020).

Acknowledgements

Thanks go to Julie Robinson for advice on finding publication venues and to Marin Alexe for helpful discussion. J.M.E. and R.B. acknowledge EPSRC grant EP/N035437/1 for financial support. In addition, J.M.E. acknowledges EPSRC Studentship 1917382.

Author contributions

J.M.E. proposed this research, wrote the code, collated training data, performed experiments and analysis, created repositories, and co-wrote this paper. R.B. supervised and co-wrote this paper.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information is available for this paper at https://doi.org/10.1038/s41598-020-65261-0.

Correspondence and requests for materials should be addressed to J.M.E.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this license, visit http://creativecommons.org/licenses/by/4.0/.

© The Author(s) 2020

Supplementary Information: Partial Scanning Transmission Electron Microscopy with Deep Learning

Jeffrey M. Ede[1,*] **and Richard Beanland**[1]

[1]University of Warwick, Department of Physics, Coventry, CV4 7AL, UK
[*]j.m.ede@warwick.ac.uk

S1 Detailed Architecture

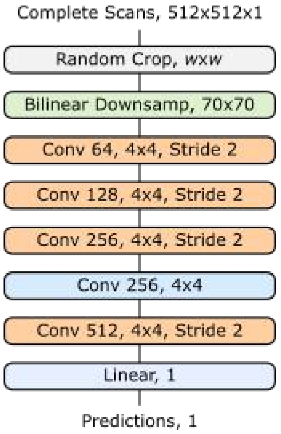


Figure S1. Discriminators examine random $w \times w$ crops to predict whether complete scans are real or generated. Generators are trained by multiple discriminators with different w. This figure was created with Inkscape[1].

Discriminator architecture is shown in Fig. S1. Generator and inner generator trainer architecture is shown in Fig. S2. The components in our networks are

Bilinear Downsamp, *w*x*w*: This is an extension of linear interpolation in one dimension to two dimensions. It is used to downsample images to $w \times w$.

Bilinear Upsamp, x*s*: This is an extension of linear interpolation in one dimension to two dimensions. It is used to upsample images by a factor of s.

Conv *d*, *w*x*w*, Stride, *x*: Convolutional layer with a square kernel of width, w, that outputs d feature channels. If the stride is specified, convolutions are only applied to every xth spatial element of their input, rather than to every element. Striding is not applied depthwise.

Linear, *d*: Flatten input and fully connect it to d feature channels.

Random Crop, *w*x*w*: Randomly sample a $w \times w$ spatial location using an external probability distribution.

⊕: Circled plus signs indicate residual connections where incoming tensors are added together. These help reduce signal attenuation and allow the network to learn perturbative transformations more easily.

All generator convolutions are followed by running mean-only batch normalization then ReLU activation, except output convolutions. All discriminator convolutions are followed by slope 0.2 leaky ReLU activation.

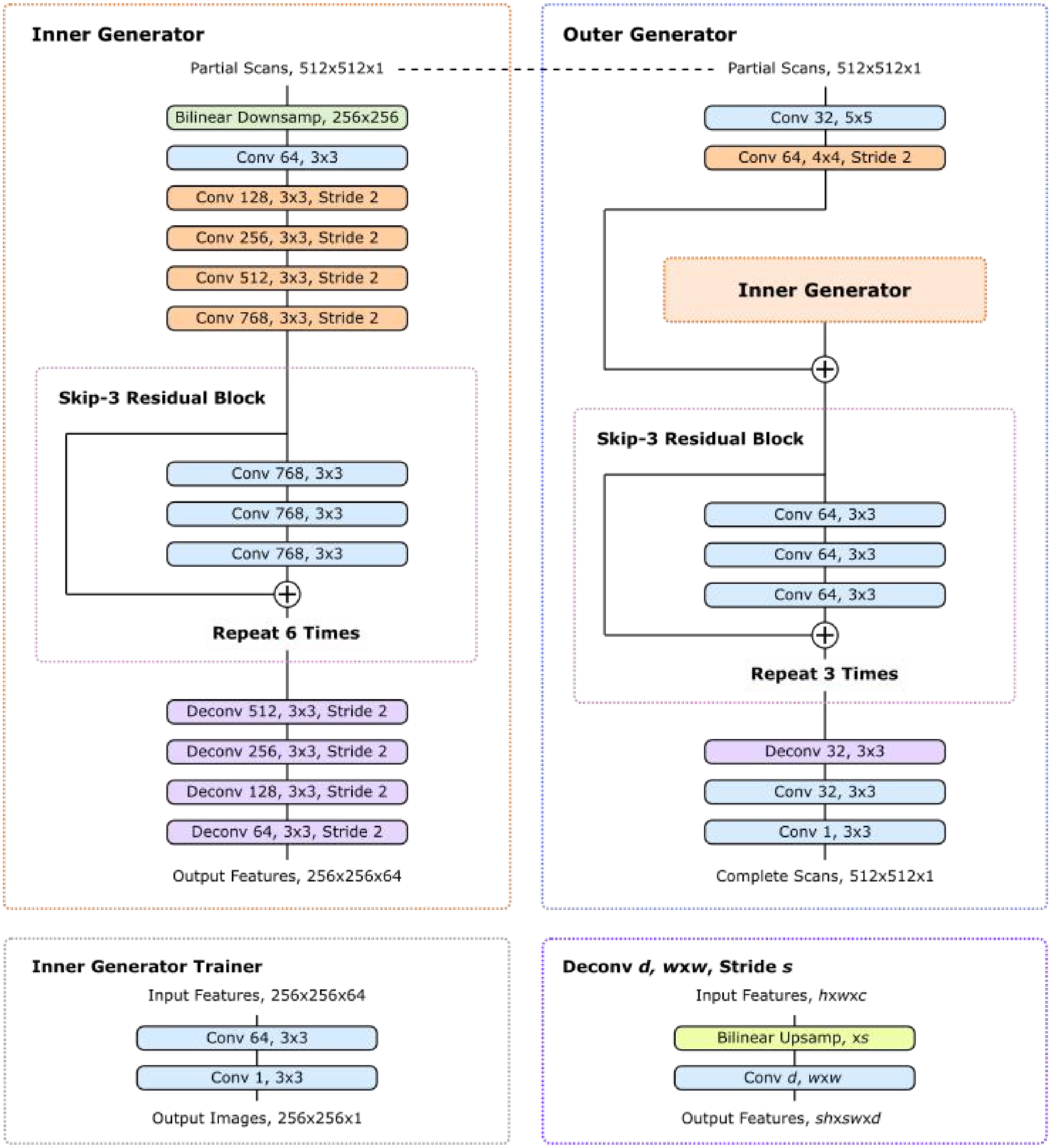


Figure S2. Two-stage generator that completes 512×512 micrographs from partial scans. A dashed line indicates that the same image is input to the inner and outer generator. Large scale features developed by the inner generator are locally enhanced by the outer generator and turned into images. An auxiliary trainer network restores images from inner generator features to provide direct feedback. This figure was created with Inkscape[1].

S2 Learning Policy

Optimizer: Training is ADAM[2] optimized and has two halves. In the first half, the generator and auxiliary trainer learn to minimize mean squared errors between their outputs and ground truth images. For the quarter of iterations, we use a constant learning rate $\eta_0 = 0.0003$ and a decay rate for the first moment of the momentum $\beta_1 = 0.9$. The learning rate is then stepwise decayed to zero in eight steps over the second quarter of iterations. Similarly, β_1 is stepwise linearly decayed to 0.5 in eight steps. In an optional second half, the generator and discriminators play an adversarial game conditioned on MSE guidance. For the third quarter of iterations, we use $\eta = 0.0001$ and $\beta_1 = 0.9$ for the generator and discriminators. In the final quarter of iterations, the generator learning rate is decayed to zero in eight steps while the discriminator learning rate remains constant. Similarly, generator and discriminator β_1 is stepwise decayed to 0.5 in eight steps.

Experiments with GAN training hyperparameters show that $\beta_1 = 0.5$ is a good choice[3]. Our decision to start at $\beta_1 = 0.9$ aims to improve the initial rate of convergence. In the first stage, generator and auxiliary trainer parameters are both updated once per training step. In the second stage, all parameters are updated once per training step. In most of our initial experiments with burred masks, we used a total of 10^6 training iterations. However, we found that validation errors do not diverge if training time is increased to 2×10^6 iterations, and used this number for experiments with binary masks. These training iterations are in-line with with other GANs, which reuse datasets containing a few thousand examples for 200 epochs[4]. The lack of validation divergence suggests that performance may be substantially improved, and means that our results present lower bounds for performance. All training was performed with a batch size of 1 due to the large model size needed to complete 512×512 scans.

Adaptive learning rate clipping: To stabilize batch size 1 training, adaptive learning rate clipping[5] (ALRC) was developed to limit high MSEs. ALRC layers were initialized with first raw moment $\mu_1 = 25$, second raw moment $\mu_2 = 30$, exponential decay rates $\beta_1 = \beta_2 = 0.999$, and $n = 3$ standard deviations.

Input normalization: Partial scans, I_scan, input to the generator are linearly transformed to $I'_\text{scan} = (I_\text{scan} + 1)/2$, where $I'_\text{scan} \in [0,1]$. The generator is trained to output ground truth crops in $[0,1]$, which are linearly transformed to $[-1,1]$. Generator outputs and ground truth crops in $[-1,1]$ are directly input to discriminators.

Weight normalization: All generator parameters are weight normalized[6]. Running mean-only batch normalization[6,7] is applied to the output channels of every convolutional layer, except the last. Channel means are tracked by exponential moving averages with decay rates of 0.99. Running mean-only batch normalization is frozen in the second half of training to improve stability[8].

Spectral normalization: Spectral normalization[3] is applied to the weights of each convolutional layer in the discriminators to limit the Lipschitz norms of the discriminators. We use the power iteration method with one iteration per training step to enforce a spectral norm of 1 for each weight matrix.

Spectral normalization stabilizes training, reduces susceptibility to mode collapse and is independent of rank, encouraging discriminators to use more input features to inform decisions[3]. In contrast, weight normalization[6] and Wasserstein weight clipping[9] impose more arbitrary model distributions that may only partially match the target distribution.

Activation: In the generator, ReLU[10] non-linearities are applied after running mean-only batch normalization. In the discriminators, slope 0.2 leaky ReLU[11] non-linearities are applied after every convolutional layer. Rectifier leakage encourages discriminators to use more features to inform decisions. Our choice of generator and discriminator non-linearities follows recent work on high-resolution conditional GANs[4].

Initialization: Generator weights were initialized from a normal distribution with mean 0.00 and standard deviation 0.05. To apply weight normalization, an example scan is then propagated through the network. Each layer output is divided by its L2 norm and the layer weights assigned their division by the square root of the L2 normalized output's standard deviation. There are no biases in the generator as running mean-only batch normalization would allow biases to grow unbounded c.f. batch normalization[12].

Discriminator weights were initialized from a normal distribution with mean 0.00 and standard deviation 0.03. Discriminator biases were zero initialized.

Experience replay: To reduce destabilizing discriminator oscillations[13], we used an experience replay[14,15] with 50 examples. Prioritizing the replay of difficult examples can improve learning[16], so we only replayed examples with losses in the top 20%. Training examples had a 20% chance to be sampled from the replay.

S3 Experiments

In this section, we present learning curves for some of our non-adversarial architecture and learning policy experiments. During training, each training set example was reused ∼8 times. In comparison, some generative adversarial networks (GANs) are trained on the same data hundreds of times[4]. As a result, we did not experience noticeable overfitting. In cases where final errors are similar; so that their difference is not significant within the error of a single experiment, we choose the lowest error approach. In practice, choices between similar errors are unlikely to have a substantial effect on performance. Each experiment took a few days with an Nvidia GTX 1080 Ti GPU. All learning curves are 2500 iteration boxcar averaged. In addition, the first 10^4 iterations before dashed lines in figures, where losses rapidly decrease, are not shown.

Following previous work on high-resolution GANs[4], we used a multi-stage training protocol for our initial experiments. The outer generator was trained separately; after the inner generator, before fine-tuning the inner and outer generator together. An alternative approach uses an auxiliary loss network for end-to-end training, similar to Inception[17,18]. This can provide a more direct path for gradients to back-propagate to the start of the network and introduces an additional regularization mechanism. Experimenting, we connected an auxiliary trainer to the inner generator and trained the network in a single stage. As shown by Fig. S3a, auxiliary network supported end-to-end training is more stable and converges to lower errors.

In encoder-decoders, residual connections[19] between strided convolutions and symmetric strided transpositional convolutions can be used to reduce information loss. This is common in noise removal networks where the output is similar to the input[20,21]. However, symmetric residual connections are also used in encoder-decoder networks for semantic image segmentation[22] where the input and output are different. Consequently, we tried adding symmetric residual connections between strided and transpositional inner generator convolutions. As shown by Fig. S3b, extra residuals accelerate initial inner generator training. However, final errors are slightly higher and initial inner generator training converged to similar errors with and without symmetric residuals. Taken together, this suggests that symmetric residuals initially accelerate training by enabling the final inner generator layers to generate crude outputs though their direct connections to the first inner generator layers. However, the symmetric connections also provide a direct path for low-information outputs of the first layers to get to the final layers, obscuring the contribution of the inner generator's skip-3 residual blocks (section S1) and lowering performance in the final stages of training.

Path information is concatenated to the partial scan input to the generator. In principle, the generator can infer electron beam paths from partial scans. However, the input signal is attenuated as it travels through the network[23]. In addition, path information would have to be deduced; rather than informing calculations in the first inner generator layers, decreasing efficiency. To compensate, paths used to generate partial scans from full scans are concatenated to inputs. As shown by Fig. S3b, concatenating path information reduces errors throughout training. Performance might be further improved by explicitly building sparsity into the network[24].

Large convolutional kernels are often used at the start of neural networks to increase their receptive field. This allows their first convolutions to be used more efficiently. The receptive field can also be increased by increasing network depth, which could also enable more efficient representation of some functions[25]. However, increasing network depth can also increase information loss[23] and representation efficiency may not be limiting. As shown by Fig. S3c, errors are lower for small first convolution kernels; 3×3 for the inner generator and 7×7 for the outer generator or both 3×3, than for large first convolution kernels; 7×7 for the inner generator and 17×17 for the outer generator. This suggests that the generator does not make effective use of the larger 17×17 kernel receptive field and that the variability of the extra kernel parameters harms learning.

Learning curves for different learning rate schedules are shown in Fig. S3d. Increasing training iterations and doubling the learning rate from 0.0002 to 0.0004 lowers errors. Validation errors do not plateau for 10^6 iterations in Fig. S3e, suggesting that continued training would improve performance. In our experiments, validation errors were calculated after every 50 training iterations.

The choice of output domain can affect performance. Training with a [0, 1] output domain is compared against $[-1,1]$ for slope 0.01 leaky ReLU activation after every generator convolution in Fig. S3f. Although $[-1,1]$ is supported by leaky ReLUs, requiring orders of magnitude differences in scale for $[-1,0)$ and $(0,1]$ hinders learning. To decrease dependence on the choice output domain, we do not apply batch normalization or activation after the last generator convolutions in our final architecture.

The $[0,1]$ outputs of Fig. S3f were linearly transformed to $[-1,1]$ and passed through a tanh non-linearity. This ensured that $[0,1]$ output errors were on the same scale as $[-1,1]$ output errors, maintaining the same effective learning rate. Initially, outputs were clipped by a tanh non-linearity to limit outputs far from the target domain from perturbing training. However, Fig. S4a shows that errors are similar without end non-linearites so they were removed. Fig. S4a also shows that replacing slope 0.01 leaky ReLUs with ReLUs and changing all kernel sizes to 3×3 has little effect. Swapping to ReLUs and 3×3 kernels is therefore an option to reduce computation. Nevertheless, we continue to use larger kernels throughout as we think they would usefully increase the receptive field with more stable, larger batch size training.

To more efficiently use the first generator convolutions, we nearest neighbour infilled partial scans. As shown by Fig. S4b, infilling reduces error. However, infilling is expected to be of limited use for low-dose applications as scans can be noisy, making meaningful infilling difficult. Nevertheless, nearest neighbour partial scan infilling is a computationally inexpensive method to improve generator performance for high-dose applications.

To investigate our generator's ability to handle STEM noise[26], we combined uniform noise with partial scans of Gaussian blurred STEM images. More noise was added to low intensity path segments and low-intensity pixels. As shown by Fig. S4c,

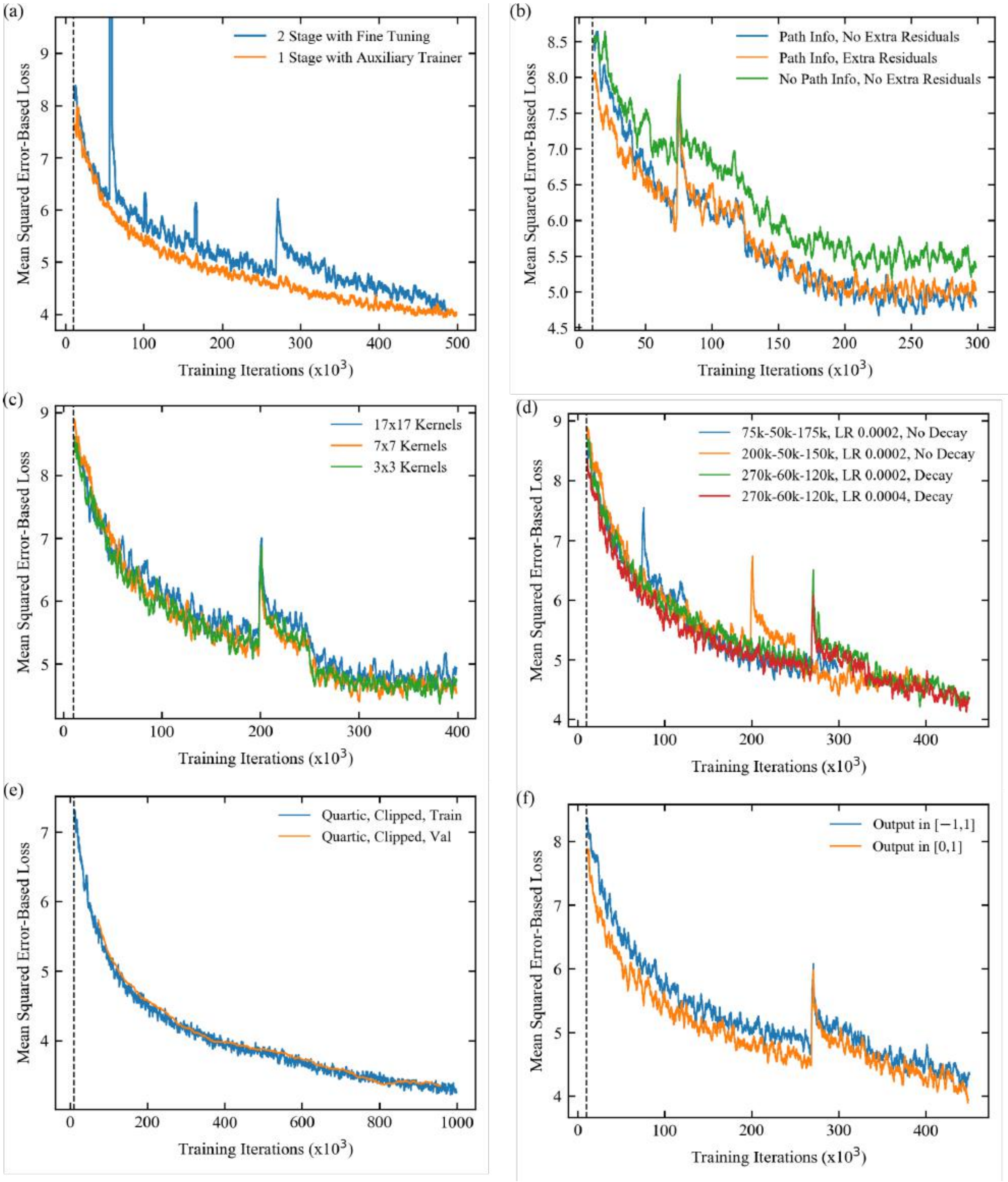


Figure S3. Learning curves. a) Training with an auxiliary inner generator trainer stabilizes training, and converges to lower than two-stage training with fine tuning. b) Concatenating beam path information to inputs decreases losses. Adding symmetric residual connections between strided inner generator convolutions and transpositional convolutions increases losses. c) Increasing sizes of the first inner and outer generator convolutional kernels does not decrease losses. d) Losses are lower after more interations, and a learning rate (LR) of 0.0004; rather than 0.0002. Labels indicate inner generator iterations - outer generator iterations - fine tuning iterations, and k denotes multiplication by 1000 e) Adaptive learning rate clipped quartic validation losses have not diverged from training losses after 10^6 iterations. f) Losses are lower for outputs in [0, 1] than for outputs in [-1, 1] if leaky ReLU activation is applied to generator outputs.

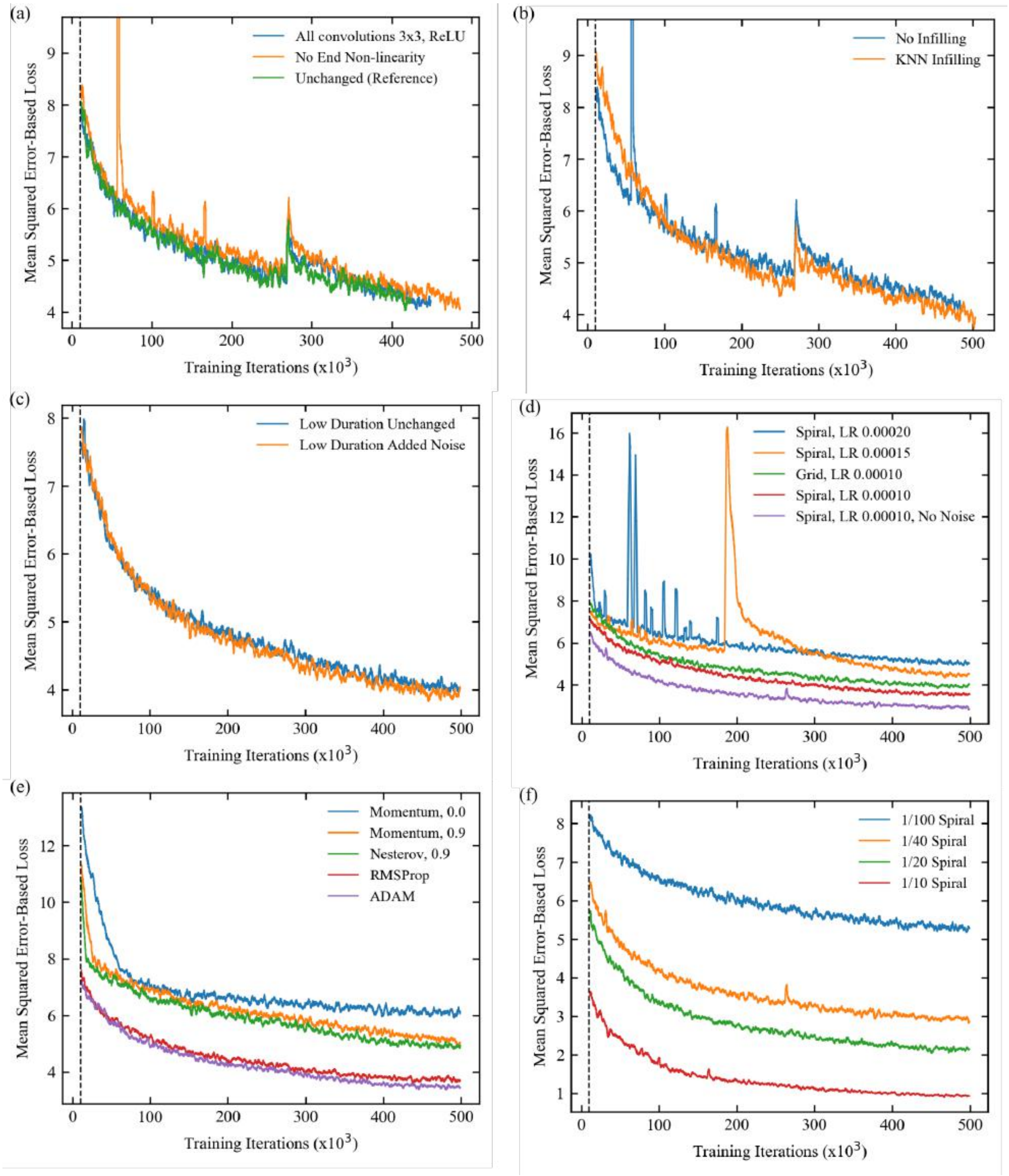


Figure S4. Learning curves. a) Making all convolutional kernels 3×3, and not applying leaky ReLU activation to generator outputs does not increase losses. b) Nearest neighbour infilling decreases losses. Noise was not added to low duration path segments for this experiment. c) Losses are similar whether or not extra noise is added to low-duration path segments. d) Learning is more stable and converges to lower errors at lower learning rates (LRs). Losses are lower for spirals than grid-like paths, and lowest when no noise is added to low-intensity path segments. e) Adaptive momentum-based optimizers, ADAM and RMSProp, outperform non-adaptive momentum optimizers, including Nesterov-accelerated momentum. ADAM outperforms RMSProp; however, training hyperparameters and learning protocols were tuned for ADAM. Momentum values were 0.9. f) Increasing partial scan pixel coverages listed in the legend decreases losses.

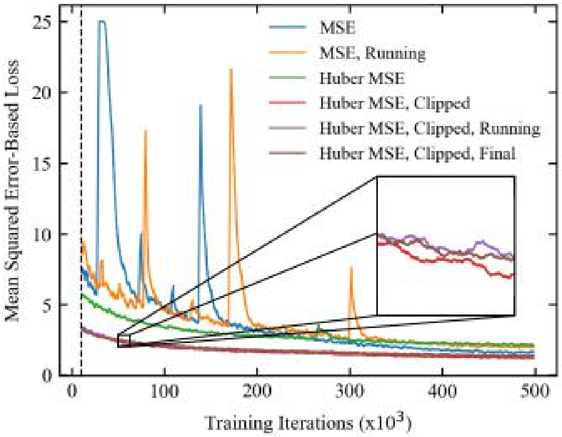


Figure S5. Adaptive learning rate clipping stabilizes learning, accelerates convergence and results in lower errors than Huberisation. Weighting pixel errors with their running or final mean errors is ineffective.

ablating extra noise for low-duration path segments increases performance.

Fig. S4d shows that spiral path training is more stable and reaches lower errors at lower learning rates. At the same learning rate, spiral paths converge to lower errors than grid-like paths as spirals have more uniform coverage. Errors are much lower for spiral paths when both intensity- and duration-dependent noise is ablated.

To choose a training optimizer, we completed training with stochastic gradient descent, momentum, Nesterov-accelerated momentum[27,28], RMSProp[29] and ADAM[2]. Learning curves are in Fig. S4e. Adaptive momentum optimizers, ADAM and RMSProp, outperform the non-adaptive optimizers. Non-adaptive momentum-based optimizers outperform momentumless stochastic gradient decent. ADAM slightly outperforms RMSProp; however, architecture and learning policy were tuned for ADAM. This suggests that RMSProp optimization may also be a good choice.

Learning curves for 1/10, 1/20, 1/40 and 1/100 px coverage spiral scans are shown in Fig. S4f. In practice, 1/20 px coverage is sufficient for most STEM images. On average, a non-adversarial generator can complete test set 1/20 px coverage partial scans with a 2.6% root mean squared intensity error. Nevertheless, higher coverage is needed to resolve fine detail in some images. Likewise, lower coverage may be appropriate for images without fine detail. Consequently, we are developing an intelligent scan system that adjusts coverage based on micrograph content.

Training is performed with a batch size of 1 due to the large network size needed for 512×512 partial scans. However, MSE training is unstable and large error spikes destabilize training. To stabilize learning, we developed adaptive learning rate clipping[5] (ALRC) to limit magnitudes of high losses while preserving their initial gradient distributions. ALRC is compared against MSE, Huberised MSE, and weighting each pixel's error by its Huberised running mean, and fixed final errors in Fig. S5. ALRC results in more stable training with the fastest convergence and lowest errors. Similar improvements have been confirmed for CIFAR-10 and STEM supersampling with ALRC[5].

S4 Additional Examples

Sheets of examples comparing non-adversarial generator outputs and true images are shown in Fig. S6-S12 for 512×512 spiral scans selected with binary masks. True images are blurred by a 5×5 symmetric Gaussian kernel with a 2.5 px standard deviation so that they are the same as the images that generators were trained output. Images are blurred to suppress high-frequency noise. Examples are presented for 1/17.9, 1/27.3, 1/38.2, 1/50.0, 1/60.5, 1/73.7, and 1/87.0 px coverage, in that order, so that higher errors become apparent for decreasing coverage with increasing page number. Quantitative performance characteristics for each generator are provided in the main article.

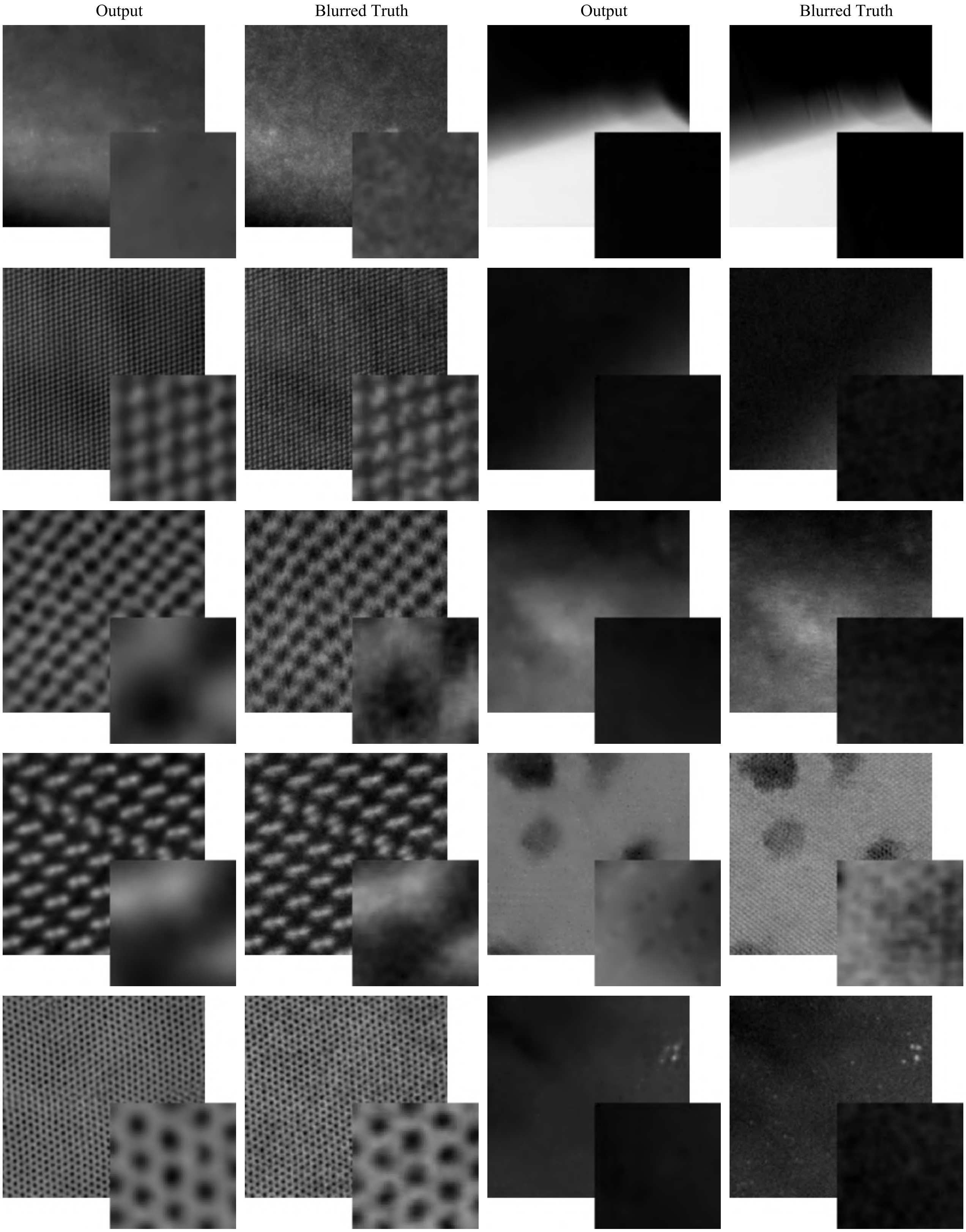


Figure S6. Non-adversarial 512×512 outputs and blurred true images for 1/17.9 px coverage spiral scans selected with binary masks.

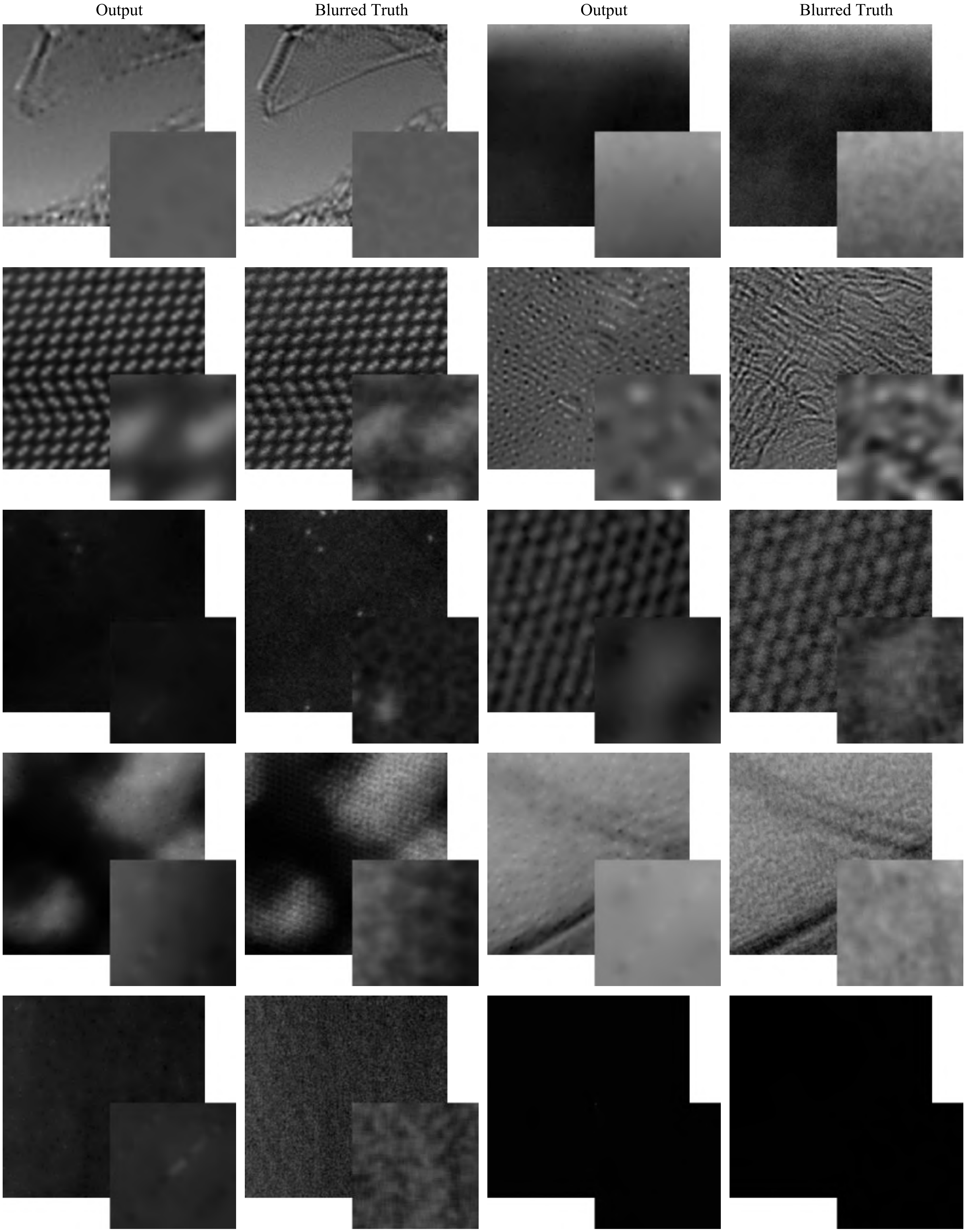


Figure S7. Non-adversarial 512×512 outputs and blurred true images for 1/27.3 px coverage spiral scans selected with binary masks.

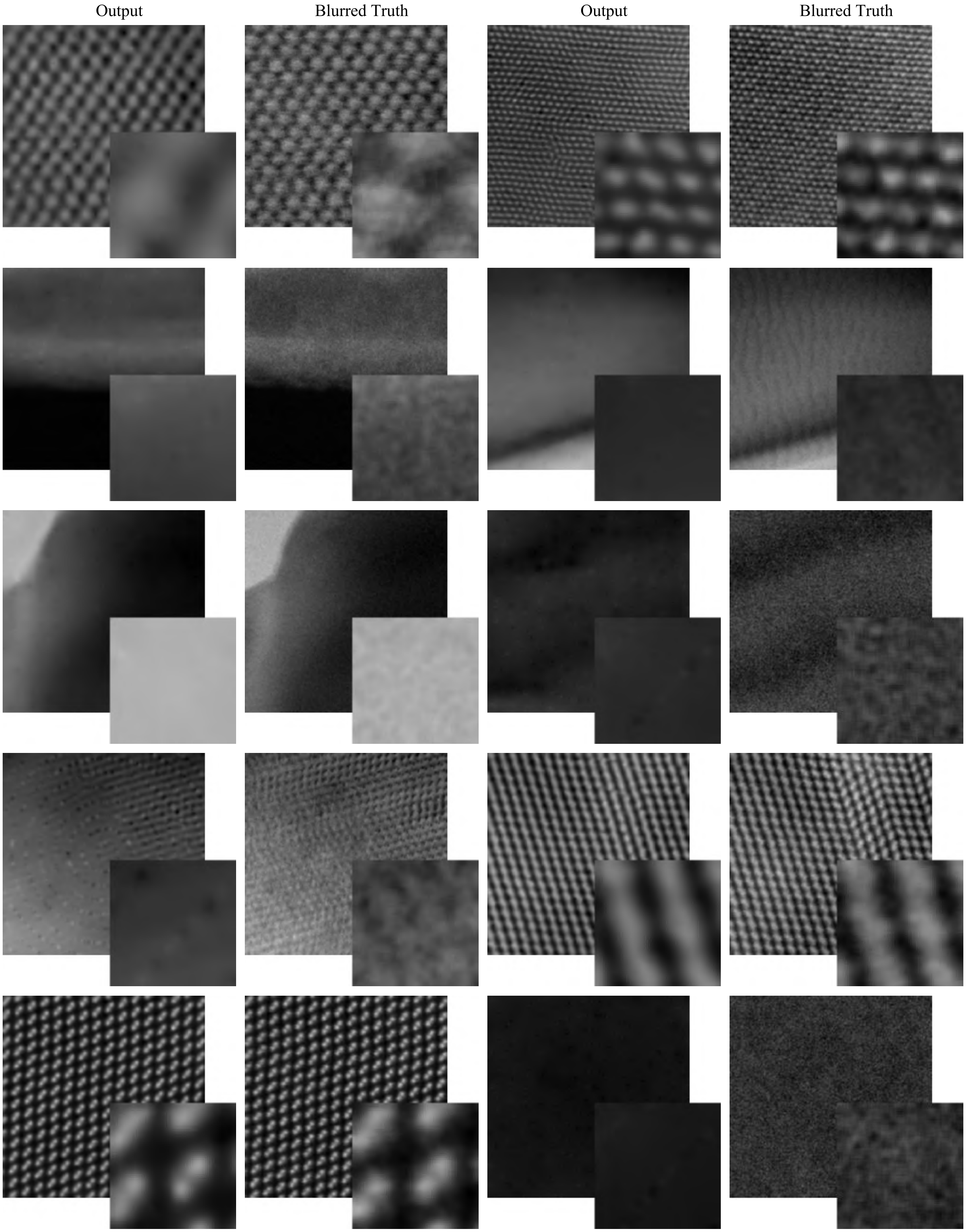


Figure S8. Non-adversarial 512×512 outputs and blurred true images for 1/38.2 px coverage spiral scans selected with binary masks.

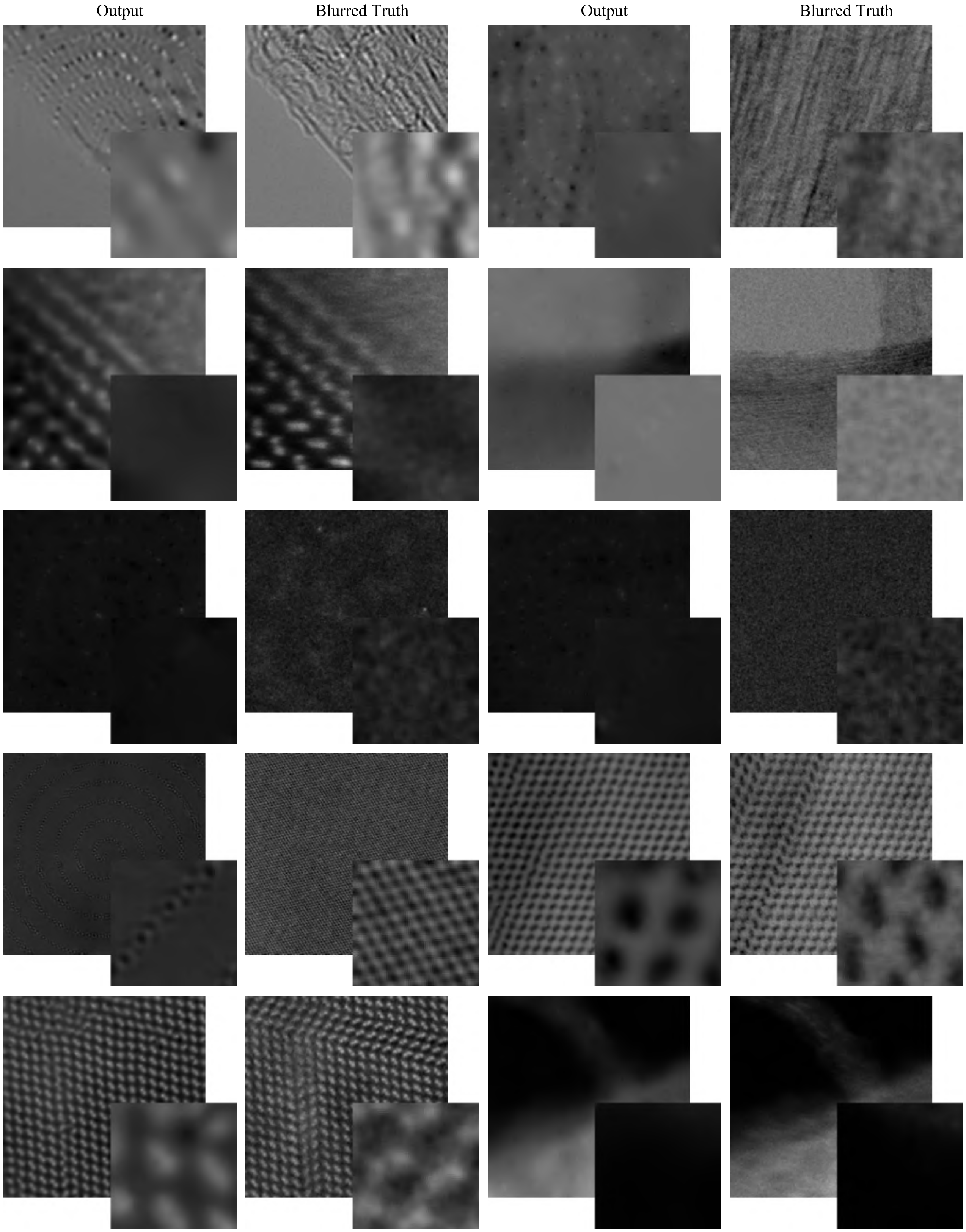


Figure S9. Non-adversarial 512×512 outputs and blurred true images for 1/50.0 px coverage spiral scans selected with binary masks.

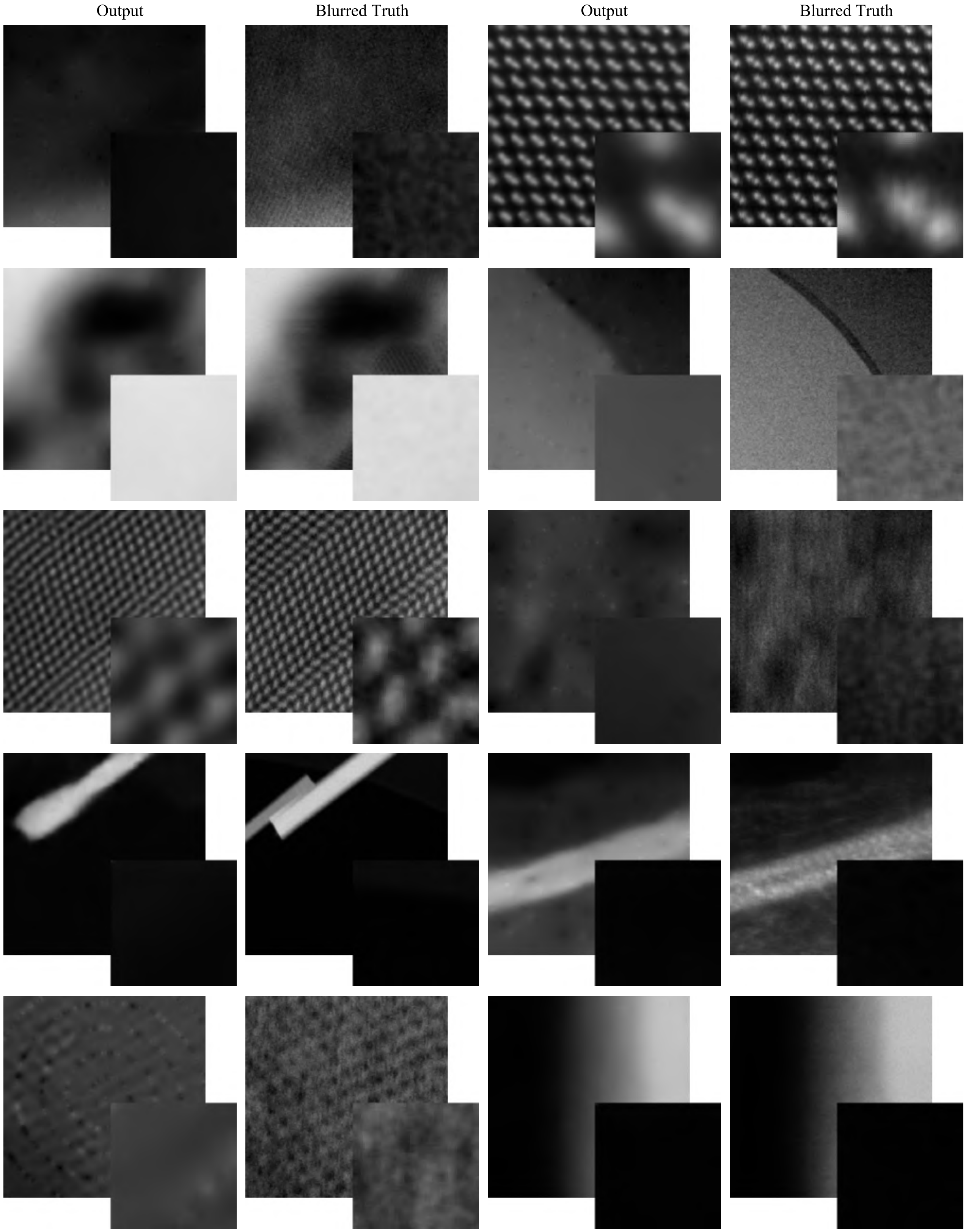


Figure S10. Non-adversarial 512×512 outputs and blurred true images for 1/60.5 px coverage spiral scans selected with binary masks.

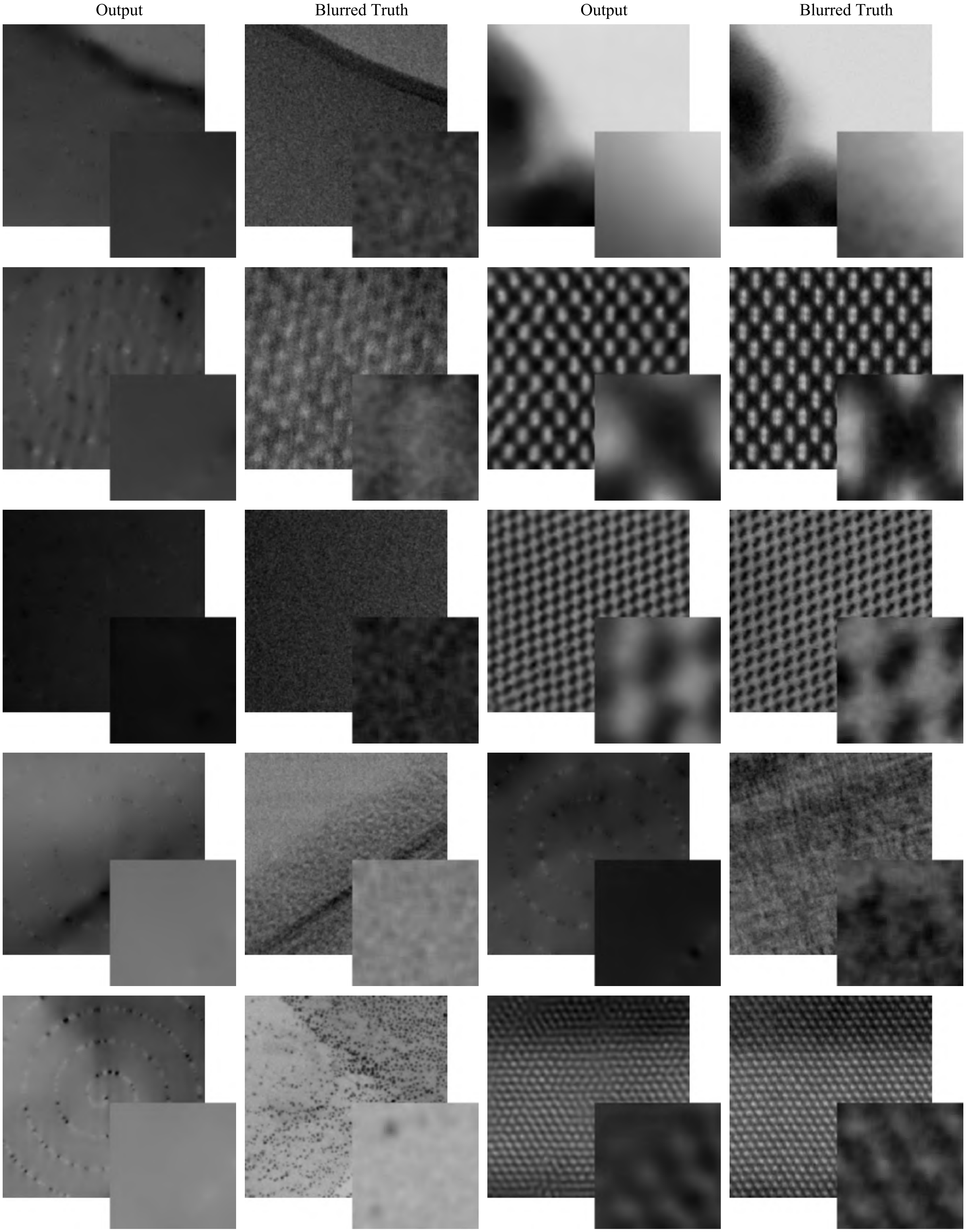


Figure S11. Non-adversarial 512×512 outputs and blurred true images for 1/73.7 px coverage spiral scans selected with binary masks.

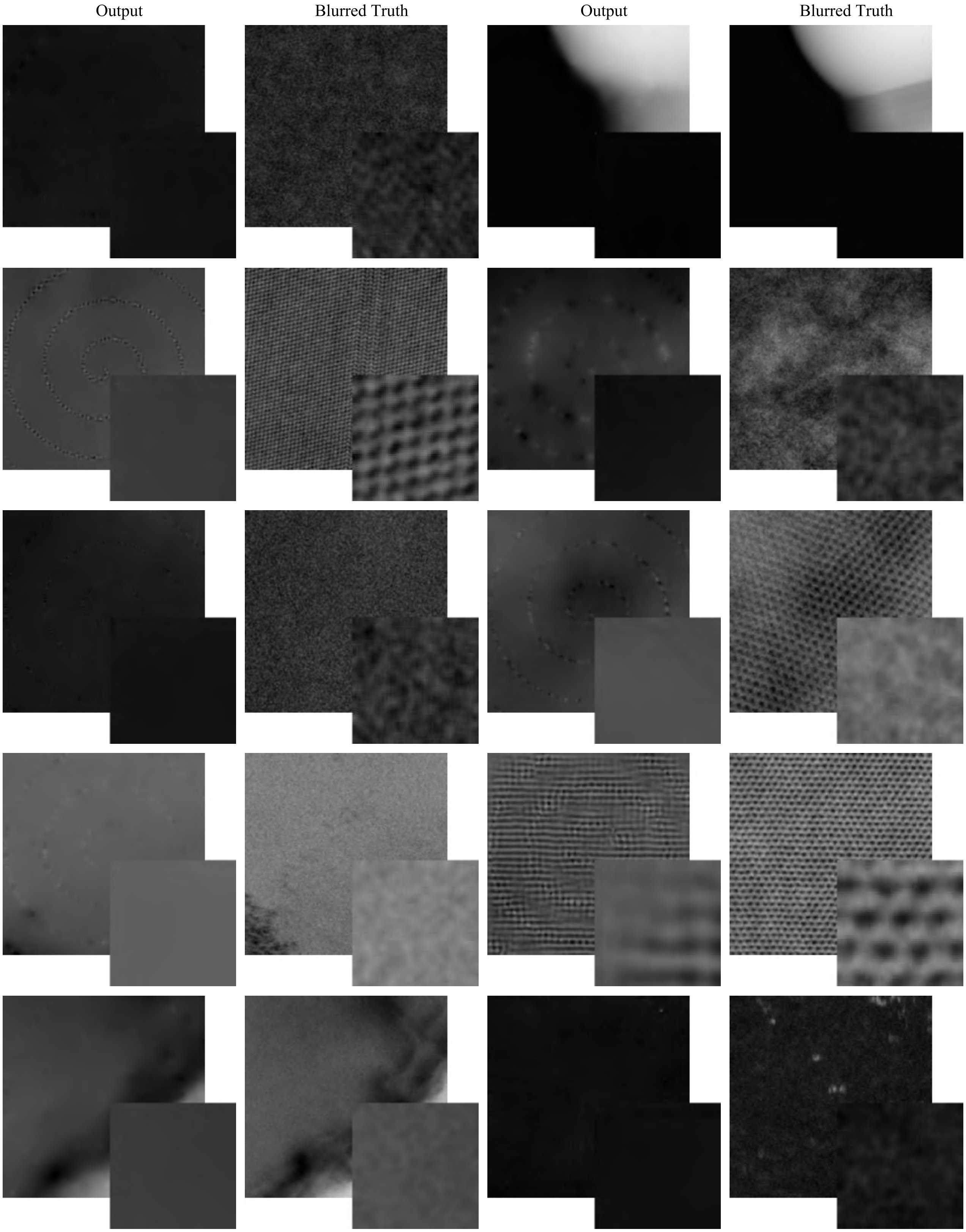


Figure S12. Non-adversarial 512×512 outputs and blurred true images for 1/87.0 px coverage spiral scans selected with binary masks.

References

1. Harrington, B. *et al.* Inkscape 0.92. Online: http://www.inkscape.org/ (2020).
2. Kingma, D. P. & Ba, J. ADAM: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* (2014).
3. Miyato, T., Kataoka, T., Koyama, M. & Yoshida, Y. Spectral Normalization for Generative Adversarial Networks. *arXiv preprint arXiv:1802.05957* (2018).
4. Wang, T.-C. *et al.* High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8798–8807 (2018).
5. Ede, J. M. & Beanland, R. Adaptive Learning Rate Clipping Stabilizes Learning. *Mach. Learn. Sci. Technol.* (2020).
6. Salimans, T. & Kingma, D. P. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. In *Advances in Neural Information Processing Systems*, 901–909 (2016).
7. Hoffer, E., Banner, R., Golan, I. & Soudry, D. Norm Matters: Efficient and Accurate Normalization Schemes in Deep Networks. In *Advances in Neural Information Processing Systems*, 2160–2170 (2018).
8. Chen, L.-C., Papandreou, G., Schroff, F. & Adam, H. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv preprint arXiv:1706.05587* (2017).
9. Arjovsky, M., Chintala, S. & Bottou, L. Wasserstein Generative Adversarial Networks. In *International Conference on Machine Learning*, 214–223 (2017).
10. Nair, V. & Hinton, G. E. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 807–814 (2010).
11. Maas, A. L., Hannun, A. Y. & Ng, A. Y. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *Proceedings of the International Conference on Machine Learning*, vol. 30, 3 (2013).
12. Ioffe, S. & Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv preprint arXiv:1502.03167* (2015).
13. Liang, K. J., Li, C., Wang, G. & Carin, L. Generative Adversarial Network Training is a Continual Learning Problem. *arXiv preprint arXiv:1811.11083* (2018).
14. Pfau, D. & Vinyals, O. Connecting Generative Adversarial Networks and Actor-Critic Methods. *arXiv preprint arXiv:1610.01945* (2016).
15. Shrivastava, A. *et al.* Learning from Simulated and Unsupervised Images through Adversarial Training. *arXiv preprint arXiv: 161207828* (2016).
16. Schaul, T., Quan, J., Antonoglou, I. & Silver, D. Prioritized Experience Replay. *arXiv preprint arXiv:1511.05952* (2015).
17. Szegedy, C. *et al.* Going Deeper with Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–9 (2015).
18. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2818–2826 (2016).
19. He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778 (2016).
20. Mao, X.-J., Shen, C. & Yang, Y.-B. Image Restoration using Convolutional Auto-encoders with Symmetric Skip Connections. *arXiv preprint arXiv:1606.08921* (2016).
21. Casas, L., Navab, N. & Belagiannis, V. Adversarial Signal Denoising with Encoder-Decoder Networks. *arXiv preprint arXiv:1812.08555* (2018).
22. Badrinarayanan, V., Kendall, A. & Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis Mach. Intell.* **39**, 2481–2495 (2017).
23. Zheng, H., Yao, J., Zhang, Y. & Tsang, I. W. Degeneration in VAE: In the Light of Fisher Information Loss. *arXiv preprint arXiv:1802.06677* (2018).
24. Graham, B. Spatially-Sparse Convolutional Neural Networks. *arXiv preprint arXiv:1409.6070* (2014).
25. Lin, H. W., Tegmark, M. & Rolnick, D. Why does Deep and Cheap Learning Work so Well? *J. Stat. Phys.* **168**, 1223–1247 (2017).

26. Seki, T., Ikuhara, Y. & Shibata, N. Theoretical Framework of Statistical Noise in Scanning Transmission Electron Microscopy. *Ultramicroscopy* **193**, 118–125 (2018).

27. Sutskever, I., Martens, J., Dahl, G. & Hinton, G. On the Importance of Initialization and Momentum in Deep Dearning. In *International Conference on Machine Learning*, 1139–1147 (2013).

28. Nesterov, Y. A Method of Solving a Convex Programming Problem with Convergence Rate O($1/k^2$). In *Soviet Mathematics Doklady*, vol. 27, 372–376 (1983).

29. Hinton, G., Srivastava, N. & Swersky, K. Neural Networks for Machine Learning Lecture 6a Overview of Mini-Batch Gradient Descent (2012).

4.2 Amendments and Corrections

There are amendments or corrections to the paper[4] covered by this chapter.

Location: Reference 13 in the bibliography.
Change: "Sang, X. *et al*. Dynamic Scan Control in STEM: Spiral Scans. Adv. Struct. Chem. Imaging 2, 6 (2017)" should say "Sang, X. *et al*. Dynamic Scan Control in STEM: Spiral Scans. Adv. Struct. Chem. Imaging 2, 1–8 (2016)".

4.3 Reflection

This chapter covers our paper titled "Partial Scanning Transmission Electron Microscopy with Deep Learning"[4] and associated research outputs[10,15,18–21,188], which were summarized by Bethany Connolly[189]. Our paper presents some of my investigations into compressed sensing of STEM images. Specifically, it combines results from two of my arXiv papers about compressed sensing with contiguous paths[18] and uniformly spaced grids[19] of probing locations. A third investigation into compressed sensing with a fixed random grid of probing locations was not published as I think that uniformly spaced grid scans are easier to implement on most scan systems. Further, reconstruction errors were usually similar for uniformly spaced and fixed random grids with the same coverage. Nevertheless, a paper I drafted on fixed random grids is openly accessible[190]. Overall, I think that compressed sensing with DNNs is a promising approach to reduce electron beam damage and scan time by 10-100× with minimal information loss.

My comparison of spiral and uniformly spaced grid scans with the same ANN architecture, learning policy and training data indicates that errors are lower for uniformly spaced grids. However, the comparison is not conclusive as ANNs were trained for a few days, rather than until validation errors plateaued. Further, a fair comparison is difficult as suitability of architectures and learning policies may vary for different scan paths. Higher performance of uniformly spaced grids can be explained by content at the focus of most electron micrographs being imaged at 5-10× its Nyquist rate[2] (ch. 2). It follows that high-frequency information that is accessible from neighbouring pixels in contiguous scans is often almost redundant. Overall, I think the best approach may combine both contiguous and uniform spaced grid scans. For example, a contiguous scan ANN could exploit high-frequency information to complete an image, which could then be mapped to a higher resolution image by an ANN for uniformly spaced scans. Indeed, functionality for contiguous and uniformly spaced grid scans could be combined into a single ANN.

Most STEM scan systems can raster uniformly spaced grids of probing locations. However, scan systems often have to be modified to perform spiral or other custom scans[191,192]. Modification is not difficult for skilled programmers. For example, Jonathan Peters[1] created a custom scan controller prototype based on my field programmable gate array[193] (FPGA) within one day. Custom scans are often more distorted than raster scans. However, distortions can be minimized by careful choice of custom scan speed and path shape[191]. Alternatively, ANNs can correct electron microscope scan distortions[194,195]. We planned to use my FPGA to develop an openly accessible custom scan controller near the end of my PhD; however, progress was stalled by COVID-19 national lockdowns in the United Kingdom[196]. As a result, I invested time that we had planned to use for FPGA deployment to review deep learning in electron microscopy[1] (ch. 1).

---

[1] Email: petersjo@tcd.ie

To complete realistic images, generators were trained with MSEs or as part of GANs. However, GANs can introduce uncertainty into scientific investigation as they can generate realistic outputs, even if scan coverage is too low to reliably complete a region[4]. Consequently, investigated reducing uncertainty by adapting scan coverage[5] to imaging regions (ch. 5). Alternatively, there are a variety of methods to quantify DNN uncertainty[197–203]. For example, uncertainty can be predicted by ANNs[204,205], Bayesian uncertainty approximation[206–209], or from variance of bootstrap aggregated[210] (bagged) model outputs. To address uncertainty, we present mean errors for 20000 test images, showing that errors are higher further away from scan paths. However, we do not provide an approach to quantify uncertainty of individual images, which could be critical to make scientific conclusions. Overall, I think that further investigation of uncertainty may be necessary before DNNs are integrated into default operating configurations of electron microscopes.

A GAN could learn to generate any realistic STEM images, rather than outputs that correspond to inputs. To train GANs to generate outputs that correspond to inputs, I added MSEs between blurred input and output images to generator losses. Blurring prevented MSEs from strongly suppressing high-frequency noise characteristics. I also investigated adding distances between features output by discriminator layers for real and generated images to generator losses[48]. However, feature distances require more computation than MSEs, and both feature distances and MSEs result in similar SSIMs[190] between completed and true scans. As a result, I do not think that other computationally inexpensive additional losses, such as SSIMs or mean absolute errors, would substantially improve performance. Finally, I considered training generators to minimize perceptual losses[211]. However, most pretrained models used for feature extraction are not trained on electron micrographs or scientific images. Consequently, I was concerned that pretrained models might not clearly perceive characteristics specific to electron micrographs, such as noise.

Chapter 5

Adaptive Partial Scanning Transmission Electron Microscopy with Reinforcement Learning

5.1 Scientific Paper

This chapter covers the following paper[5] and its supplementary information[11].

> J. M. Ede. Adaptive Partial Scanning Transmission Electron Microscopy with Reinforcement Learning. *arXiv preprint arXiv:2004.02786 (under review by Machine Learning: Science and Technology)*, 2020
>
> J. M. Ede. Supplementary Information: Adaptive Partial Scanning Transmission Electron Microscopy with Reinforcement Learning. Zenodo, Online: `https://doi.org/10.5281/zenodo.4384708`, 2020

Adaptive Partial Scanning Transmission Electron Microscopy with Reinforcement Learning

Jeffrey M. Ede[1,a]

[1]University of Warwick, Department of Physics, Coventry, CV4 7AL, UK
[a]j.m.ede@warwick.ac.uk

ABSTRACT

Compressed sensing can decrease scanning transmission electron microscopy electron dose and scan time with minimal information loss. Traditionally, sparse scans used in compressed sensing sample a static set of probing locations. However, dynamic scans that adapt to specimens are expected to be able to match or surpass the performance of static scans as static scans are a subset of possible dynamic scans. Thus, we present a prototype for a contiguous sparse scan system that piecewise adapts scan paths to specimens as they are scanned. Sampling directions for scan segments are chosen by a recurrent neural network based on previously observed scan segments. The recurrent neural network is trained by reinforcement learning to cooperate with a feedforward convolutional neural network that completes the sparse scans. This paper presents our learning policy, experiments, and example partial scans, and discusses future research directions. Source code, pretrained models, and training data is openly accessible at `https://github.com/Jeffrey-Ede/adaptive-scans`.

Keywords: adaptive scans, compressed sensing, deep learning, electron microscopy, reinforcement learning.

1 Introduction

Most scan systems sample signals at sequences of discrete probing locations. Examples include atomic force microscopy[1,2], computerized axial tomography[3,4], electron backscatter diffraction[5], scanning electron microscopy[6], scanning Raman spectroscopy[7], scanning transmission electron microscopy[8] (STEM) and X-ray diffraction spectroscopy[9]. In STEM, the high current density of electron probes produces radiation damage in many materials, limiting the range and types of investigations that can be performed[10,11]. In addition, most STEM signals are oversampled[12] to ease visual inspection and decrease sub-Nyquist artefacts[13]. As a result, a variety of compressed sensing[14] algorithms have been developed to enable decreased STEM probing[15]. In this paper, we introduce a new approach to STEM compressed sensing where a scan system is trained to piecewise adapt partial scans[16] to specimens by deep reinforcement learning[17] (RL).

Established compressed sensing strategies include random sampling[18–20], uniformly spaced sampling[19,21–23], sampling based on a model of a sample[24,25], partials scans with fixed paths[16], dynamic sampling to minimize entropy[26–29] and dynamic sampling based on supervised learning[30]. Complete signals can be extrapolated from partial scans by an infilling algorithm, estimating their fast Fourier transforms[31] or inferred by an artificial neural network[16,23] (ANN). In general, the best sampling strategy varies for different specimens. For example, uniformly spaced sampling is often better than spiral paths for oversampled STEM images[16]. However, sampling strategies designed by humans usually have limited ability to leverage an understanding of physics to optimize sampling. As proposed by our earlier work[16], we have therefore developed ANNs to dynamically adapt scan paths to specimens. Expected performance of dynamic scans can always match or surpass expected performance of static scans as static scan paths are a special case of dynamic scan paths.

Exploration of STEM specimens is a finite-horizon partially observed Markov decision process[32,33] (POMDP) with sparse losses: A partial scan can be constructed from path segments sampled at each step of the POMDP and a loss can be based on the quality of an scan completion generated from the partial scan with an ANN. Most scan systems support custom scan paths or can be augmented with a field programmable gate array[34,35] (FPGA) to support custom scan paths. However, there is a delay before a scan system can execute or is ready to receive a new command. Total latency can be reduced by using both fewer and larger steps, and decreasing steps may also reduce distortions due to cumulative errors in probing positions[34] after commands are executed. Command execution can also be delayed by ANN inference. However, inference delay can be minimized by using a computationally lightweight ANN and inferring future commands while previous commands are executing.

Markov decision processes (MDPs) can be optimized by recurrent neural networks (RNNs) based on long short-term memory[36,37] (LSTM), gated recurrent unit[38] (GRU), or other cells[39–41]. LSTMs and GRUs are popular as they solve the vanishing gradient problem[42] and have consistently high performance[40]. Small RNNs are computationally inexpensive and are often applied to MDPs as they can learn to extract and remember state information to inform future decisions. To solve dynamic graphs, an RNN can be augmented with dynamic external memory to create a differentiable neural computer[43] (DNC). To optimize a MDP, a discounted future loss, Q_t, at step t in a MDP with T steps can be calculated from step losses, L_t, with Bellman's equation,

$$Q_t = \sum_{t'=t}^{T} \gamma^{t'-t} L_{t'} \,, \tag{1}$$

where $\gamma \in [0,1)$ discounts future step losses. Equations for RL are often presented in terms of rewards, e.g. $r_t = -L_t$; however, losses are an equivalent representation that avoids complicating our equations with minus signs. Discounted future loss backpropagation through time[44] (BPTT) enables RNNs to be trained by gradient descent[45]. However, losses for partial scan completions are not differentiable with respect to (w.r.t.) RNN actions, $(a_1, ..., a_T)$, controlling which path segments are sampled.

Many MDPs have losses that are not differentiable w.r.t. agent actions. Examples include agents directing their vision[46,47], managing resources[48], and playing score-based computer games[49,50]. Actors can be trained with non-differentiable losses by introducing a differentiable surrogate[51] or critic[52] to predict losses that can be backpropagated to actor parameters. Alternatively, non-differentiable losses can be backpropagated to agent parameters if actions are sampled from a differentiable probability distribution[46,53] as training losses given by products of losses and sampling probabilities are differentiable. There are also a variety of alternatives to gradient descent, such as simulated annealing[54] and evolutionary algorithms[55], that do not require differentiable loss functions. Such alternatives can outperform gradient descent[56]; however, they usually achieve similar or lower performance than gradient descent for deep ANN training.

2 Training

In this section, we outline our training environment, ANN architecture and learning policy. Our ANNs were developed in Python with TensorFlow[57]. Detailed architecture and learning policy is in supplementary information. In addition, source code and pretrained models are openly accessible from GitHub[58], and training data is openly accessible[12,59].

2.1 Environment

To create partial scans from STEM images, an actor, μ, infers action unit vectors, $\mu(h_t)$, based on a history, $h_t = (o_1^i, a_1, ..., o_t, a_t)$, of previous actions, a, and observations, o. To encourage exploration, $\mu(h_t)$ is rotated to a_t by Ornstein-Uhlenbeck[60] (OU) exploration noise[61], ε_t,

$$a_t = \begin{bmatrix} \cos\varepsilon_t & -\sin\varepsilon_t \\ \sin\varepsilon_t & \cos\varepsilon_t \end{bmatrix} \mu(h_t) \tag{2}$$

$$\varepsilon_t = \theta(\varepsilon_{\text{avg}} - \varepsilon_{t-1}) + \sigma W \tag{3}$$

where we chose $\theta = 0.1$ to decay noise to $\varepsilon_{\text{avg}} = 0$, a scale factor, $\sigma = 0.2$, to scale a standard normal variate, W, and start noise $\varepsilon_0 = 0$. OU noise is linearly decayed to zero throughout training. Correlated OU exploration noise is recommended for continuous control tasks optimized by deep deterministic policy gradients[49] (DDPG) and recurrent deterministic policy gradients[50] (RDPG). Nevertheless, follow-up experiments with twin delayed deep deterministic policy gradients[62] (TD3) and distributed distributional deep deterministic policy gradients[63] (D4PG) have found that uncorrelated Gaussian noise can produce similar results.

An action, a_t, is the direction to move to observe a path segment, o_t, from the position at the end of the previous path segment. Partial scans are constructed from complete histories of actions and observations, h_T. A simplified partial scan is shown in figure 1. In our experiments, partial scans, s, are constructed from $T = 20$ straight path segments selected from 96×96 STEM images. Each segment has 20 probing positions separated by $d = 2^{1/2}$ px and positions can be outside an image. The pixels in the image nearest each probing position are sampled, so a separation of $d \geq 2^{1/2}$ simplied development by preventing successive probing positions in a segment from sampling the same pixel. A separation of $d < 2^{1/2}$ would allow a pixel to sampled more than once by moving diagonally, potentially incentivising orthogonal scan motion to sample more pixels.

Following our earlier work[16,23,64], we select subsets of pixels from STEM images to create partial scans to train ANNs for compressed sensing. Selecting a subset of pixels is easier than preparing a large, carefully partitioned and representative dataset[65,66] containing experimental partial scan and full image pairs, and selected pixels have realistic noise characteristics as they are from experimental images. However, selecting a subset of pixels does not account for probing location errors varying with scan shape[34]. We use a Warwick Electron Microscopy Dataset (WEMD) containing 19769 32-bit 96×96 images cropped and downsampled from full images[12,59]. Cropped images were blurred by a symmetric 5×5 Gaussian kernel with a 2.5 px

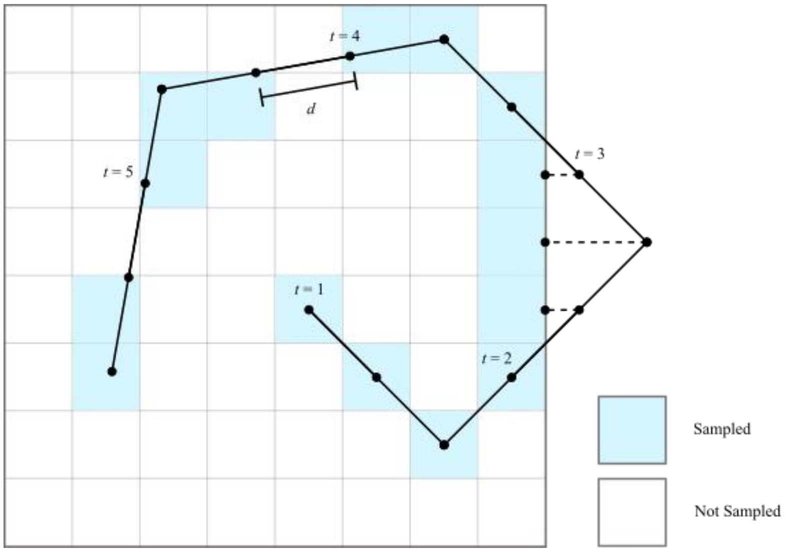


Figure 1. Example 8×8 partial scan with $T=5$ straight path segments. Each segment in this example has 3 probing positions separated by $d=2^{1/2}$ px and their starts are labelled by step numbers, t. Partial scans are selected from STEM images by sampling pixels nearest probing positions, even if the probing position is nominally outside an imaging region.

standard deviation to decrease any training loss variation due to varying noise characteristics. Finally, images, I, were linearly transformed to normalized images, I_N, with minimum and maximum values of -1 and 1. To test performance, the 19769 images were split, without shuffling, into a training set containing 15815 images and a test set containing 3954 images.

2.2 Architecture

For training, our adaptive scan system consists of an actor, μ, target actor, μ', critic, Q, target critic, Q', and generator, G. To minimize latency, our actors and critics are computationally inexpensive deep LSTMs[67] with a depth of 2 and 256 hidden units. Our generator is a convolutional neural network[68,69] (CNN). A recurrent actor selects actions, a_t and observes path segments, o_t, that are added to an experience replay[70], R, containing 10^5 sequences of actions and observations, $h_T=(o_1,a_1,...,o_T,a_T)$. Partial scans, s, are constructed from histories sampled from the replay to train a generator to complete partial scans, $I_G^i=G(s^i)$. The actor and generator cooperate to minimize generator losses, L_G, and are the only networks needed for inference.

Generator losses are not differentiable w.r.t. actor actions used to construct partial scans i.e. $\partial L_G/\partial a_t=0$. Following RDPG[50], we therefore introduce recurrent critics to predict losses from actor actions and observations that can be backpropagated to actors for training by BPTT. Actor and critic RNNs have the same architecture, except actors have two outputs to parameterize actions whereas critics have one output to predict losses. Target networks[49,71] use exponential moving averages of live actor and critic network parameters and are introduced to stabilize learning. For training by RDPG, live and target ANNs separately replay experiences. However, we propagate live RNN states to target RNNs at each step as a precaution against any cumulative divergence of target network behaviour from live network behaviour across multiple steps.

2.3 Learning Policy

To train actors to cooperate with a generator to complete partial scans, we developed cooperative recurrent deterministic policy gradients (CRDPG, algorithm 1). This is an extension of RDPG to an actor that cooperates with another ANN to minimize its loss. We train our networks by ADAM[72] optimized gradient descent for $M=10^6$ iterations with a batch size, $N=32$. We use constant learning rates $\eta_\mu=0.0005$ and $\eta_Q=0.0010$ for the actor and critic, respectively. For the generator, we use an initial learning rate $\eta_G=0.0030$ with an exponential decay factor of $0.75^{5m/M}$ at iteration m. The exponential decay envelope is multiplied by a sawtooth cyclic learning rate[73] with a period of $2M/9$ that oscillates between 0.2 and 1.0. Training takes two days with an Intel i7-6700 CPU and an Nvidia GTX 1080 Ti GPU.

We augment training data by a factor of eight by applying a random combination of flips and 90° rotations, mapping $s\rightarrow s'$ and $I_N\rightarrow I'_N$, similar to our earlier work[16,23,64,74]. Our generator is trained to minimize mean squared errors (MSEs),

$$L_G=\mathrm{MSE}(G(s'),I_N)\,, \tag{12}$$

between scan completions, $G(s')$, and normalized target images, I_N. Generator losses decrease during training as the generator learns, and may vary due to loss spikes[64], learning rate oscillations[73] or other training phenomena. Normalizing losses can

Algorithm 1. Cooperative recurrent deterministic policy gradients (CRDPG).

```
Initialize actor, $\mu$, critic, $Q$, and generator, $G$, networks with parameters $\omega$, $\theta$ and $\phi$, respectively.
Initialize target networks, $\mu'$ and $Q'$, with parameters $\omega' \leftarrow \omega$, $\theta' \leftarrow \theta$, respectively.
Initialize replay buffer, $R$.
Initialize average generator loss, $L_\text{avg}$.
for iteration $m = 1, M$ do
  Initialize empty history, $h_0$.
  for step $t = 1, T$ do
    Make observation, $o_t$.
    $h_t \leftarrow h_{t-1}, a_t, o_t$ (append action and corresponding observation to history).
    Select action, $a_t$, by computing $\mu(h_t)$ and applying exploration noise, $\epsilon_t$.
  end for
  Store the sequence $(o_1, a_1, ..., o_T, a_T)$ in $R$.
  Sample a minibatch of $N$ histories, $h_T^i = (o_1^i, a_1^i, ..., o_T^i, a_T^i)$, from $R$.
  Construct partial scans, $s^i$, from $h_T^i$.
  Use generator to complete partial scans, $I_G^i = G(s^i)$.
  Compute step losses, $(L_1^i, ..., L_T^i)$, from generator losses, $L_G^i$, and over edge losses, $E_t^i$,

      $L_t^i = E_t^i + \delta_{tT} \frac{\text{clip}(L_G^i)}{L_\text{avg}}$,        (4)

  where the Kronecker delta, $\delta_{tT}$, is 1 if $t = T$ and 0 otherwise, and clip($L_G^i$) is the smaller of $L_G^i$ and three standard deviations above its running mean.
  Compute target values, $(y_1^i, ..., y_T^i)$, with target networks,

      $y_t^i = L_t^i + \gamma Q'(H_Q^i, o_{t+1}^i, a_{t+1}^i, \mu'(H_\mu^i, o_{t+1}^i, a_{t+1}^i))$,        (5)

  where $H_Q^i$ and $H_\mu^i$ are states of live networks after computing $Q(h_t^i, a_t^i)$ and $\mu(h_t^i)$, respectively.
  Compute critic update (using BPTT),

      $\Delta\omega = \frac{1}{NT} \sum_i^N \sum_t^T (y_t^i - Q(h_t^i, a_t^i)) \frac{\partial Q(h_t^i, a_t^i)}{\partial \omega}$.        (6)

  Compute actor update (using BPTT),

      $\Delta\theta = \frac{1}{NT} \sum_i^N \sum_t^T \frac{\partial Q(h_t^i, a_t^i)}{\partial \mu(h_t^i)} \frac{\partial \mu(h_t^i)}{\partial \theta}$.        (7)

  Compute generator update,

      $\Delta\phi = \frac{1}{N} \sum_i^N \frac{\partial L_G^i}{\partial \phi}$.        (8)

  Update the actor, critic and generator by gradient descent.
  Update the target networks and average generator loss,

      $\omega' \leftarrow \beta_\omega \omega' + (1 - \beta_\omega)\omega$,        (9)
      $\theta' \leftarrow \beta_\theta \theta' + (1 - \beta_\theta)\theta$,        (10)
      $L_\text{avg} \leftarrow \beta_L L_\text{avg} + \frac{1 - \beta_L}{N} \sum_i^N (L_G^i)$.        (11)

end for
```

improve RL[75], so we divide generator losses used for critic training by their running mean,

$$L_{\text{avg}} \leftarrow \beta_L L_{\text{avg}} + \frac{1-\beta_L}{N} \sum_{i}^{N} L_G \,, \tag{13}$$

where we chose $\beta_L = 0.997$ and L_{avg} is updated at each training iteration.

Heuristically, an optimal policy does not go over image edges as there is no information there in our training environment. To accelerate convergence, we therefore added a small loss penalty, $E_t = 0.1$, at step t if an action results in a probing position being over an image edge. The total loss at each step is

$$L_t = E_t + \delta_{tT} \frac{\text{clip}(L_G)}{L_{\text{avg}}} \,, \tag{14}$$

where clip(L_G) clips losses used for RL to three standard deviations above their running mean. This adaptive loss clipping is inspired by adaptive learning rate clipping[64] (ALRC) and reduces learning destabilization by high loss spikes. However, we expect that clipping normalized losses to a fixed threshold[71] would achieve similar results. The Kronecker delta, δ_{tT}, in equation 14 is 1 if $t = T$ and 0 otherwise, so it only adds the generator loss at the final step, T.

To estimate discounted future losses, Q_t^{rl}, for RL, we use a target actor and critic,

$$Q_t^{\text{rl}} = L_t + \gamma Q'(h_{t+1}, \mu'(h_{t+1})) \,, \tag{15}$$

where we chose $\gamma = 0.97$. Target networks stabilize learning and decrease policy oscillations[76–78]. The critic is trained to minimize mean squared differences, L_Q, between predicted and target losses, and the actor is trained to minimize losses, L_μ, predicted by the critic,

$$L_Q = \frac{1}{2T} \sum_{t=1}^{T} (y_t - Q(h_t, a_t))^2 \,, \tag{16}$$

$$L_\mu = \frac{1}{T} \sum_{t=1}^{T} Q(h_t, a_t) \,. \tag{17}$$

Our target actor and critic have trainable parameters ω' and θ', respectively, that track live parameters, ω and θ, by soft updates[49],

$$\omega'_m = \beta_\omega \omega'_{m-1} + (1-\beta_\omega)\omega_m \,, \tag{18}$$

$$\theta'_m = \beta_\theta \theta'_{m-1} + (1-\beta_\theta)\theta_m \,, \tag{19}$$

where we chose $\beta_\omega = \beta_\theta = 0.9997$. We also investigated hard updates[71], where target networks are periodically copied from live networks; however, we found that soft updates result in faster convergence and more stable training.

3 Experiments

In this section, we present examples of adaptive partial scans and select learning curves for architecture and learning policy experiments. Examples of 1/23.04 px coverage partial scans, target outputs and generator completions are shown in figure 2 for 96×96 crops from test set STEM images. They show both adaptive and spiral scans after flips and rotations to augment data for the generator. The first actions select a path segment from the middle of image in the direction of a corner. Actors then use the first and following observations to inform where to sample the remaining $T - 1 = 19$ path segments. Actors adapt scan paths to specimens. For example, if an image contains regular atoms, an actor might cover a large area to see if there is a region where that changes. Alternatively, if an image contains a uniform region, actors, may explore near image edges and far away from the uniform region to find region boundaries.

The main limitation of our experiments is that generators trained to complete a variety of partial scan paths generated by an actor achieves lower performance than a generate trained to complete partial scans with a fixed path. For example, figure 3(a) shows that generators trained to cooperate with LSTM or GRU actors are outperformed by generators trained with fixed spiral or other scan paths shown in figure 3(b). Spiral paths outperform fixed scan paths; however, we emphasize that paths generated by actors are designed for individual training data, rather than all training data. Freezing actor training to prevent changes in actor policy does not result in clear improvements in generator performance. Consequently, we think that improvements to generator architecture or learning policy should be a starting point for further investigation. To find the best practical actor

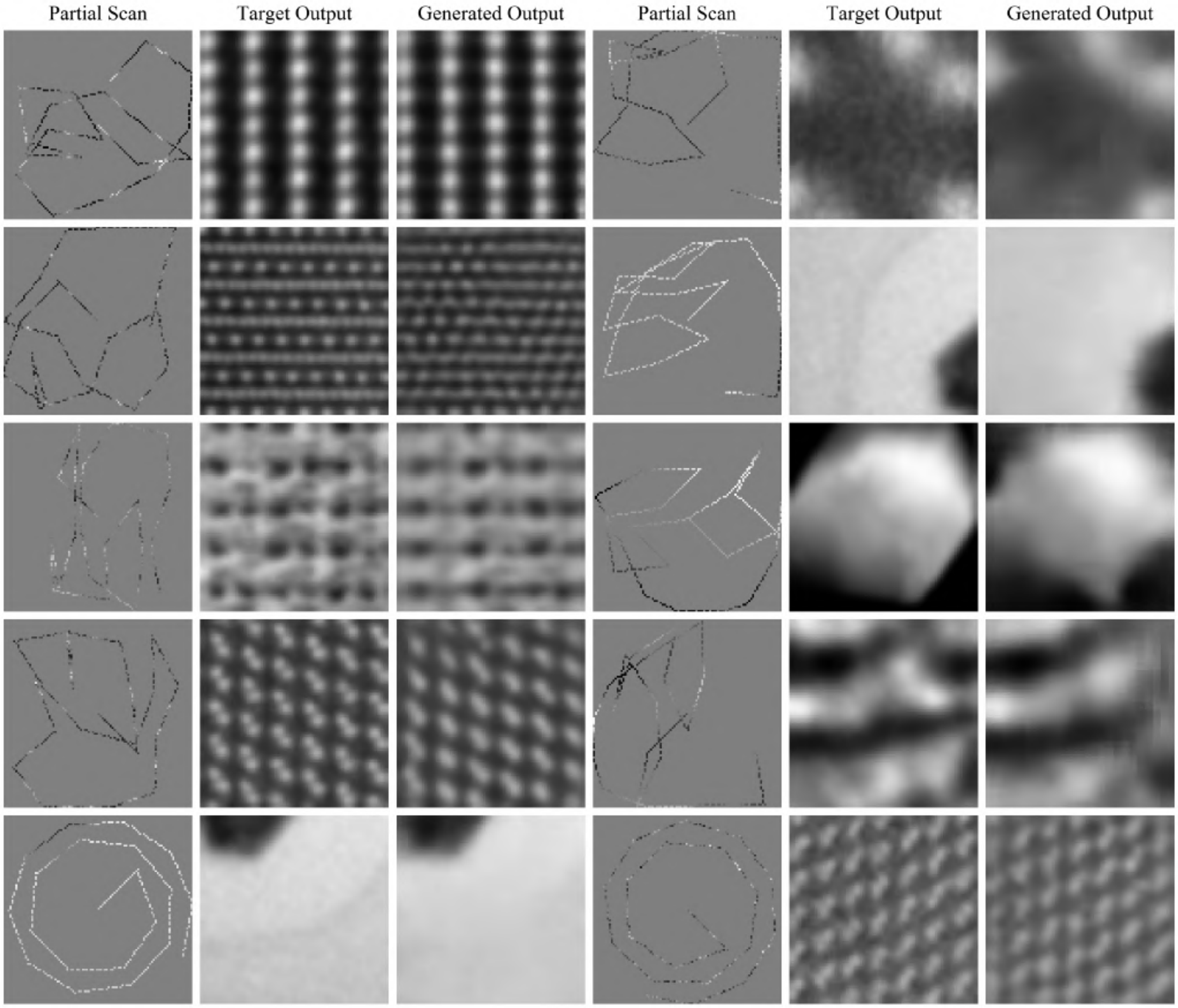


Figure 2. Test set 1/23.04 px coverage partial scans, target outputs and generated partial scan completions for 96×96 crops from STEM images. The top four rows show adaptive scans, and the bottom row shows spiral scans. Input partial scans are noisy, whereas target outputs are blurred.

policy, we think that a generator trained for a variety of scan paths should achieve comparable performance to generators trained for single scan paths.

We investigated a variety of popular RNN architectures to minimize inference time. Learning curves in figure 3(a) show that performance is similar for LSTMs and GRUs. GRUs require less computation. However, LSTM and GRU inference time is comparable and GRU training seems to be more prone to loss spikes, so LSTMs may be preferable. We also created a DNC by augmenting a deep LSTM with dynamic external memory. However, figure 3(c) shows that LSTM and DNC performance is similar, and inference time and computational requirements are much higher for our DNC. We tried to reduce computation and accelerate convergence by applying projection layers to LSTM hidden states[79]. However, we found that performance decreased with decreasing projection layer size.

Experienced replay buffers for RL often have heuristic sizes, such as 10^6 examples. However, RL can be sensitive to replay buffer size[70]. Indeed, learning curves in figure 3(d) show that increasing buffer size improves learning stability and decreases test set errors. Increasing buffer size usually improves learning stability and decreases forgetting by exposing actors and critics to a higher variety of past policies. However, we expect that convergence would be slowed if the buffer became too large as increasing buffer size increases expected time before experiences with new policies are replayed. We also found that increasing

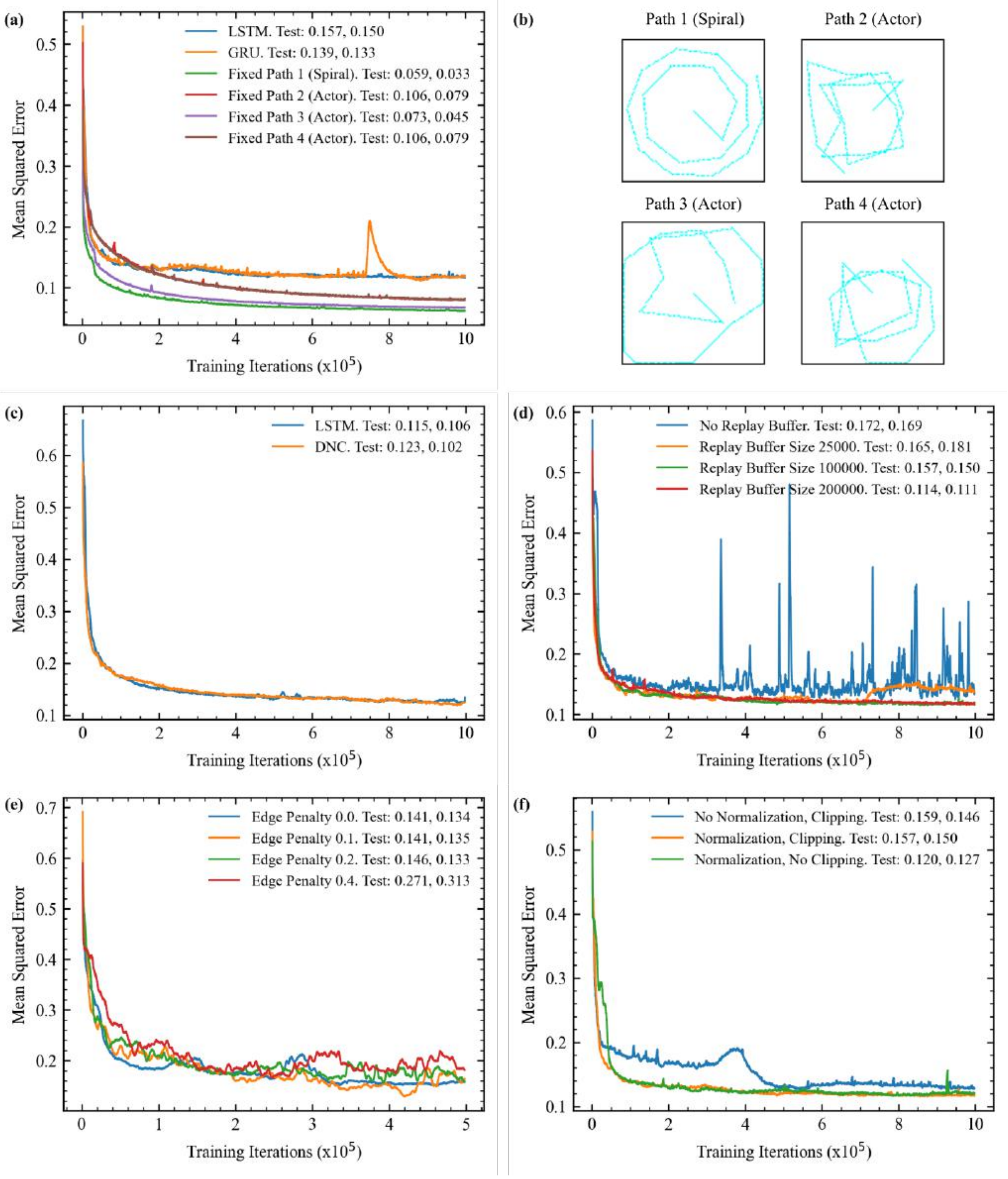


Figure 3. Learning curves for a-b) adaptive scan paths chosen by an LSTM or GRU, and fixed spiral and other fixed paths, c) adaptive paths chosen by an LSTM or DNC, d) a range of replay buffer sizes, e) a range of penalties for trying to sample at probing positions over image edges, and f) with and without normalizing or clipping generator losses used for critic training. All learning curves are 2500 iteration boxcar averaged and results in different plots are not directly comparable due to varying experiment settings. Means and standard deviations of test set errors, "Test: Mean, Std Dev", are at the ends of labels in graph legends.

buffer sized decreased the size of small loss oscillations[76–78], which have a period near 2000 iterations. However, the size of loss oscillations does not appear to affect performance.

We found that initial convergence is usually delayed if a large portion of initial actions go outside the imaging region. This would often delay convergence by about 10^4 iterations before OU noise led to the discovery of better exploration strategies away from image edges. Although 10^4 iterations is only 1% of our 10^6 iteration learning policy, it often impaired development by delaying debugging or evaluation of changes to architecture and learning policy. Augmenting RL losses with subgoal-based heuristic rewards can accelerate convergence by making problems more tractable[80]. Thus, we added loss penalties if actors tried to go over image edges, which accelerated initial convergence. Learning curves in figure 3(e) show that over edge penalties at each step smaller than $E_t = 0.2$ have a similar effect on performance. Further, performance is lower for higher over edge penalties, $E_t \geq 0.2$. We also found that training is more stable if over edge penalties are added at individual steps, rather than propagated to past steps as part of a discounted future loss.

Our actor, critic and generator are trained together. It follows that generator losses, which our critic learns to predict, decrease throughout training as generator performance improves. However, normalizing loss sizes usually improves RL[75], so we divide by their running means in equation 14. Learning curves in figure 3(f) show that loss normalization improves learning stability and decreases final errors. Clipping training losses can improve RL[71], so we clipped generator losses used for critic training to 3 standard deviations above their running means. We found that clipping increases test set errors, possibly because most training errors are in a similar regime. Thus, we expect that clipping may be more helpful for training with sparser scans as higher uncertainty may increase likelihood of unusually high generator losses.

4 Discussion

The main limitation of our adaptive scan system is that generator errors are much higher when a generator is trained for a variety of scan paths than when it is trained for a single scan path. However, we expect that generator performance for a variety of scans could be improved to match performance for single scans by developing a larger neural network with a better learning policy. To train actors to cooperate with generators, we developed CRDPG. This is an extension of RDPG[50], and RDPG is based on DDPG[49]. Alternatives to DDPG, such as TD3[62] and D4PG[63], arguably achieve higher performance, so we expect that they could form the basis of a future training algorithm. Further, we expect that architecture and learning policy could be improved by AdaNet[81], Ludwig[82], or other automatic machine learning[83–87] (AutoML) algorithms as AutoML can often match or surpass the performance of human developers[88,89]. Finally, test set losses for a variety of scans appear to be decreasing at the end of training, so we expect that performance could be improved by increasing training iterations.

After generator performance is improved, we expect the main limitation of our adaptive scan system to be distortions caused by probing position errors. Errors usually depend on scan path shape[34] and accumulate for each path segment. Non-linear scan distortions can be corrected by comparing pairs of orthogonal raster scans[90,91], and we expect this method can be extended to partial scans. However, orthogonal scanning would complicate measurement by limiting scan paths to two half scans to avoid doubling electron dose on beam-sensitive materials. Instead, we propose that a cyclic generator[92] could be trained to correct scan distortions and provide a detailed method as supplementary information[93]. Another limitation is that our generators do not learn to correct STEM noise[94]. However, we expect that generators can learn to remove noise, for example, from single noisy examples[95] or by supervised learning[74].

To simplify our preliminary investigation, our scan system samples straight path segments and cannot go outside a specified imaging region. However, actors could learn to output actions with additional degrees of freedom to describe curves, multiple successive path segments, or sequences of non-contiguous probing positions. Similarly, additional restrictions could be applied to actions. For example, actions could be restricted to avoid actions that cause high probing position errors. Training environments could also be modified to allow actors to sample pixels over image edges by loading images larger than partial scan regions. In practice, actors can sample outside a scan region and being able to access extra information outside an imaging region could improve performance. However, using larger images may slow development by increasing data loading and processing times.

Not all scan systems support non-raster scan paths. However, many scan controllers can be augmented with an FPGA to enable custom scan paths[34,35]. Recent versions of Gatan DigitalMicrograph support Python[96], so our ANNs can be readily integrated into existing scan systems. Alternatively, an actor could be synthesized on a scan-controlling FPGA[97,98] to minimize inference time. There could be hundreds of path segments in a partial scan, so computationally lightweight and parallelizable actors are essential to minimize scan time. We have therefore developed actors based computationally inexpensive RNNs, which can remember state information to inform future decisions. Another approach is to update a partial scan at each step to be input to feedforward neural network (FNN), such as a CNN, to decide actions. However, we expect that FNNs are less practical than RNNs as FNNs may require additional computation to reprocess all past states at each step.

5 Conclusions

Our initial investigation demonstrates that actor RNNs can be trained by RL to direct piecewise adaption of contiguous scans to specimens for compressed sensing. We introduce CRDPG to train an RNN to cooperate with a CNN to complete STEM images from partial scans and present our learning policy, experiments, and example applications. After further development, we expect that adaptive scans will become the most effective approach to decrease electron beam damage and scan time with minimal information loss. Static sampling strategies are a subset of possible dynamic sampling strategies, so the performance of static sampling can always be matched by or outperformed by dynamic sampling. Further, we expect that adaptive scan systems can be developed for most areas of science and technology, including for the reduction of medical radiation. To encourage further investigation, our source code, pretrained models, and training data is openly accessible.

6 Supplementary Information

Supplementary information is openly accessible at `https://doi.org/10.5281/zenodo.4384708`. Therein, we present detailed ANN architecture, additional experiments and example scans, and a new method to correct partial scan distortions.

Data Availability

The data that support the findings of this study are openly available.

Acknowledgements

Thanks go to Jasmine Clayton, Abdul Mohammed, and Jeremy Sloan for internal review. The author acknowledges funding from EPSRC grant EP/N035437/1 and EPSRC Studentship 1917382.

Competing Interests

The author declares no competing interests.

References

1. Krull, A., Hirsch, P., Rother, C., Schiffrin, A. & Krull, C. Artificial-Intelligence-Driven Scanning Probe Microscopy. *Commun. Phys.* **3**, 1–8 (2020).
2. Rugar, D. & Hansma, P. Atomic Force Microscopy. *Phys. Today* **43**, 23–30 (1990).
3. New, P. F., Scott, W. R., Schnur, J. A., Davis, K. R. & Taveras, J. M. Computerized Axial Tomography with the EMI Scanner. *Radiology* **110**, 109–123 (1974).
4. Heymsfield, S. B. *et al.* Accurate Measurement of Liver, Kidney, and Spleen Volume and Mass by Computerized Axial Tomography. *Annals Intern. Medicine* **90**, 185–187 (1979).
5. Schwartz, A. J., Kumar, M., Adams, B. L. & Field, D. P. *Electron Backscatter Diffraction in Materials Science*, vol. 2 (Springer, 2009).
6. Vernon-Parry, K. D. Scanning Electron Microscopy: An Introduction. *III-Vs Rev.* **13**, 40–44 (2000).
7. Keren, S. *et al.* Noninvasive Molecular Imaging of Small Living Subjects Using Raman Spectroscopy. *Proc. Natl. Acad. Sci.* **105**, 5844–5849 (2008).
8. Tong, Y.-X., Zhang, Q.-H. & Gu, L. Scanning Transmission Electron Microscopy: A Review of High Angle Annular Dark Field and Annular Bright Field Imaging and Applications in Lithium-Ion Batteries. *Chin. Phys. B* **27**, 066107 (2018).
9. Scarborough, N. M. *et al.* Dynamic X-Ray Diffraction Sampling for Protein Crystal Positioning. *J. Synchrotron Radiat.* **24**, 188–195 (2017).
10. Hujsak, K., Myers, B. D., Roth, E., Li, Y. & Dravid, V. P. Suppressing Electron Exposure Artifacts: An Electron Scanning Paradigm with Bayesian Machine Learning. *Microsc. Microanal.* **22**, 778–788 (2016).
11. Egerton, R. F., Li, P. & Malac, M. Radiation Damage in the TEM and SEM. *Micron* **35**, 399–409 (2004).
12. Ede, J. M. Warwick Electron Microscopy Datasets. *Mach. Learn. Sci. Technol.* **1**, 045003 (2020).
13. Amidror, I. Sub-Nyquist Artefacts and Sampling Moiré Effects. *Royal Soc. Open Sci.* **2**, 140550 (2015).
14. Binev, P. *et al.* Compressed Sensing and Electron Microscopy. In *Modeling Nanoscale Imaging in Electron Microscopy*, 73–126 (Springer, 2012).

15. Ede, J. M. Review: Deep Learning in Electron Microscopy. *arXiv preprint arXiv:2009.08328* (2020).

16. Ede, J. M. & Beanland, R. Partial Scanning Transmission Electron Microscopy with Deep Learning. *arXiv preprint arXiv:1910.10467* (2020).

17. Li, Y. Deep Reinforcement Learning: An Overview. *arXiv preprint arXiv:1701.07274* (2017).

18. Hwang, S., Han, C. W., Venkatakrishnan, S. V., Bouman, C. A. & Ortalan, V. Towards the Low-Dose Characterization of Beam Sensitive Nanostructures via Implementation of Sparse Image Acquisition in Scanning Transmission Electron Microscopy. *Meas. Sci. Technol.* **28**, 045402 (2017).

19. Hujsak, K., Myers, B. D., Roth, E., Li, Y. & Dravid, V. P. Suppressing Electron Exposure Artifacts: An Electron Scanning Paradigm with Bayesian Machine Learning. *Microsc. Microanal.* **22**, 778–788 (2016).

20. Anderson, H. S., Ilic-Helms, J., Rohrer, B., Wheeler, J. & Larson, K. Sparse Imaging for Fast Electron Microscopy. In *Computational Imaging XI*, vol. 8657, 86570C (International Society for Optics and Photonics, 2013).

21. Fang, L. *et al.* Deep Learning-Based Point-Scanning Super-Resolution Imaging. *bioRxiv* 740548 (2019).

22. de Haan, K., Ballard, Z. S., Rivenson, Y., Wu, Y. & Ozcan, A. Resolution Enhancement in Scanning Electron Microscopy Using Deep Learning. *Sci. Reports* **9**, 1–7 (2019).

23. Ede, J. M. Deep Learning Supersampled Scanning Transmission Electron Microscopy. *arXiv preprint arXiv:1910.10467* (2019).

24. Mueller, K. Selection of Optimal Views for Computed Tomography Reconstruction (2011). US Patent App. 12/842,274.

25. Wang, Z. & Arce, G. R. Variable Density Compressed Image Sampling. *IEEE Transactions on Image Process.* **19**, 264–270 (2009).

26. Ji, S., Xue, Y. & Carin, L. Bayesian Compressive Sensing. *IEEE Transactions on Signal Process.* **56**, 2346–2356 (2008).

27. Seeger, M. W. & Nickisch, H. Compressed Sensing and Bayesian Experimental Design. In *Proceedings of the 25th International Conference on Machine Learning*, 912–919 (2008).

28. Braun, G., Pokutta, S. & Xie, Y. Info-Greedy Sequential Adaptive Compressed Sensing. *IEEE J. Sel. Top. Signal Process.* **9**, 601–611 (2015).

29. Carson, W. R., Chen, M., Rodrigues, M. R., Calderbank, R. & Carin, L. Communications-Inspired Projection Design with Application to Compressive Sensing. *SIAM J. on Imaging Sci.* **5**, 1185–1212 (2012).

30. Godaliyadda, G. D. P. *et al.* A Framework for Dynamic Image Sampling Based on Supervised Learning. *IEEE Transactions on Comput. Imaging* **4**, 1–16 (2017).

31. Ermeydan, E. S. & Cankaya, I. Sparse Fast Fourier Transform for Exactly Sparse Signals and Signals with Additive Gaussian Noise. *Signal, Image Video Process.* **12**, 445–452 (2018).

32. Saldi, N., Yüksel, S. & Linder, T. Asymptotic Optimality of Finite Model Approximations for Partially Observed Markov Decision Processes With Discounted Cost. *IEEE Transactions on Autom. Control.* **65**, 130–142 (2019).

33. Jaakkola, T., Singh, S. P. & Jordan, M. I. Reinforcement Learning Algorithm for Partially Observable Markov Decision Problems. In *Advances in Neural Information Processing Systems*, 345–352 (1995).

34. Sang, X. *et al.* Dynamic Scan Control in STEM: Spiral Scans. *Adv. Struct. Chem. Imaging* **2**, 6 (2017).

35. Sang, X. *et al.* Precision Controlled Atomic Resolution Scanning Transmission Electron Microscopy Using Spiral Scan Pathways. *Sci. Reports* **7**, 43585 (2017).

36. Hochreiter, S. & Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **9**, 1735–1780 (1997).

37. Olah, C. Understanding LSTM Networks. Online: https://colah.github.io/posts/2015-08-Understanding-LSTMs (2015).

38. Cho, K. *et al.* Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv preprint arXiv:1406.1078* (2014).

39. Weiss, G., Goldberg, Y. & Yahav, E. On the Practical Computational Power of Finite Precision RNNs for Language Recognition. *arXiv preprint arXiv:1805.04908* (2018).

40. Jozefowicz, R., Zaremba, W. & Sutskever, I. An Empirical Exploration of Recurrent Network Architectures. In *International Conference on Machine Learning*, 2342–2350 (2015).

41. Bayer, J., Wierstra, D., Togelius, J. & Schmidhuber, J. Evolving Memory Cell Structures for Sequence Learning. In *International Conference on Artificial Neural Networks*, 755–764 (Springer, 2009).

42. Pascanu, R., Mikolov, T. & Bengio, Y. On the Difficulty of Training Recurrent Neural Networks. In *International Conference on Machine Learning*, 1310–1318 (2013).

43. Graves, A. *et al.* Hybrid Computing Using a Neural Network with Dynamic External Memory. *Nature* **538**, 471–476 (2016).

44. Werbos, P. J. Backpropagation Through Time: What It Does and How To Do It. *Proc. IEEE* **78**, 1550–1560 (1990).

45. Ruder, S. An Overview of Gradient Descent Optimization Algorithms. *arXiv preprint arXiv:1609.04747* (2016).

46. Mnih, V., Heess, N., Graves, A. & Kavukcuoglu, K. Recurrent Models of Visual Attention. In *Advances in Neural Information Processing Systems*, 2204–2212 (2014).

47. Ba, J., Mnih, V. & Kavukcuoglu, K. Multiple Object Recognition with Visual Attention. *arXiv preprint arXiv:1412.7755* (2014).

48. Vinyals, O. *et al.* AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/ (2019).

49. Lillicrap, T. P. *et al.* Continuous Control with Deep Reinforcement Learning. *arXiv preprint arXiv:1509.02971* (2015).

50. Heess, N., Hunt, J. J., Lillicrap, T. P. & Silver, D. Memory-Based Control with Recurrent Neural Networks. *arXiv preprint arXiv:1512.04455* (2015).

51. Grabocka, J., Scholz, R. & Schmidt-Thieme, L. Learning Surrogate Losses. *arXiv preprint arXiv:1905.10108* (2019).

52. Konda, V. R. & Tsitsiklis, J. N. Actor-Critic Algorithms. In *Advances in Neural Information Processing Systems*, 1008–1014 (2000).

53. Zhao, T., Hachiya, H., Niu, G. & Sugiyama, M. Analysis and Improvement of Policy Gradient Estimation. In *Advances in Neural Information Processing Systems*, 262–270 (2011).

54. Rere, L. R., Fanany, M. I. & Arymurthy, A. M. Simulated Annealing Algorithm for Deep Learning. *Procedia Comput. Sci.* **72**, 137–144 (2015).

55. Young, S. R., Rose, D. C., Karnowski, T. P., Lim, S.-H. & Patton, R. M. Optimizing Deep Learning Hyper-Parameters Through an Evolutionary Algorithm. In *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, 1–5 (2015).

56. Such, F. P. *et al.* Deep Neuroevolution: Genetic Algorithms are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. *arXiv preprint arXiv:1712.06567* (2017).

57. Abadi, M. *et al.* TensorFlow: A System for Large-Scale Machine Learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 265–283 (2016).

58. Ede, J. M. Adaptive Partial STEM Repository. Online: https://github.com/Jeffrey-Ede/adaptive-scans (2020).

59. Ede, J. M. & Beanland, R. Electron Microscopy Datasets. Online: https://github.com/Jeffrey-Ede/datasets/wiki (2020).

60. Uhlenbeck, G. E. & Ornstein, L. S. On the Theory of the Brownian Motion. *Phys. Rev.* **36**, 823 (1930).

61. Plappert, M. *et al.* Parameter Space Noise for Exploration. *arXiv preprint arXiv:1706.01905* (2017).

62. Fujimoto, S., Van Hoof, H. & Meger, D. Addressing Function Approximation Error in Actor-Critic Methods. *arXiv preprint arXiv:1802.09477* (2018).

63. Barth-Maron, G. *et al.* Distributed Distributional Deterministic Policy Gradients. *arXiv preprint arXiv:1804.08617* (2018).

64. Ede, J. M. & Beanland, R. Adaptive Learning Rate Clipping Stabilizes Learning. *Mach. Learn. Sci. Technol.* **1**, 015011 (2020).

65. Raschka, S. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. *arXiv preprint arXiv:1811.12808* (2018).

66. Roh, Y., Heo, G. & Whang, S. E. A Survey on Data Collection for Machine Learning: A Big Data-AI Integration Perspective. *IEEE Transactions on Knowl. Data Eng.* (2019).

67. Zaremba, W., Sutskever, I. & Vinyals, O. Recurrent Neural Network Regularization. *arXiv preprint arXiv:1409.2329* (2014).

68. McCann, M. T., Jin, K. H. & Unser, M. Convolutional Neural Networks for Inverse Problems in Imaging: A Review. *IEEE Signal Process. Mag.* **34**, 85–95 (2017).

69. Krizhevsky, A., Sutskever, I. & Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, 1097–1105 (2012).

70. Zhang, S. & Sutton, R. S. A Deeper Look at Experience Replay. *arXiv preprint arXiv:1712.01275* (2017).

71. Mnih, V. *et al.* Human-Level Control Through Deep Reinforcement Learning. *Nature* **518**, 529–533 (2015).

72. Kingma, D. P. & Ba, J. ADAM: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* (2014).

73. Smith, L. N. Cyclical Learning Rates for Training Neural Networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 464–472 (IEEE, 2017).

74. Ede, J. M. & Beanland, R. Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder. *Ultramicroscopy* **202**, 18–25 (2019).

75. van Hasselt, H. P., Guez, A., Hessel, M., Mnih, V. & Silver, D. Learning Values Across Many Orders of Magnitude. In *Advances in Neural Information Processing Systems*, 4287–4295 (2016).

76. Czarnecki, W. M. *et al.* Distilling Policy Distillation. *arXiv preprint arXiv:1902.02186* (2019).

77. Lipton, Z. C. *et al.* Combating Reinforcement Learning's Sisyphean Curse with Intrinsic Fear. *arXiv preprint arXiv:1611.01211* (2016).

78. Wagner, P. A Reinterpretation of the Policy Oscillation Phenomenon in Approximate Policy Iteration. In *Advances in Neural Information Processing Systems*, 2573–2581 (2011).

79. Jia, Y., Wu, Z., Xu, Y., Ke, D. & Su, K. Long Short-Term Memory Projection Recurrent Neural Network Architectures for Piano's Continuous Note Recognition. *J. Robotics* **2017** (2017).

80. Ng, A. Y., Harada, D. & Russell, S. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *International Conference on Machine Learning*, vol. 99, 278–287 (1999).

81. Weill, C. *et al.* AdaNet: A Scalable and Flexible Framework for Automatically Learning Ensembles. *arXiv preprint arXiv:1905.00080* (2019).

82. Molino, P., Dudin, Y. & Miryala, S. S. Ludwig: A Type-Based Declarative Deep Learning Toolbox. *arXiv preprint arXiv:1909.07930* (2019).

83. He, X., Zhao, K. & Chu, X. AutoML: A Survey of the State-of-the-Art. *arXiv preprint arXiv:1908.00709* (2019).

84. Malekhosseini, E., Hajabdollahi, M., Karimi, N. & Samavi, S. Modeling Neural Architecture Search Methods for Deep Networks. *arXiv preprint arXiv:1912.13183* (2019).

85. Jaafra, Y., Laurent, J. L., Deruyver, A. & Naceur, M. S. Reinforcement Learning for Neural Architecture Search: A Review. *Image Vis. Comput.* **89**, 57–66 (2019).

86. Elsken, T., Metzen, J. H. & Hutter, F. Neural Architecture Search: A Survey. *arXiv preprint arXiv:1808.05377* (2018).

87. Waring, J., Lindvall, C. & Umeton, R. Automated Machine Learning: Review of the State-of-the-Art and Opportunities for Healthcare. *Artif. Intell. Medicine* 101822 (2020).

88. Hanussek, M., Blohm, M. & Kintz, M. Can AutoML Outperform Humans? An Evaluation on Popular OpenML Datasets Using AutoML Benchmark. *arXiv preprint arXiv:2009.01564* (2020).

89. Zoph, B., Vasudevan, V., Shlens, J. & Le, Q. V. Learning Transferable Architectures for Scalable Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8697–8710 (2018).

90. Ophus, C., Ciston, J. & Nelson, C. T. Correcting Nonlinear Drift Distortion of Scanning Probe and Scanning Transmission Electron Microscopies from Image Pairs with Orthogonal Scan Directions. *Ultramicroscopy* **162**, 1–9 (2016).

91. Ning, S. *et al.* Scanning Distortion Correction in STEM Images. *Ultramicroscopy* **184**, 274–283 (2018).

92. Zhu, J.-Y., Park, T., Isola, P. & Efros, A. A. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2223–2232 (2017).

93. Ede, J. M. Supplementary Information: Adaptive Partial Scanning Transmission Electron Microscopy with Reinforcement Learning. Zenodo, Online: https://doi.org/10.5281/zenodo.4384708 (2020).

94. Seki, T., Ikuhara, Y. & Shibata, N. Theoretical Framework of Statistical Noise in Scanning Transmission Electron Microscopy. *Ultramicroscopy* **193**, 118–125 (2018).

95. Laine, S., Karras, T., Lehtinen, J. & Aila, T. High-Quality Self-Supervised Deep Image Denoising. In *Advances in Neural Information Processing Systems*, 6968–6978 (2019).

96. Miller, B. & Mick, S. Real-Time Data Processing Using Python in DigitalMicrograph. *Microsc. Microanal.* **25**, 234–235 (2019).

97. Noronha, D. H., Salehpour, B. & Wilton, S. J. LeFlow: Enabling Flexible FPGA High-Level Synthesis of TensorFlow Deep Neural Networks. In *FSP Workshop 2018; Fifth International Workshop on FPGAs for Software Programmers*, 1–8 (VDE, 2018).

98. Ruan, A., Shi, A., Qin, L., Xu, S. & Zhao, Y. A Reinforcement Learning Based Markov-Decision Process (MDP) Implementation for SRAM FPGAs. *IEEE Transactions on Circuits Syst. II: Express Briefs* (2019).

Supplementary Information: Adaptive Partial Scanning Transmission Electron Microscopy with Reinforcement Learning

Jeffrey M. Ede[1,a]

[1]University of Warwick, Department of Physics, Coventry, CV4 7AL, UK
[a]j.m.ede@warwick.ac.uk

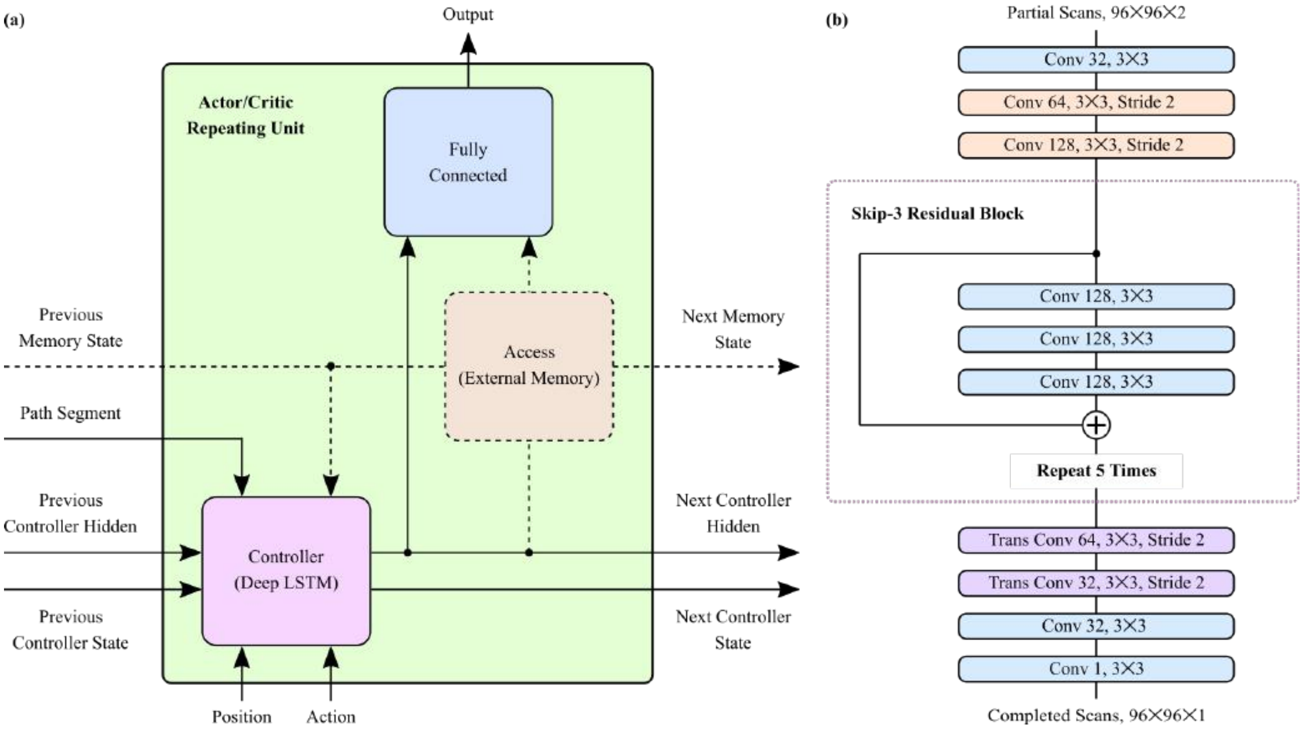


Figure S1. Actor, critic and generator architecture. a) An actor outputs action vectors whereas a critic predicts losses. Dashed lines are for extra components in a DNC. b) A convolutional generator completes partial scans.

S1 Detailed Architecture

Detailed actor, critic and generator architecture is shown in figure S1. Actors and critics have almost identical architecture, except actor fully connected layers output action vectors whereas critic fully connected layers output predicted losses. In most of our experiments, actors and critics are deep LSTMs[1]. However, we also augment deep LSTMs with dynamic external memory to create DNCs[2] in some of our experiments. Configuration details of actor and critic components shown in figure S1(a) follow.

Controller (Deep LSTM): A two-layer deep LSTM with 256 hidden units in each layer. To reduce signal attenuation, we add skip connections from inputs to the second LSTM layer and from the first LSTM layer to outputs. Weights are initialized from truncated normal distributions and biases are zero initialized. In addition, we add a bias of 1 to the forget gate to reduce forgetting at the start of training[3]. Initial LSTM cell and hidden states are initialized with trainable variables[4].

Access (External Memory): Our DNC implementation is adapted from Google Deepmind's[2,5]. We use 4 read heads and 1 write head to control access to dynamic external memory, which has 16 slots with a word size of 64.

Fully Connected: A dense layer linearly connects inputs to outputs. Weights are initialized from a truncated normal distribution and there are no biases.

Conv *d*, *w*x*w*, Stride, *x*: Convolutional layer with a square kernel of width, w, that outputs d feature channels. If the stride is specified, convolutions are only applied to every xth spatial element of their input, rather than to every element. Striding is not applied depthwise.

Trans Conv *d*, *w*x*w*, Stride, *x*: Transpositional convolutional layer with a square kernel of width, w, that outputs d feature channels. If the stride is specified, convolutions are only applied to every xth spatial element of their input, rather than to every element. Striding is not applied depthwise.

⊕: Circled plus signs indicate residual connections where incoming tensors are added together. Residuals help reduce signal attenuation and allow a network to learn perturbative transformations more easily.

The actor and critic cooperate with a convolutional generator, shown in figure S1(b), to complete partial scans. Our generator is constructed from convolutional layers[6] and skip-3 residual blocks[7]. Each convolutional layer is followed by ReLU[8] activation then batch normalization[9], and residual connections are added between activation and batch normalization. The convolutional weights are Xavier[10] initialized and biases are zero initialized.

S2 Additional Regularization

We apply L2 regularization[11] to decay generator parameters by a factor, $\beta = 0.99999$, at each training iteration. This decay rate is heuristic and the L2 regularization is primarily a precaution against overfitting. Further, adding L2 regularization did not have a noticeable effect on performance. We also investigated gradient clipping[12–15] to a range of static and dynamic thresholds for actor and critic training. However, we found that gradient clipping decreases convergence if clipping thresholds are too small and otherwise does not have a noticeable effect.

S3 Additional Experiments

This section present additional learning curves for architecture and learning policy experiments in figure S2. For example, learning curves in figure S2(a) show that generator training with an exponentially decayed cyclic learning rate[16] results in faster convergence and lower final errors than just using an exponentially decayed learning rate. We were concerned that a cyclic learning rate might cause generator loss oscillations if the learning rate oscillated too high. Indeed, our investigation of loss normalization was, in part, to prevent potential generator loss oscillations from destabilizing critic training. However, our learning policy results in generator losses that steadily decay throughout training.

To train actors by BPTT, we differentiate losses predicted by critics w.r.t. actor parameters by the chain rule,

$$\Delta\theta = \frac{1}{NT}\sum_{i}^{N}\sum_{t}^{T}\frac{\partial Q(h_t^i, a_t^i)}{\partial\theta} = \frac{1}{NT}\sum_{i}^{N}\sum_{t}^{T}\frac{\partial Q(h_t^i, a_t^i)}{\partial\mu(h_t^i)}\frac{\partial\mu(h_t^i)}{\partial\theta}\,. \tag{S1}$$

An alternative approach is to replace $\partial Q(h_t^i, a_t^i)/\partial\mu(h_t^i)$ with a derivative w.r.t. replayed actions, $\partial Q(h_t^i, a_t^i)/\partial a_t^i$. This is equivalent to adding noise, $\text{stop\_gradient}(a_t^i - \mu(h_t^i))$, to an actor action, $\mu(h_t^i)$, where $\text{stop\_gradient}(x)$ is a function that stops gradient backpropagation to x. However, learning curves in figure S2(b) show that differentiation w.r.t. live actor actions results in faster convergence to lower losses. Results for $\partial Q(h_t^i, a_t^i)/\partial a_t^i$ are similar if OU exploration noise is doubled.

Most STEM signals are imaged at several times their Nyquist rates[17]. To investigate adaptive STEM performance on signals imaged close to their Nyquist rates, we downsampled STEM images to 96×96. Learning curves in figure S2(c) show that losses are lower for oversampled STEM crops. Following, we investigated if MSEs vary for training with different loss metrics by adding a Sobel loss, $\lambda_S L_S$, to generator losses. Our Sobel loss is

$$L_S = \text{MSE}(S(G(s)), S(I_N))\,, \tag{S2}$$

where $S(x)$ computes a channelwise concatenation of horizontal and vertical Sobel derivatives[18] of x, and we chose $\lambda_S = 0.1$ to weight the contribution of L_S to the total generator loss, $L_G + \lambda_S L_S$. Learning curves in figure S2(c) show that Sobel losses do not decrease training MSEs for STEM crops. However, Sobel losses decrease MSEs for downsampled STEM images. This motivates the exploration of alternative loss functions[19] to further improve performance. For example, our earlier work shows that generator training as part of a generative adversarial network[20–23] (GAN) can improve STEM image realism[24]. Similarly, we expect that generated image realism could be improved by training generators with perceptual losses[25].

After we found that adding a Sobel loss can decrease MSEs, we also experimented with other loss functions, such as the maximum MSE of 5×5 regions. Learning curves in figure S2(d) show that MSEs result in faster convergence than maximum

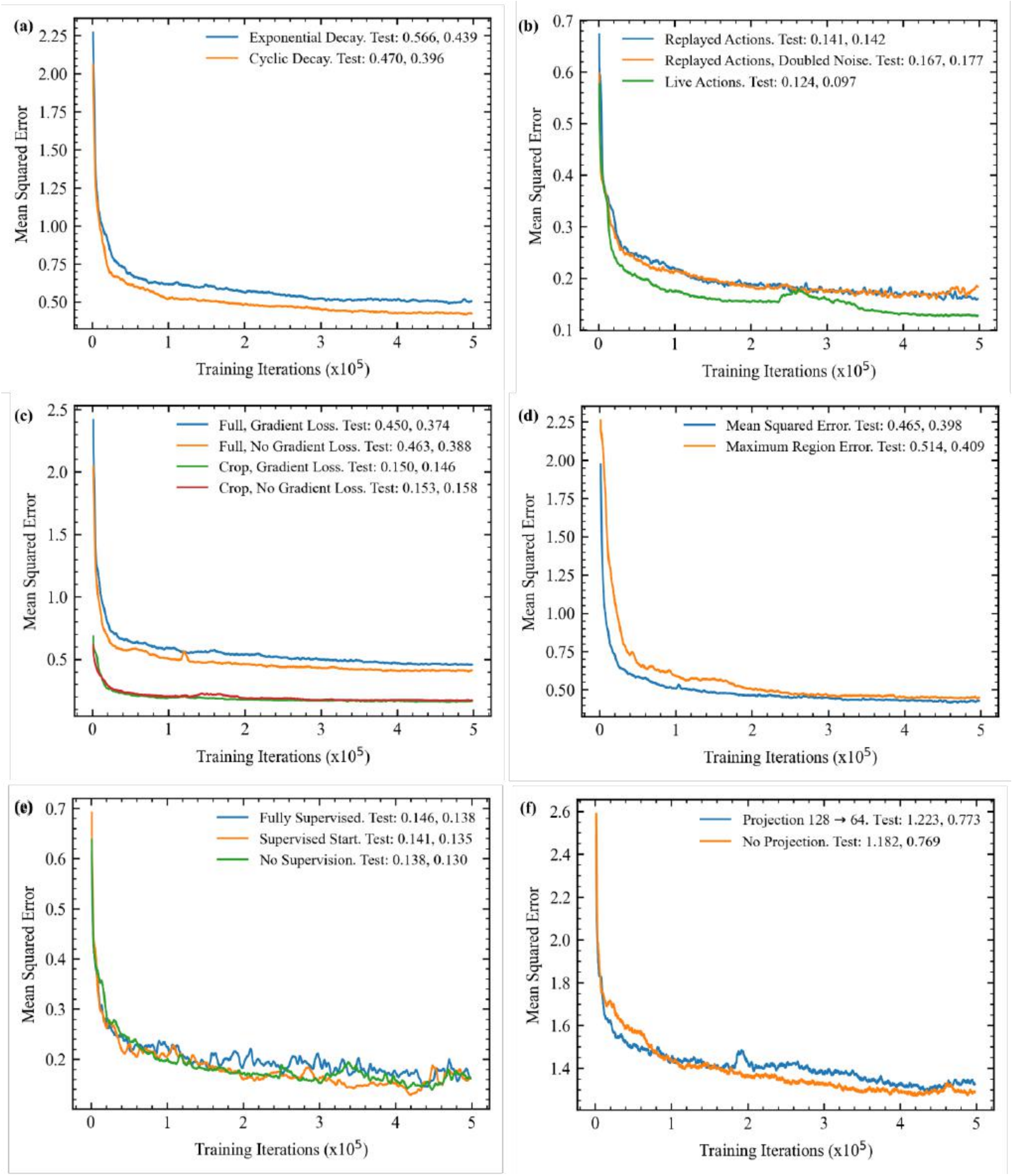


Figure S2. Learning curves for a) exponentially decayed and exponentially decayed cyclic learning rate schedules, b) actor training with differentiation w.r.t. live or replayed actions, c) images downsampled or cropped from full images to 96×96 with and without additional Sobel losses, d) mean squared error and maximum regional mean squared error loss functions, e) supervision throughout training, supervision only at the start, and no supervision, and f) projection from 128 to 64 hidden units or no projection. All learning curves are 2500 iteration boxcar averaged, and results in different plots are not directly comparable due to varying experiment settings. Means and standard deviations of test set errors, "Test: Mean, Std Dev", are at the ends of graph labels.

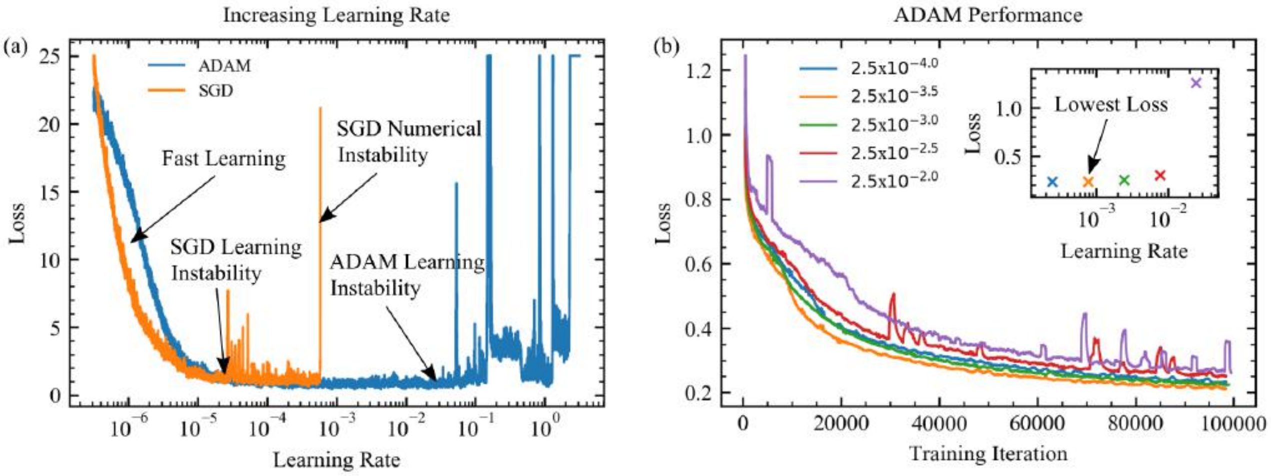


Figure S3. Learning rate optimization. a) Learning rates are increased from $10^{-6.5}$ to $10^{0.5}$ for ADAM and SGD optimization. At the start, convergence is fast for both optimizers. Learning with SGD becomes unstable at learning rates around 2.2×10^{-5}, and numerically unstable near 5.8×10^{-4}, whereas ADAM becomes unstable around 2.5×10^{-2}. b) Training with ADAM optimization for learning rates listed in the legend. Learning is visibly unstable at learning rates of $2.5\times10^{-2.5}$ and 2.5×10^{-2}, and the lowest inset validation loss is for a learning rate of $2.5\times10^{-3.5}$. Learning curves in (b) are 1000 iteration boxcar averaged. Means and standard deviations of test set errors, "Test: Mean, Std Dev", are at the ends of graph labels.

region losses; however, both loss functions result in similar final MSEs. We expect that MSEs calculated with every output pixel result in faster convergence than maximum region errors as more pixels inform gradient calculations. In any case, we expect that a better approach to minimize maximum errors is to use a higher order loss function, such as mean quartic errors. If training with a higher-order loss function is unstable, it might be stabilized by adaptive learning rate clipping[26].

Target losses can be directly computed with Bellman's equation, rather than with target networks. We refer to such directly computed target losses as "supervised" losses,

$$Q_t^{\mathrm{super}} = \sum_{t'=t}^{T} \gamma^{t'-t} L_{t'} \,, \tag{S3}$$

where where $\gamma \in [0,1)$ discounts future step losses, L_t. Learning curves for full supervision, supervision linearly decayed to zero in the first 10^5 iterations, and no supervision are shown in figure S2(e). Overall, final errors are similar for training with and without supervision. However, we find that learning is usually more stable without supervised losses. As a result, we do not recommend using supervised losses.

To accelerate convergence and decrease computation, an LSTM with n_h hidden units can be augmented by a linear projection layer with $n_p < 3n_h/4$ units[27]. Learning curves in figure S2(f) are for $n_h = 128$ and compare training with a projection to $n_p = 64$ units and no projection. Adding a projecting layer increases the initial rate of convergence; however, it also increases final losses. Further, we found that training becomes increasingly prone to instability as n_p is decreased. As a result, we do not use projection layers in our actor or critic networks.

Generator learning rate optimization is shown in figure S3. To find the best initial learning rate for ADAM optimization, we increased the learning rate until training became unstable, as shown in figure S3(a). We performed the learning rate sweep over 10^4 iterations to avoid results being complicated by losses rapidly decreasing in the first couple of thousand. The best learning rate was then selected by training for 10^5 iterations with learning rates within a factor of 10 from a learning rate 10× lower than where training became unstable, as shown in figure S3(b). We performed initial learning rate sweeps in figure S3(a) for both ADAM and stochastic gradient descent[28] (SGD) optimization. We chose ADAM as it is less sensitive to hyperparameter choices than SGD and because ADAM is recommended in the RDPG paper[29].

S4 Test Set Errors

Test set errors are computed for 3954 test set images. Most test set errors are similar to or slightly higher than training set errors. However, training with fixed paths, which is shown in figure 3(a) of the main article, results in high divergence of test and training set errors. We attribute this divergence to the generator overfitting to complete large regions that are not covered by fixed scan paths. In comparison, our learning policy was optimized for training with a variety of adaptive scan paths where overfitting is minimal. After all 10^6 training iterations, means and standard deviations (mean, std dev) of test set errors for fixed paths 2, 3 and 4 are (0.170, 0.182), (0.135, 0.133), and (0.171, 0.184). Instead, we report lower test set errors of (0.106, 0.090), (0.073, 0.045), and (0.106. 0.090), respectively, at 5×10^5 training iterations, which correspond to early stopping[30,31]. All other test set errors were computed after final training iterations.

S5 Distortion Correction

A limitation of partial STEM is that images are usually distorted by probing position errors, which vary with scan path shape[32]. Distortions in raster scans can be corrected by comparing series of images[33,34]. However, distortion correction of adaptive scans is complicated by more complicated scan path shapes and microscope-specific actor command execution characteristics. We expect that command execution characteristics are almost static. Thus, it follows that there is a bijective mapping between probing locations in distorted adaptive partial scans and raster scans. Subsequently, we propose that distortions could be corrected by a cyclic generative adversarial network[35] (GAN). To be clear, this section outlines a possible starting point for future research that can be refined or improved upon. The method's main limitation is that the cyclic GAN would need to be trained or fine-tuned for individual scan systems.

Let I_partial and I_raster be unpaired partial scans and raster scans, respectively. A binary mask, M, can be constructed to be 1 at nominal probing positions in I_partial and 0 elsewhere. We introduce generators $G_{p\rightarrow r}(I_\text{partial})$ and $G_{r\rightarrow p}(I_\text{raster}, M)$ to map from partial scans to raster scans and from raster scans to partial scans, respectively. A mask must be input to the partial scan generator for it to output a partial scan with a realistic distortion field as distortions depend on scan path shape[32]. Finally, we introduce discriminators D_partial and D_raster are trained to distinguish between real and generated partial scans and raster scans, respectively, and predict losses that can be used to train generators to create realistic images. In short, partial scans could be mapped to raster scans by minimizing

$$L^\text{GAN}_{p\rightarrow r} = D_\text{raster}(G_{p\rightarrow r}(I_\text{partial}))\,, \tag{S4}$$

$$L^\text{GAN}_{r\rightarrow p} = D_\text{partial}(MG_{r\rightarrow p}(I_\text{raster}, M))\,, \tag{S5}$$

$$L^\text{cycle}_{r\rightarrow p} = \text{MSE}(MG_{r\rightarrow p}(G_{p\rightarrow r}(I_\text{partial}), M), I_\text{partial})\,, \tag{S6}$$

$$L^\text{cycle}_{p\rightarrow r} = \text{MSE}(G_{p\rightarrow r}(MG_{r\rightarrow p}(I_\text{raster}, M)), I_\text{raster})\,, \tag{S7}$$

$$L_{p\rightarrow r} = L^\text{GAN}_{p\rightarrow r} + bL^\text{cycle}_{r\rightarrow p}\,, \tag{S8}$$

$$L_{r\rightarrow p} = L^\text{GAN}_{r\rightarrow p} + bL^\text{cycle}_{p\rightarrow r}\,, \tag{S9}$$

where $L_{p\rightarrow r}$ and $L_{p\rightarrow r}$ are total losses to optimize $G_{p\rightarrow r}$ and $G_{p\rightarrow r}$, respectively. A scalar, b, balances adversarial and cycle-consistency losses.

S6 Additional Examples

Additional sheets of test set adaptive scans are shown in figure S4 and figure S5. In addition, a sheet of test set spiral scans is shown in figure S6. Target outputs were low-pass filtered by a 5×5 symmetric Gaussian kernel with a 2.5 px standard deviation to suppress high-frequency noise.

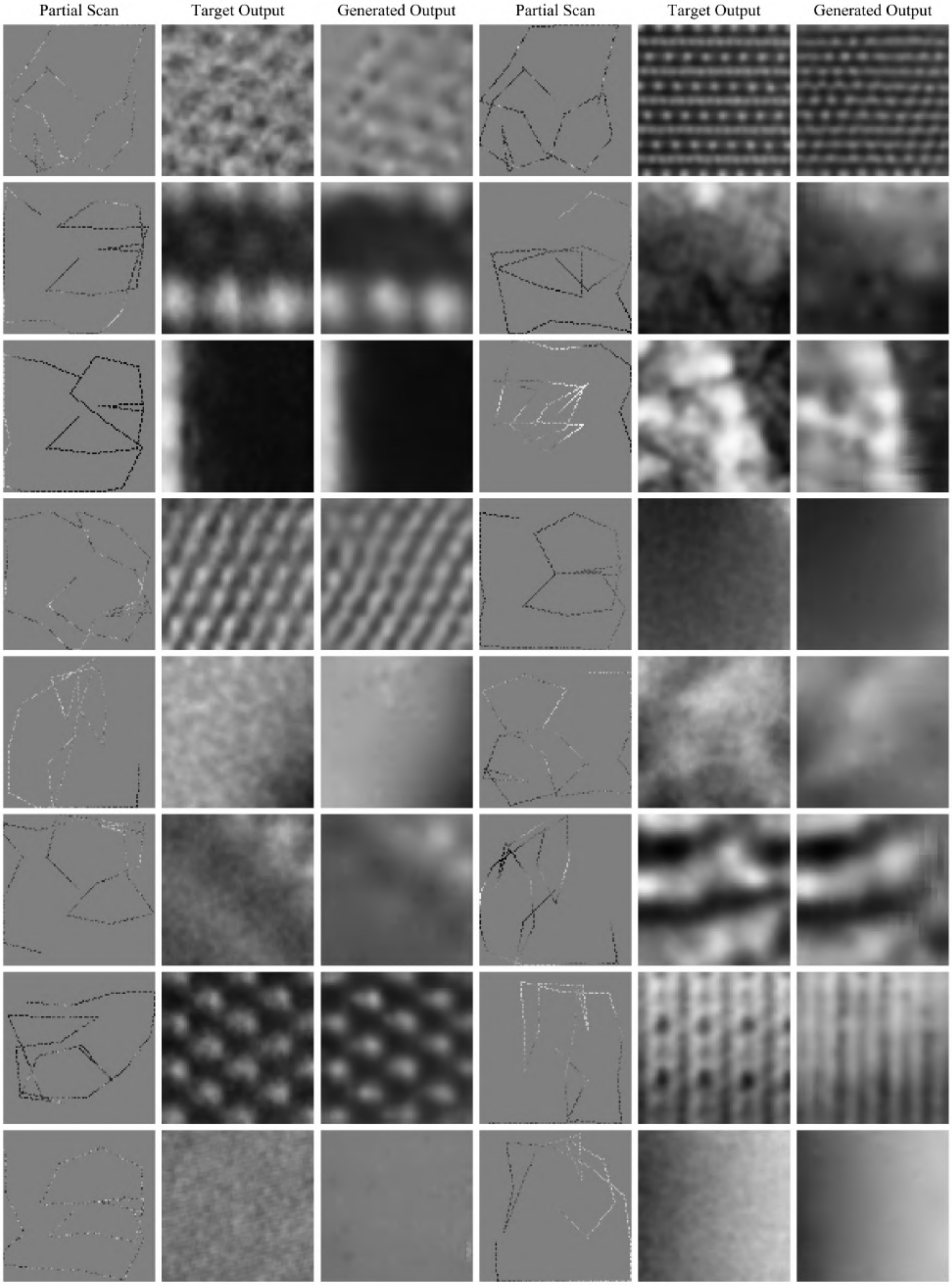


Figure S4. Test set 1/23.04 px coverage adaptive partial scans, target outputs, and generated partial scan completions for 96×96 crops from STEM images.

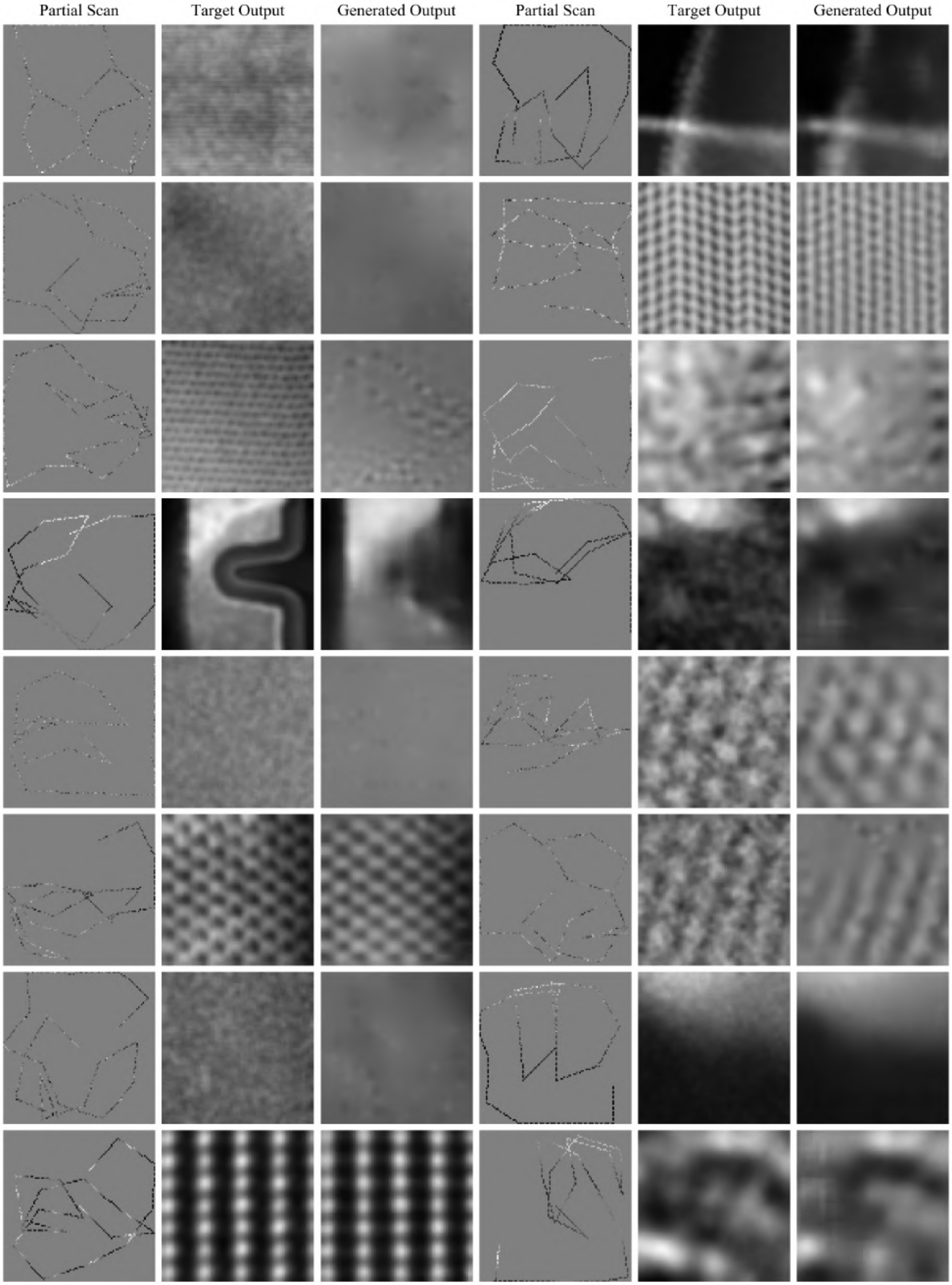


Figure S5. Test set 1/23.04 px coverage adaptive partial scans, target outputs, and generated partial scan completions for 96×96 crops from STEM images.

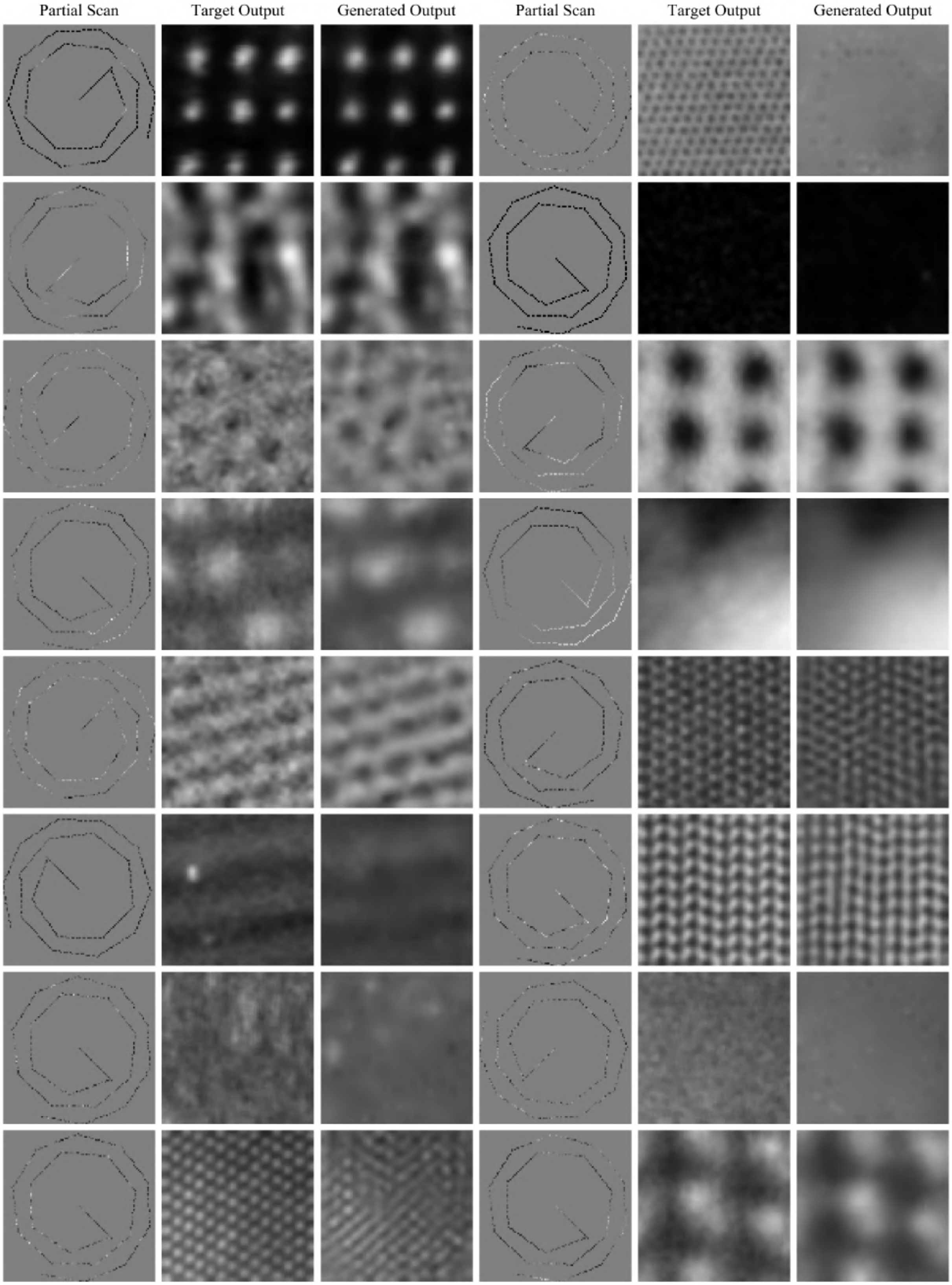


Figure S6. Test set 1/23.04 px coverage spiral partial scans, target outputs, and generated partial scan completions for 96×96 crops from STEM images.

References

1. Zaremba, W., Sutskever, I. & Vinyals, O. Recurrent Neural Network Regularization. *arXiv preprint arXiv:1409.2329* (2014).
2. Graves, A. *et al.* Hybrid Computing Using a Neural Network with Dynamic External Memory. *Nature* **538**, 471–476 (2016).
3. Jozefowicz, R., Zaremba, W. & Sutskever, I. An Empirical Exploration of Recurrent Network Architectures. In *International Conference on Machine Learning*, 2342–2350 (2015).
4. Pitis, S. Non-Zero Initial States for Recurrent Neural Networks. Online: https://r2rt.com/non-zero-initial-states-for-recurrent-neural-networks.html (2016).
5. DeepMind. Differentiable Neural Computer. Online: https://github.com/deepmind/dnc (2018).
6. Dumoulin, V. & Visin, F. A Guide to Convolution Arithmetic for Deep Learning. *arXiv preprint arXiv:1603.07285* (2016).
7. He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. CoRR abs/1512.03385 (2015).
8. Nair, V. & Hinton, G. E. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 807–814 (2010).
9. Ioffe, S. & Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv preprint arXiv:1502.03167* (2015).
10. Glorot, X. & Bengio, Y. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256 (2010).
11. Kukačka, J., Golkov, V. & Cremers, D. Regularization for Deep Learning: A Taxonomy. *arXiv preprint arXiv:1710.10686* (2017).
12. Zhang, J., He, T., Sra, S. & Jadbabaie, A. Why Gradient Clipping Accelerates Training: A Theoretical Justification for Adaptivity. *arXiv preprint arXiv:1905.11881* (2019).
13. Gorbunov, E., Danilova, M. & Gasnikov, A. Stochastic Optimization with Heavy-Tailed Noise via Accelerated Gradient Clipping. *arXiv preprint arXiv:2005.10785* (2020).
14. Chen, X., Wu, Z. S. & Hong, M. Understanding Gradient Clipping in Private SGD: A Geometric Perspective. *arXiv preprint arXiv:2006.15429* (2020).
15. Menon, A. K., Rawat, A. S., Reddi, S. J. & Kumar, S. Can Gradient Clipping Mitigate Label Noise? In *International Conference on Learning Representations* (2019).
16. Smith, L. N. Cyclical Learning Rates for Training Neural Networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 464–472 (IEEE, 2017).
17. Ede, J. M. Warwick Electron Microscopy Datasets. *Mach. Learn. Sci. Technol.* **1**, 045003 (2020).
18. Vairalkar, M. K. & Nimbhorkar, S. Edge Detection of Images Using Sobel Operator. *Int. J. Emerg. Technol. Adv. Eng.* **2**, 291–293 (2012).
19. Zhao, H., Gallo, O., Frosio, I. & Kautz, J. Loss Functions for Neural Networks for Image Processing. *arXiv preprint arXiv:1511.08861* (2015).
20. Gui, J., Sun, Z., Wen, Y., Tao, D. & Ye, J. A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications. *arXiv preprint arXiv:2001.06937* (2020).
21. Saxena, D. & Cao, J. Generative Adversarial Networks (GANs): Challenges, Solutions, and Future Directions. *arXiv preprint arXiv:2005.00065* (2020).
22. Pan, Z. *et al.* Recent Progress on Generative Adversarial Networks (GANs): A Survey. *IEEE Access* **7**, 36322–36333 (2019).
23. Wang, Z., She, Q. & Ward, T. E. Generative Adversarial Networks: A Survey and Taxonomy. *arXiv preprint arXiv:1906.01529* (2019).
24. Ede, J. M. & Beanland, R. Partial Scanning Transmission Electron Microscopy with Deep Learning. *arXiv preprint arXiv:1910.10467* (2020).
25. Grund Pihlgren, G., Sandin, F. & Liwicki, M. Improving Image Autoencoder Embeddings with Perceptual Loss. In *International Joint Conference on Neural Networks* (2020).

26. Ede, J. M. & Beanland, R. Adaptive Learning Rate Clipping Stabilizes Learning. *Mach. Learn. Sci. Technol.* **1**, 015011 (2020).

27. Jia, Y., Wu, Z., Xu, Y., Ke, D. & Su, K. Long Short-Term Memory Projection Recurrent Neural Network Architectures for Piano's Continuous Note Recognition. *J. Robotics* **2017** (2017).

28. Ruder, S. An Overview of Gradient Descent Optimization Algorithms. *arXiv preprint arXiv:1609.04747* (2016).

29. Heess, N., Hunt, J. J., Lillicrap, T. P. & Silver, D. Memory-Based Control with Recurrent Neural Networks. *arXiv preprint arXiv:1512.04455* (2015).

30. Li, M., Soltanolkotabi, M. & Oymak, S. Gradient Descent with Early Stopping is Provably Robust to Label Noise for Overparameterized Neural Networks. In *International Conference on Artificial Intelligence and Statistics*, 4313–4324 (2020).

31. Flynn, T., Yu, K. M., Malik, A., D'Imperio, N. & Yoo, S. Bounding the Expected Run-Time of Nonconvex Optimization with Early Stopping. *arXiv preprint arXiv:2002.08856* (2020).

32. Sang, X. *et al.* Dynamic Scan Control in STEM: Spiral Scans. *Adv. Struct. Chem. Imaging* **2**, 6 (2017).

33. Zhang, C., Berkels, B., Wirth, B. & Voyles, P. M. Joint Denoising and Distortion Correction for Atomic Column Detection in Scanning Transmission Electron Microscopy Images. *Microsc. Microanal.* **23**, 164–165 (2017).

34. Jin, P. & Li, X. Correction of Image Drift and Distortion in a Scanning Electron Microscopy. *J. Microsc.* **260**, 268–280 (2015).

35. Zhu, J.-Y., Park, T., Isola, P. & Efros, A. A. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2223–2232 (2017).

5.2 Reflection

This chapter covers my paper titled "Adaptive Partial Scanning Transmission Electron Microscopy with Reinforcement Learning"[5] and associated research outputs[15,22]. It presents an initial investigation into STEM compressed sensing with contiguous scans that are piecewise adapted to specimens. Adaptive scanning is a finite-horizon partially observed Markov decision process[212,213] (POMDP) with continuous actions and sparse rewards: Scan directions are chosen at each step based on previously observed path segments and a sparse reward is given by correctness completed sparse scans. Scan directions are decided by an actor RNN that cooperates with a generator CNN that completes full scans from sparse scans. Generator losses are not differentiable with respect to actor actions, so I introduced a differentiable critic RNN to predict generator losses from actor actions and observations. The actor and critic are trained by reinforcement learning with a new extension of recurrent deterministic policy gradients[214], and the generator is trained by supervised learning.

This preliminary investigation was unsuccessful insofar that my prototype dynamic scan system does not convincingly outperform static scan systems. However, I believe that it is important to report my progress, despite publication bias against negative results[215–221], as it establishes starting points for further investigation. The main limitation of my scan system is that generator performance is much lower when it is trained for a variety of adaptive scan paths than when it is trained for a single static scan path. For an actor to learn an optimal policy, the generator should ideally be trained until convergence to the highest possible performance for every scan path. However, my generator architecture and learning policy was limited by available computational resources and development time. I also suspect that performance might be improved by replacing RNNs with transformers[222,223] as transformers often achieve similar or higher performance than RNNs[224,225].

There are a variety of additional refinements that could improve training. As an example, RNN computation is delayed by calling a Python function to observe each path segment. Delay could be reduced by more efficient sampling e.g. by using a parallelized routine coded in C/C++; by selecting several possible path segments in advance and selecting the segment that most closely corresponds to an action; or by choosing actions at least one step in advance rather than at each step. In addition, it may help if the generator undergoes additional training iterations in parallel to actor and critic training as improving the generator is critical to improving performance. Finally, increasing generator training iterations may result in overfitting, so it may help to train generators as part of a GAN or introduce other regularization mechanisms. For context, I find that adversarial training can reduce validation divergence[7] (ch. 7) and produce more realistic partial scan completions[4] (ch. 4).

Chapter 6

Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder

6.1 Scientific Paper

This chapter covers the following paper[6].

> J. M. Ede and R. Beanland. Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder. *Ultramicroscopy*, 202:18–25, 2019

Ultramicroscopy 202 (2019) 18–25

Contents lists available at ScienceDirect

Ultramicroscopy

journal homepage: www.elsevier.com/locate/ultramic

Improving electron micrograph signal-to-noise with an atrous convolutional encoder-decoder

Jeffrey M. Ede*, Richard Beanland

Department of Physics, University of Warwick, Coventry, England CV4 7AL, United Kingdom

ARTICLE INFO

Keywords:
Deep learning
Denoising
Electron microscopy
Low dose

ABSTRACT

We present an atrous convolutional encoder-decoder trained to denoise electron micrographs. It consists of a modified Xception backbone, atrous convoltional spatial pyramid pooling module and a multi-stage decoder. Our neural network was trained end-to-end using 512×512 micrographs created from a large dataset of high-dose ($>$ 2500 counts per pixel) micrographs with added Poisson noise to emulate low-dose ($\ll$ 300 counts per pixel) data. It was then fine-tuned for high dose data (200–2500 counts per pixel). Its performance is benchmarked against bilateral, Gaussian, median, total variation, wavelet, and Wiener restoration methods with their default parameters. Our network outperforms their best mean squared error and structural similarity index performances by 24.6% and 9.6% for low doses and by 43.7% and 5.5% for high doses. In both cases, our network's mean squared error has the lowest variance. Source code and links to our high-quality dataset and pretrained models are available at https://github.com/Jeffrey-Ede/Electron-Micrograph-Denoiser.

1. Introduction

Many imaging modes in electron microscopy are limited by noise [1]. Increasingly, ever more sophisticated and expensive hardware and software based methods are being developed to increase resolution, including aberration correctors [2,3], advanced cold field emission guns [4,5], holography [6,7] and others [8–10]. However, techniques that produce low signals [9], or are low-dose to reduce beam damage [11] are fundamentally limited by the signal-to-noise ratios in the micrographs they produce.

Many general [12] and electron microscopy-specific [1,13] denoising algorithms have been developed. However, most of these algorithms rely on hand-crafted filters and are rarely, if ever, fully optimized for their target domains [14]. Neural networks are universal approximators [15] that overcome these difficulties [16] through representation learning [17]. As a result, networks are increasingly being applied to noise removal [18–21] and other applications in electron microscopy [22–25].

Image processing by convolutional neural networks (CNNs) takes the form a series of convolutions that are applied to the input image. While a single convolution may appear to be an almost trivial image processing tool, successive convolutions [26] can transform the data into different mappings. For example, a discrete Fourier transformation can be represented by a single-layer neural network with a linear transfer function [27]. The weightings in each convolution are effectively the parameters that link the neurons in each successive layer of the CNN and allow any conceivable image processing to be undertaken by a general CNN architecture, if trained appropriately. Training in this context means the use of some optimisation routine to adjust the weights of the many convolutions (often several thousand parameters) to minimise a loss function that compares the output image with a desired one and a generally applicable CNN requires training on tens of thousands of model images, which is a non-trivial task. The recent success of large neural networks in computer vision may be attributed to the advent of graphical processing unit (GPU) acceleration [28,29], particularly GPU acceleration of large CNNs [30,31] (CNNs) in distributed settings [32,33], allowing this time-consuming training to be completed on acceptable timescales. Application of these techniques to electron microscopy may allow significant improvements in peformance, particularly in areas that are limited by signal-to-noise.

At the time of writing, there are no large CNNs for electron micrograph denoising. Instead, most denoising networks act on small overlapping crops e.g. [20]. This makes them computationally inefficient and unable to utilize all the information available. Some large denoising networks have been trained as part of generative adversarial networks [34] and try to generate images resembling high-quality training data as closely as possible. This can avoid the blurring effect of most filters by generating features that might be in high-quality

---

* Corresponding author.
E-mail addresses: j.m.ede@warwick.ac.uk (J.M. Ede), r.beanland@warwick.ac.uk (R. Beanland).

https://doi.org/10.1016/j.ultramic.2019.03.017
Received 5 November 2018; Received in revised form 19 March 2019; Accepted 25 March 2019
Available online 26 March 2019
0304-3991/ © 2019 Elsevier B.V. All rights reserved.

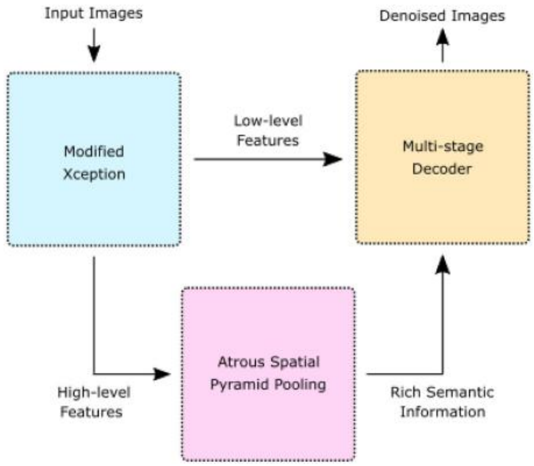


Fig. 1. Simplified network showing how features produced by an Xception backbone are processed. Complex high-level features flow into an atrous spatial pyramid pooling module that produces rich semantic information. This is combined with simple low-level features in a multi-stage decoder to resolve denoised micrographs.

micrographs. However, this means that they are prone to producing undesirable artefacts.

This paper presents the deep CNN in Fig. 1 for electron micrograph denoising. Our network architecture and training hyperparameters are similar to DeepLab3 [35] and DeepLab3+ [36], with the modifications discussed in [37]. Briefly, image processing starts in a modified Xception [38] encoder, which spatially downsamples its 512 × 512 input to a 32 × 32 × 728 tensor. These high-level features flow into an atrous spatial pyramid pooling (ASPP) module [35,36] that combines the outputs of atrous convolutions acting on different spatial scales into a 32 × 32 × 256 tensor. A multi-stage decoder then upsamples the rich ASPP semantics to a 512 × 512 output by combining them with low-level encoder features. This recombination with low-level features helps to reduce signal attenuaiton. For computational and parameter efficiency, most convolutions are depthwise separated into pointwise and depthwise convolutions [38]; rather than standard convolutions.

2. Training

An ideal training dataset might have a wide variety of images and zero noise, enabling the CNN to be trained by inputting artificially degraded images and comparing its output with the zero-noise image. Such datasets can only be produced by simulation (which may be a time-consuming task), or approximated by experimental data. Here, we used 17,267 electron micrographs saved to University of Warwick data servers by scores of scientists working on hundreds of projects over several years. The data set therefore has a diverse constitution, including for example phase contrast images of polymers, diffraction contrast images of semiconductors, high resolution lattice imaging of crystals and a small number of CBED patterns. it is comprised of 32-bit image collected on Gatan SC600 or SC1000 Orius cameras on JEOL 2000FX, 2100, 2100plus and ARM200F microscopes. Scanning TEM (STEM) images were not included. There are several contributions to noise from these charge-coupled device (CCD) cameras, which form an image of an optically coupled scintillator, including [39]: Poisson noise, dictated by the size of the detected signal; electrical readout and shot noise; systematic errors in dark reference, linearity, gain reference, dead pixels or dead columns of pixels (some, but not all, of which is typically corrected by averaging in the camera software); and X-ray noise, which results in individual pixels having extreme high or low values.

In order to minimize the effects of Poisson noise in this dataset we only included micrographs with mean counts per pixel above 2500. X-ray noise, typically affecting only 0.05–0.10% of pixels, was left uncorrected. Each micrograph was cropped to 2048 × 2048 and binned by a factor of two to 1024 × 1024. This increased the mean count per pixel to above 10,000, i.e. a signal-to-(Poisson)noise ratio above 100:1. The effects of systematic errors were mitigated by taking 512 × 512 crops at random positions followed by a random combination of flips and 90° rotations (in the process, augmenting the dataset by a factor of eight). Finally, each image was then scaled to have single-precision (32-bit) pixel values between zero and one.

Our dataset was split into 11,350 training, 2431 validation and 3486 test micrographs. This was pipelined used the TensorFlow [33] deep learning framework to a replica network on each of a pair of Nvidia GTX 1080 Ti GPUs for training via ADAM [40] optimized synchronous stochastic gradient descent [32].

To train the network for low doses, Poisson noise was applied to each 512 × 512 training image after multiplying it by a scale factor, effectively setting the dose in electrons per pixel for a camera with perfect detective quantum efficiency (DQE). These doses were generated by adding 25.0 to numbers, x, sampled from an exponential distribution with probability density function

$$f\left(x, \frac{1}{\beta}\right) = \frac{1}{\beta}\exp\left(-\frac{x}{\beta}\right),\ (x, \beta) \in \mathbb{R}^2_{>0}, \tag{1}$$

where we chose $\beta = 75.0$. To place this in context, the minimum dose in this training data is equivalent to only 25 e^{-2} for a camera with perfect DQE and pixel size size of 5 µm at 50,000 × . These numbers and distribution thus exposed the network to a continuous range of signal-to-noise ratios (most below 10:1) appropriate for typical low-dose electron microscopy [41]. After noise application, ground truth training images were scaled to have the same mean as their noisy counterparts.

After being trained for low-dose applications, the network was fine-tuned for high doses by training it on crops scaled by numbers uniformly distributed between 200 and 2500. That is, by scale factors for signal-to-noise ratios between $10\sqrt{2}$:1 and 50:1.

The learning curve for our network is shown in Fig. 2. It was trained to minimize the mean squared error (MSE) between its denoised output and the original image before the addition of noise. To surpass our low-dose performance benchmarks, our network had to achieve a MSE lower than 7.5×10^{-4}, as tabulated in Table 1. Consequently MSEs were scaled by 1000, limiting trainable parameter perturbations by MSEs

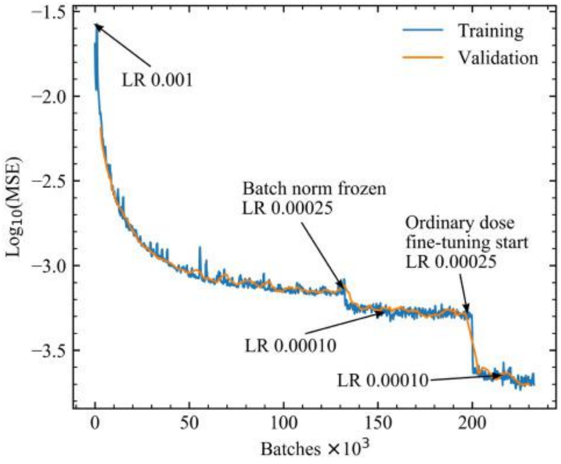


Fig. 2. Mean squared error (MSE) losses of our neural network during training on low dose ($\ll$300 counts ppx) and fine-tuning for high doses (200–2500 counts ppx). Learning rates (LRs) and the freezing of batch normalization are annotated. Validation losses were calculated using one validation example after every five training batches.

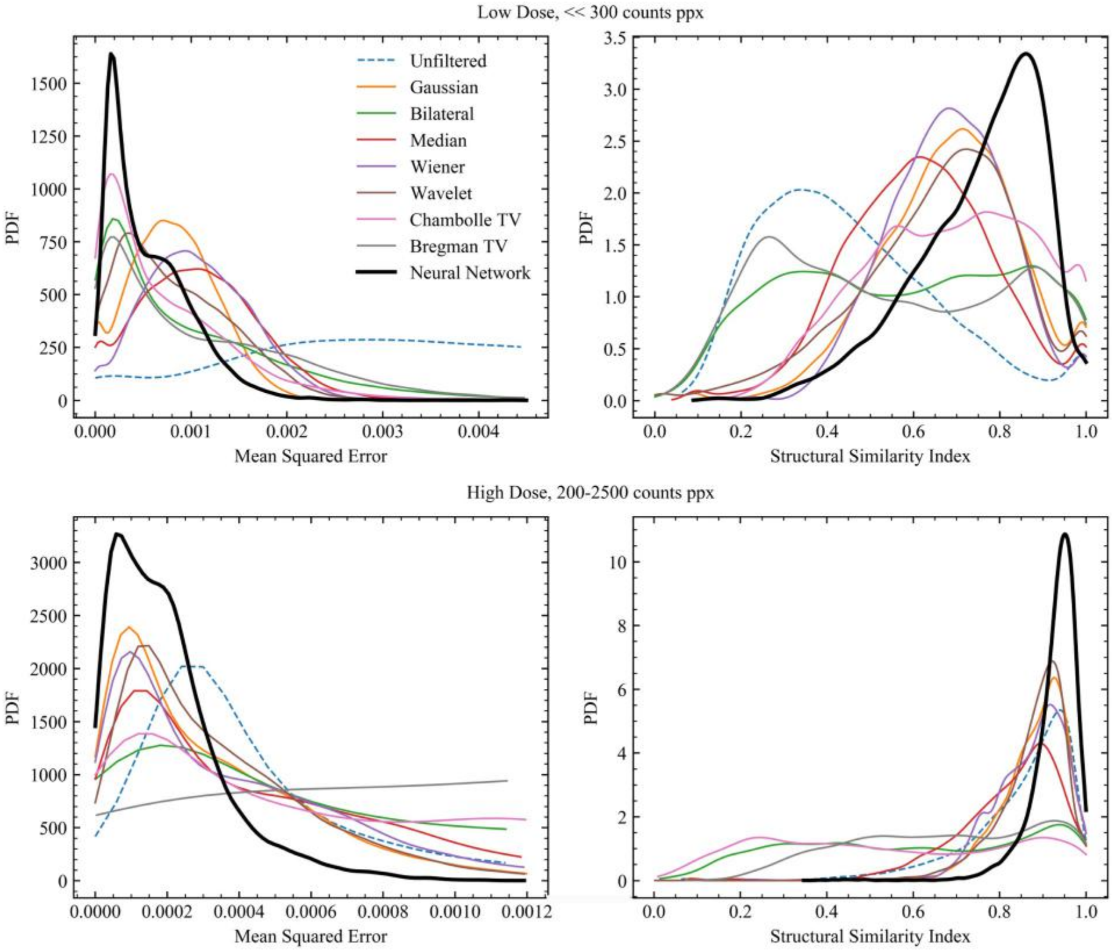


Fig. 3. Gaussian kernel density estimated (KDE) [48,49] MSE and SSIM probability density functions (PDFs) for the denoising methods in Table 1. Only the starts of MSE PDFs are shown. MSE and SSIM performances were divided into 200 equispaced bins in [0.0, 1.2] × 10^{-3} and [0.0, 1.0], respectively, for both low and high doses. KDE bandwidths were found using Scott's Rule [50].

larger than 1.0×10^{-3}. More subtly, this also increased our network's effective learning rate by a factor of 1000.

Our MSE loss was Huberized [42] (i.e. extreme values were replaced with their square root) to prevent the network from being too disturbed by batches with especially noisy training images, i.e.

$$L = \begin{cases} 1000\,MSE, & 1000\,MSE < 1.0 \\ (1000\,MSE)^{\frac{1}{2}}, & 1000\,MSE \geq 1.0 \end{cases} \tag{2}$$

Table 1

Mean MSE and SSIM for several denoising methods applied to 20,000 instances of Poisson noise and their standard errors. All methods were implemented with default parameters. Gaussian: 3 × 3 kernel with a 0.8 px standard deviation. Bilateral: 9 × 9 kernel with radiometric and spatial scales of 75 (scales below 10 have little effect while scales above 150 cartoonize images). Median: 3 × 3 kernel. Wiener: no parameters. Wavelet: BayesShrink adaptive wavelet soft-thresholding with wavelet detail coefficient thresholds estimated using [56]. Chambolle and Bregman TV: iterative total-variation (TV) based denoising [57–59], both with denoising weights of 0.1 and applied until the fractional change in their cost function fell below 2.0×10^{-4} or they reached 200 iterations. Times are for 1000 examples on a 3.4 GHz i7-6700 processor and 1 GTX 1080 Ti GPU, except for our neural network time, which is for 20,000 examples.

	Low Dose, ≪ 300 counts per pixel		High Dose, 200–2500 counts per pixel		
Method	MSE ($\times 10^{-3}$)	SSIM	MSE ($\times 10^{-3}$)	SSIM	Time (ms)
Unfiltered	4.357 ± 2.558	0.454 ± 0.208	0.508 ± 0.682	0.850 ± 0.123	0.0
Gaussian [51]	0.816 ± 0.452	0.685 ± 0.159	0.344 ± 0.334	0.878 ± 0.087	1.0
Bilateral [51,52]	1.025 ± 1.152	0.574 ± 0.261	1.243 ± 1.392	0.600 ± 0.271	5.7
Median [51]	1.083 ± 0.618	0.618 ± 0.171	0.507 ± 0.512	0.821 ± 0.126	1.2
Wiener [53]	1.068 ± 0.546	0.681 ± **0.137**	0.402 ± 0.389	0.870 ± 0.085	133.4
Wavelet [54,55]	0.832 ± 0.580	0.657 ± 0.186	0.357 ± 0.312	0.875 ± 0.085	42.4
Chambolle TV [54]	0.746 ± 0.725	0.680 ± 0.192	0.901 ± 0.909	0.674 ± 0.217	313.6
Bregman TV [54]	1.109 ± 1.031	0.544 ± 0.268	4.074 ± 3.025	0.348 ± 0.312	2061.3
Neural network	**0.562** ± **0.449**	**0.752** ± 0.147	**0.201** ± **0.169**	**0.926** ± **0.057**	77.0

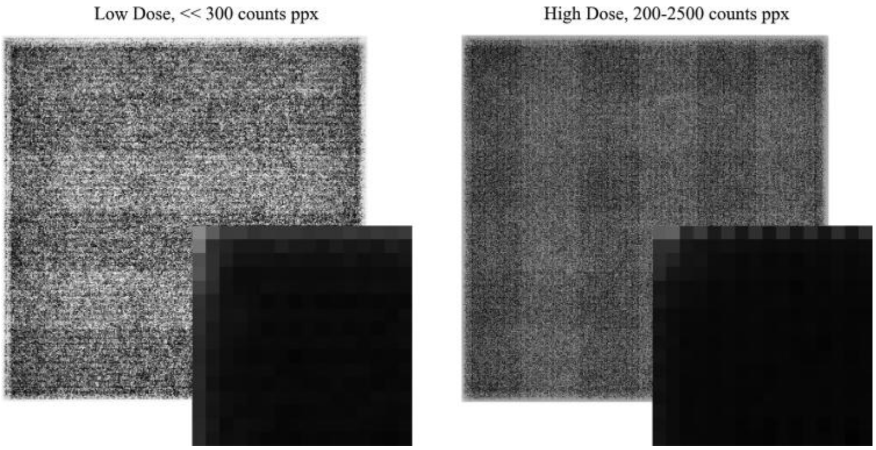


Fig. 4. Mean absolute errors of our low and high dose networks' 512 × 512 outputs for 20,000 instances of Poisson noise. Contrast limited adaptive histogram equalization [61] has been used to massively increase contrast, revealing grid-like error variation. Subplots show the top-left 16 × 16 pixels' mean absolute errors unadjusted. Variations are small and errors are close to the minimum everywhere, except at the edges where they are higher. Low dose errors are in [0.0169, 0.0320]; high dose errors are in [0.0098, 0.0272].

All neurons were ReLU6 [43] activated. Our experiments with other activations are discussed in [37]. Weights were Xavier uniform initialized [44] and biases were zero initialized. During training, L2 regularization [45] was applied by adding 5×10^{-5} times the quadrature sum of all trainable variables to the loss function. This prevented trainable parameters growing unbounded, decreasing their ability to learn in proportion [46]. Importantly, this ensures that our network continues to learn effectively if it is fine-tuned or given additional training. We did not perform an extensive search for our regularization rate and think that 5×10^{-5} may be too high.

Our network is allowed to produce outputs outside the range of the input image, i.e. [0.0,1.0]. However, outputs can be optionally clipped to this range during inference. Noisy images are expected to have more extreme values than restored images so clipping the restored images to [0.0,1.0] helps to safeguard against overly extreme outputs. Consequently, all performance statistics; including losses during training, are reported for clipped outputs.

We trained batch normalization layers from [47] with a decay rate of 0.999 until the instabilities introduced by their trainable parameters began to limit convergence. Then, after 134,108 batches, batch normalization was frozen. During training, batch normalization layers map features, y, using their means, μ and standard deviations, σ, and a small number, ε, to the normalized frames

$$y' := \frac{y - \mu}{\sqrt{\sigma^2 + \varepsilon}}. \tag{3}$$

Batch normalization has a number of advantages, including reducing covariate shift [47] and improving gradient stability [60] to decrease training time and improve accuracy. We found that batch normalization also seems to significantly reduced structured error variation in our output images (see Section 3).

ADAM [40] optimization was used throughout training with a stepped learning rate. For the low dose version of the network, we used a learning rate of 1.0×10^{-3} for 134,108 batches, 2.5×10^{-4} for another 17,713 batches and then 1.0×10^{-4} for 46,690 batches. The network was then fine-tuned for high doses using a learning rate of 2.5×10^{-4} for 16,773 batches, then 1.0×10^{-4} for 17,562 batches. These unusual intervals are a result of learning rates being adjusted at wall clock times.

We found the recommended [33,40] ADAM decay rate for the first moment of the momentum, $\beta_1 = 0.9$, to be too high and chose $\beta_1 = 0.5$ instead. This lower β_1 made training more responsive to varying noise levels in batches.

We designed our network to be trained end-to-end; rather than in stages, so that it is easy to fine-tune or retrain for other applications. This is important as multi-stage training regiments introduce additional hyperparameters and complexity that may make the network difficult to use in practice. Nevertheless, we expect it to be possible to achieve slightly higher performance by training components of our neural network in stages and then fine-tuning the whole network end-to-end. Multistage training to eek out slightly higher performance may be appropriate if our network is to be tasked upon a specific, performance-critical application.

3. Performance

To benchmark our network's performance, we applied it and eight popular denoising methods to 20,000 instances of noise applied to 512 × 512 test micrographs. Table 1 shows the results for both low-dose and high dose networks and data, giving the mean MSE and structural similarity index (SSIM) [62] for the denoised images compared with the original images before noise was added. The first row gives statistics for the unfiltered data, establishing a baseline. Our network outperforms all other methods using both metrics (N.B. SSIM is 1 for perceptually similar images; 0 for perceptually dissimilar). The improved performance can be seen in more detail in Fig. 3, which shows performance probability density functions (PDFs) for the both low- and high-dose versions of our network. Notably, the fraction of images with a MSE above 0.002 is negligible for our low-dose neural network, while all other methods have a noticeable tail of difficult-to-correct images that retain higher MSEs.

All methods produce much smaller MSEs for the high-dose data; however, a similar trend is present. The network consistently produces better results and has fewer images that have high errors. Interestingly, the mean squared error PDFs for the network appear to have two main modes: there is a sharp peak at 0.0002 and a second at 0.0008 in the MSE PDF plots of Fig. 3. Similarly, a bimodal distribution is present in the high dose data. This may be due to different performance for different types of micrograph, perhaps reflecting the mixture of diffraction contrast and phase contrast images used in training and testing. If this is the case, it may be possible to improve performance significantly for specific applications by training on a narrower range of data.

Mean absolute errors of our network's output for 20,000 examples are shown in Fig. 4. Absolute errors are almost uniformly low. They are only significantly higher near the edges of the output, as shown by the inset image showing 16 × 16 corner pixels. The mean absolute errors per pixel are 0.0177 and 0.0102 for low and high doses, respectively. Small, grid-like variations in absolute error are revealed by contrast-limited adaptive histogram equalization [61] in Fig. 4. These variations are common in deep learning and are often associated with transpositional convolutions. Consequently, some authors [63] have recommended their replacement with bilinear upsampling followed by convolution. We tried this; however, we found that while it made the errors less grid-like, it did not change the absolute errors significantly.

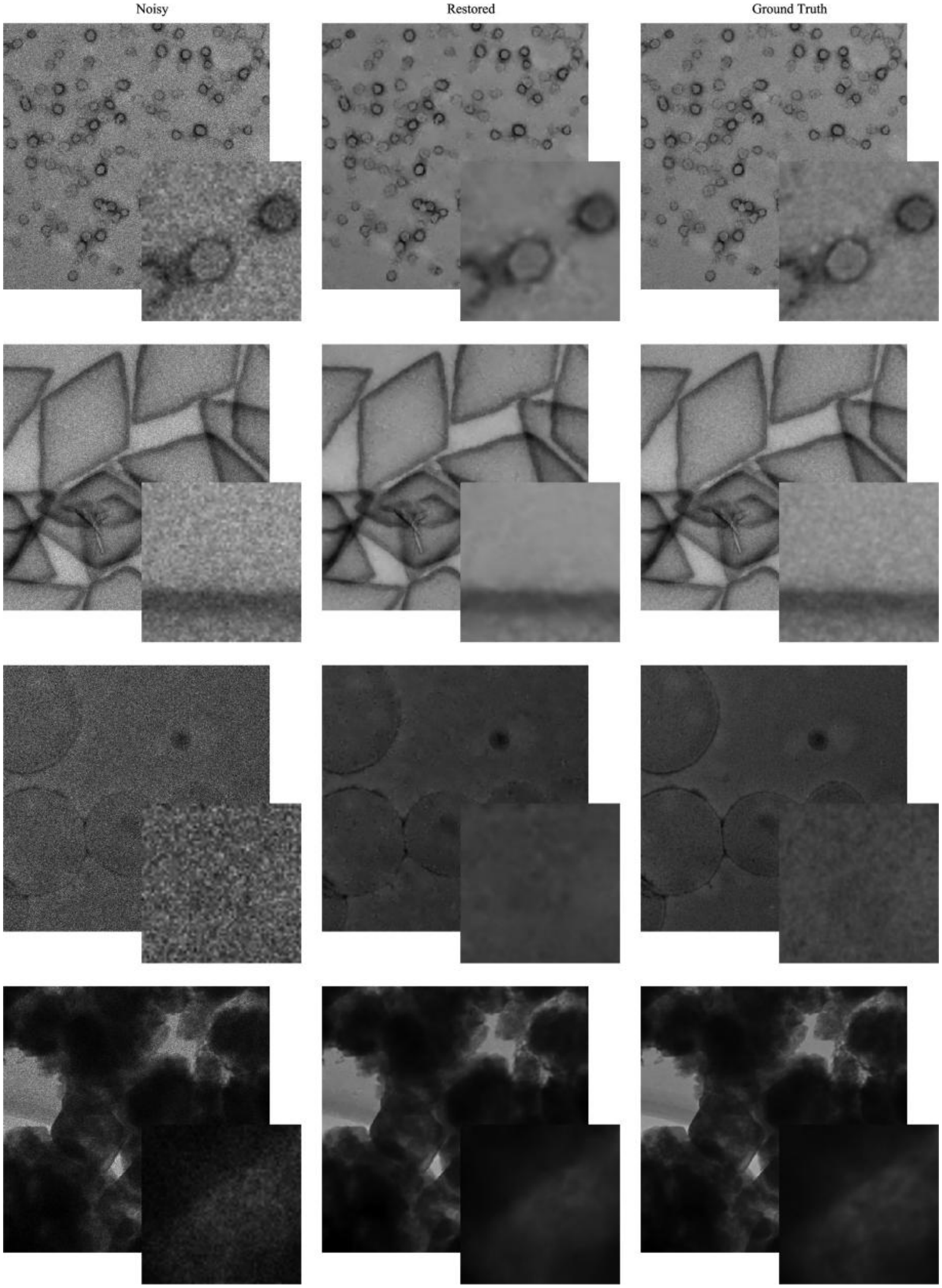


Fig. 5. Example applications of the noise-removal network to instances of Poisson noise applied to 512 × 512 crops from high-quality micrographs. Enlarged 64 × 64 regions from the top left of each crop are shown to ease comparison.

Instead, we found batch normalization to be a simple and effective way to reduce structured error variation, likely due to the regularizing effect of its instability. This is evident from the more grid-like errors in the high dose version of our network, which was trained for longer after batch normalization was frozen. More advanced methods that reduce structured error variation are discussed in [64] but were not applied here.

Example applications of our low-dose network being use to removed applied noise from high-quality 512 × 512 electron micrographs are shown in Fig. 5. In practice, our program may be applied to arbitrarily large images by dividing them into slightly overlapping 512 × 512 crops that can be processed. Our code does this by default. Slightly overlapping crops allows the higher errors at the edges of the neural network output to be avoided, decreasing errors below the values we report. To reduce errors at image edges, where crops cannot be overlapped, we use reflection padding. Users can customize the amount overlap, padding and many other options or use default values.

4. Discussion

The most successful conventional noise-reduction method applied to our data is the iterative Chambolle total variation algorithm, c.f. Fig. 3, which takes more than four times the runtime of our neural network on our hardware. As part of development, we experimented with shallower architectures similar to [18,20,21]; however, these networks could not surpass Chambolle's low-dose benchmark (Table 1). Consequently, we switched to the deeper Xception-based architecture presented here.

Overall, our neural network demonstrates that deep learning is a promising avenue to improve low-dose electron microscopic imaging. While our network significantly outperforms Chambolle TV for our data, it still has the capacity to be improved through better learning protocols or further training for specific datasets. It is most useful in applications limited by noise, particularly biological low-dose applications, and tuning its performance for the noise characteristics of a specific dose, microscope and camera may be worthwhile for optimal performance. Further improvement of the encoder-decoder architecture may also be possible, producing further gains in performance. One of the advantages for network algorithms is their speed in comparison with other techniques. We speed-tested our network by applying it to 20,000 512 × 512 images with one external GTX 1080 Ti GPU and one thread of an i7-6700 processor. Once loaded, it has a mean worst-case (i.e. batch size 1) inference time of 77.0 ms, which means that it can readily be applied to large amounts of data. This compares favorably with the best conventional method on our data; Chambolle's, which has an average runtime of 313.6 ms.

We designed our network to have a high capacity so that it can discriminate between and learn from experiences in multiple domains. It has also been L2 regularized to keep its weights and biases low, ensuring that it will continue to learn effectively. This means that it is well-suited for further training to improve performance in other domains. Empirically, pre-training a model in a domain other than the target domain often improves performance. Consequently, we recommend the pretrained models we provide as a starting point to be fine-tuned for other domains.

5. Other work

Our original write-up of this work; which is less targeted at electron microscopists, is available as [37]. Our original preprint has more example applications to TEM and STEM images, a more detailed discussion of the architecture and additional experiments we did to refine it.

6. Summary

We have developed a deep neural network for electron micrograph denoising using a modified Xception backbone for encoding, an atrous spatial pyramid pooling module and a multi-stage decoder. We find that it outperforms existing methods for low and high electron doses. It is fast and easy to apply to arbitrarily large datasets. While our network generally performs well on most noisy images as-is, further optimization for specific applications is possible. We expect applications to be found in low-dose imaging, which is limited by noise.

Our code and pre-trained low- and high-dose models are available at: https://github.com/Jeffrey-Ede/Electron-Micrograph-Denoiser.

Appendix A. Architecture

A detailed schematic of our neural network architecture is show in Fig. 6. The components in our network are

Avg Pool *w* x *w*, Stride *x*: Average pooling is applied by calculating mean values for squares of width *w* that are spatially separated by *x* elements.

Bilinear Upsamp x *m*: This is an extension of linear interpolation in one dimension to two dimensions. It is used to upsample images by a factor of *m*.

Clip [*a*,*b*] : Clip the inputs tensor values so that they are in a specified range. If values are less than *a*, they are set to *a*; if values are more than *b*, they are set to *b*.

Concat, *d*: Concatenation of two tensors with the same spatial dimensions to a new tensor with the same spatial dimensions and both their feature spaces. The size of the new feature depth, *d*, is the sum of the feature depths of the tensors being concatenated.

Conv *d*,*w* x *w*, Stride, *x*: Convolution with a square kernel of width, *w*, that outputs *d* feature layers. If the stride is specified, convolutions are only applied to every *x*th spatial element of their input, rather than to every element. Striding is not applied depthwise.

Sep Conv *d*,*w* x *w*, Stride, *x*, Rate, *r*: Depthwise separable convolutions consist of depthwise convolutions that acts on each feature layer followed by pointwise convolutions. The separation of the convolution into two parts allows it to be implemented more efficiently on most modern GPUs. The arguments specify a square kernel of width *w* that outputs *d* feature layers. If the stride is specified, convolutions are only applied to every *x*th spatial element of their input, rather than to every element. Strided convolutions are used so that networks can learn their own downsampling and are not applied depthwise. If an atrous rate, *r*, is specified, kernel elements are spatially spread out by an extra $r - 1$ elements, rather than being next to each other.

Trans Conv *d*, *w* x *w*, Stride, *x*: Transpositional convolutions; sometimes called deconvolutions after [65], allow the network to learn its own upsampling. They can be thought of as adding $x - 1$ zeros between spatial elements, then applying a convolution with a square kernel of width *w* that outputs *d* feature maps.

⊕: Circled plus signs indicate residual connections where incoming tensors are added together. These help reduce signal attenuation and allow the network to learn identity mappings more easily.

All convolutions are followed by batch normalization then ReLU6 activation. Extra batch normalization is added between the depthwise and pointwise convolutions of depthwise separable convolutions. Weights were Xavier uniform initialized; biases were zero-initialized.

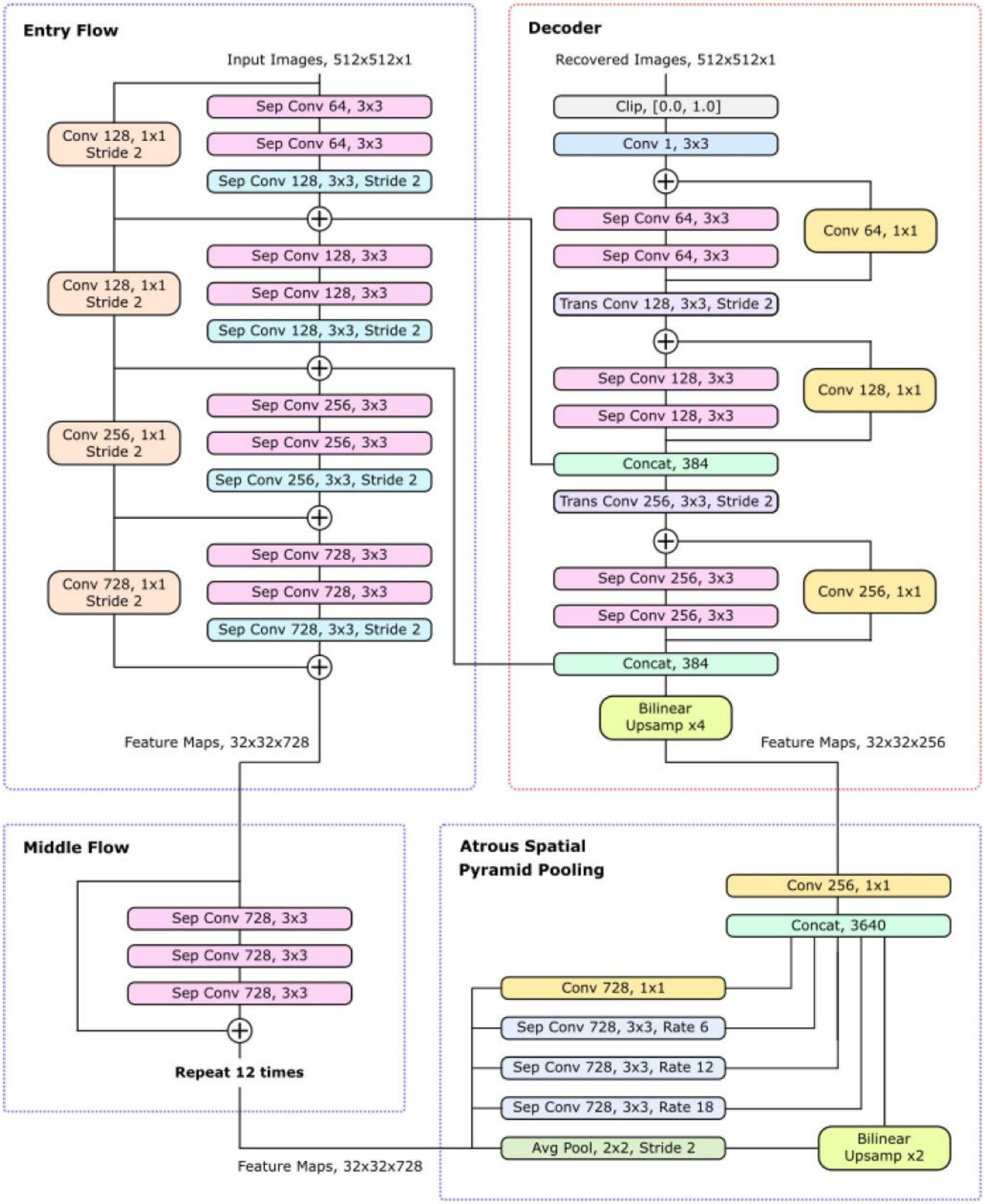


Fig. 6. Architecture of our deep convolutional encoder-decoder for electron micrograph denoising. The entry and middle flows develop high-level features that are sampled at multiple scales by the atrous spatial pyramid pooling module. This produces rich semantic information that is concatenated with low-level entry flow features and resolved into denoised micrographs by the decoder.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ultramic.2019.03.017.

References

[1] E. Oho, N. Ichise, W.H. Martin, K.-R. Peters, Practical method for noise removal in scanning electron microscopy, Scanning 18 (1) (1996) 50–54.
[2] S.J. Pennycook, The impact of stem aberration correction on materials science, Ultramicroscopy 180 (2017) 22–33.
[3] M. Linck, P. Hartel, S. Uhlemann, F. Kahl, H. Müller, J. Zach, M. Haider, M. Niestadt, M. Bischoff, J. Biskupek, et al., Chromatic aberration correction for atomic resolution tem imaging from 20 to 80 kv, Phys. Rev. Lett. 117 (7) (2016) 076101.
[4] F. Houdellier, L. De Knoop, C. Gatel, A. Masseboeuf, S. Mamishin, Y. Taniguchi, M. Delmas, M. Monthioux, M. Hÿtch, E. Snoeck, Development of tem and sem high brightness electron guns using cold-field emission from a carbon nanotip, Ultramicroscopy 151 (2015) 107–115.
[5] T. Akashi, Y. Takahashi, T. Tanigaki, T. Shimakura, T. Kawasaki, T. Furutsu, H. Shinada, H. Müller, M. Haider, N. Osakabe, et al., Aberration corrected 1.2-mv cold field-emission transmission electron microscope with a sub-50-pm resolution, Appl. Phys. Lett. 106 (7) (2015) 074101.
[6] H. Adaniya, M. Cheung, C. Cassidy, M. Yamashita, T. Shintake, Development of a sem-based low-energy in-line electron holography microscope for individual particle imaging, Ultramicroscopy 188 (2018) 31–40.
[7] C.T. Koch, Towards full-resolution inline electron holography, Micron 63 (2014) 69–75.
[8] A. Feist, N. Bach, N.R. da Silva, T. Danz, M. Möller, K.E. Priebe, T. Domröse, J.G. Gatzmann, S. Rost, J. Schauss, et al., Ultrafast transmission electron microscopy using a laser-driven field emitter: femtosecond resolution with a high coherence electron beam, Ultramicroscopy 176 (2017) 63–73.
[9] V. Migunov, H. Ryll, X. Zhuge, M. Simson, L. Strüder, K.J. Batenburg, L. Houben, R.E. Dunin-Borkowski, Rapid low dose electron tomography using a direct electron detection camera, Sci. Rep. 5 (2015) 14516.
[10] Y. Jiang, Z. Chen, Y. Han, P. Deb, H. Gao, S. Xie, P. Purohit, M.W. Tate, J. Park, S.M. Gruner, et al., Electron ptychography of 2d materials to deep sub-ångström resolution, Nature 559 (7714) (2018) 343.
[11] J. Hattne, D. Shi, C. Glynn, C.-T. Zee, M. Gallagher-Jones, M.W. Martynowycz, J.A. Rodriguez, T. Gonen, Analysis of global and site-specific radiation damage in cryo-em, Structure (2018).
[12] M.C. Motwani, M.C. Gadiya, R.C. Motwani, F.C. Harris, Survey of image denoising techniques, Proceedings of GSPX, (2004), pp. 27–30.
[13] Q. Zhang, C.L. Bajaj, Cryo-electron microscopy data denoising based on the generalized digitized total variation method, Far East J. Appl. Math. 45 (2) (2010) 83.
[14] H.S. Kushwaha, S. Tanwar, K. Rathore, S. Srivastava, De-noising filters for tem (transmission electron microscopy) image of nanomaterials, Advanced Computing & Communication Technologies (ACCT), 2012 Second International Conference on, IEEE, 2012, pp. 276–281.
[15] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.
[16] H.W. Lin, M. Tegmark, D. Rolnick, Why does deep and cheap learning work so well? J. Stat. Phys. 168 (6) (2017) 1223–1247.
[17] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828.
[18] X. Yang, V. De Andrade, W. Scullin, E.L. Dyer, N. Kasthuri, F. De Carlo, D. Gürsoy, Low-dose x-ray tomography through a deep convolutional neural network, Sci. Rep. 8 (1) (2018) 2575.
[19] T. Remez, O. Litany, R. Giryes, A.M. Bronstein, Deep convolutional denoising of low-light images, arXiv preprint arXiv:/1701.01687 (2017).
[20] X.-J. Mao, C. Shen, Y.-B. Yang, Image restoration using convolutional auto-encoders with symmetric skip connections, arXiv preprint arXiv:/1606.08921 (2016).
[21] K. Zhang, W. Zuo, Y. Chen, D. Meng, L. Zhang, Beyond a Gaussian denoiser: residual learning of deep cnn for image denoising, IEEE Trans. Image Process. 26 (7) (2017) 3142–3155.
[22] W. Xu, J.M. LeBeau, A deep convolutional neural network to analyze position averaged convergent beam electron diffraction patterns, arXiv preprint arXiv:/1708.00855 (2017).
[23] K. Lee, J. Zung, P. Li, V. Jain, H.S. Seung, Superhuman accuracy on the snemi3d connectomics challenge, arXiv preprint arXiv:/1706.00120 (2017).
[24] D. Ciresan, A. Giusti, L.M. Gambardella, J. Schmidhuber, Deep neural networks segment neuronal membranes in electron microscopy images, Advances in Neural Information Processing Systems, (2012), pp. 2843–2851.
[25] Y. Zhu, Q. Ouyang, Y. Mao, A deep convolutional neural network approach to single-particle recognition in cryo-electron microscopy, BMC Bioinform. 18 (1) (2017) 348.
[26] V. Dumoulin, F. Visin, A guide to convolution arithmetic for deep learning, arXiv preprint arXiv:/1603.07285 (2016).
[27] R. Velik, Discrete fourier transform computation using neural networks, 2008 International Conference on Computational Intelligence and Security, IEEE, 2008, pp. 120–123.
[28] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Netw. 61 (2015) 85–117.
[29] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, E. Shelhamer, Cudnn: efficient primitives for deep learning, arXiv preprint arXiv:/1410.0759 (2014).
[30] M.T. McCann, K.H. Jin, M. Unser, A review of convolutional neural networks for inverse problems in imaging, arXiv preprint arXiv:/1710.04011 (2017).
[31] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, F.E. Alsaadi, A survey of deep neural network architectures and their applications, Neurocomputing 234 (2017) 11–26.
[32] J. Chen, X. Pan, R. Monga, S. Bengio, R. Jozefowicz, Revisiting distributed synchronous sgd, arXiv preprint arXiv:/1604.00981 (2016).
[33] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: a system for large-scale machine learning. OSDI, 16 (2016), pp. 265–283.
[34] Q. Yang, P. Yan, Y. Zhang, H. Yu, Y. Shi, X. Mou, M.K. Kalra, Y. Zhang, L. Sun, G. Wang, Low dose ct image denoising using a generative adversarial network with wasserstein distance and perceptual loss, IEEE Trans. Med. Imaging (2018).
[35] L.-C. Chen, G. Papandreou, F. Schroff, H. Adam, Rethinking atrous convolution for semantic image segmentation, arXiv preprint arXiv:/1706.05587 (2017).
[36] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, Encoder-decoder with atrous separable convolution for semantic image segmentation, arXiv preprint arXiv:/1802.02611 (2018).
[37] J.M. Ede, Improving electron micrograph signal-to-noise with an atrous convolutional encoder-decoder, arXiv preprint arXiv:/1807.11234 (2018).
[38] F. Chollet, Xception: deep learning with depthwise separable convolutions, arXiv preprint (2016).
[39] J.R. Janesick, Scientific charge-coupled devices(2001).
[40] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint arXiv:/1412.6980 (2014).
[41] G. McMullan, A. Faruqi, D. Clare, R. Henderson, Comparison of optimal performance at 300kev of three direct electron detectors for use in low dose electron microscopy, Ultramicroscopy 147 (2014) 156–163, doi:10.1016/j.ultramic.2014.08.002. URL http://www.sciencedirect.com/science/article/pii/S030439911400151X.
[42] P.J. Huber, Robust estimation of a location parameter, Ann. Math. Stat. (1964) 73–101.
[43] A. Krizhevsky, G. Hinton, Convolutional deep belief networks on cifar-10, Technical report, U. Toronto, 2010.
[44] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, (2010), pp. 249–256.
[45] J. Kukačka, V. Golkov, D. Cremers, Regularization for deep learning: a taxonomy, arXiv preprint arXiv:/1710.10686 (2017).
[46] T. Salimans, D.P. Kingma, Weight normalization: A simple reparameterization to accelerate training of deep neural networks, Advances in Neural Information Processing Systems, (2016), pp. 901–909.
[47] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv:/1502.03167 (2015).
[48] B.A. Turlach, Bandwidth selection in kernel density estimation: a review, CORE and Institut de Statistique, Citeseer, 1993.
[49] D.M. Bashtannyk, R.J. Hyndman, Bandwidth selection for kernel conditional density estimation, Comput. Stat. Data Anal. 36 (3) (2001) 279–298.
[50] D.W. Scott, Multivariate Density Estimation: Theory, Practice, and Visualization, John Wiley & Sons, 2015.
[51] G. Bradski, The opencv library, Dr. Dobb's J. Softw. Tools (2000).
[52] C. Tomasi, R. Manduchi, Bilateral filtering for gray and color images, Computer Vision, 1998. Sixth International Conference on, IEEE, 1998, pp. 839–846.
[53] E. Jones, T. Oliphant, P. Peterson, et al., SciPy: Open source scientific tools for Python, 2001, [Online; accessed], URL http://www.scipy.org/.
[54] S. Van der Walt, J.L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J.D. Warner, N. Yager, E. Gouillart, T. Yu, Scikit-image: image processing in python, PeerJ 2 (2014) e453.
[55] S.G. Chang, B. Yu, M. Vetterli, Adaptive wavelet thresholding for image denoising and compression, IEEE Trans. Image Process. 9 (9) (2000) 1532–1546.
[56] D.L. Donoho, J.M. Johnstone, Ideal spatial adaptation by wavelet shrinkage, Biometrika 81 (3) (1994) 425–455.
[57] A. Chambolle, An algorithm for total variation minimization and applications, J. Math. Imaging Vis. 20 (1–2) (2004) 89–97.
[58] T. Goldstein, S. Osher, The split bregman method for l1-regularized problems, SIAM J. Imaging Sci. 2 (2) (2009) 323–343.
[59] P. Getreuer, Rudin-osher-fatemi total variation denoising using split bregman, Image Process. On Line 2 (2012) 74–95.
[60] S. Santurkar, D. Tsipras, A. Ilyas, A. Madry, How does batch normalization help optimization? (no, it is not about internal covariate shift), arXiv preprint arXiv:/1805.11604 (2018).
[61] K. Zuiderveld, Contrast limited adaptive histogram equalization, Graphics Gems (1994) 474–485.
[62] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612.
[63] A. Odena, V. Dumoulin, C. Olah, Deconvolution and checkerboard artifacts, Distill 1 (10) (2016) e3.
[64] Y. Sugawara, S. Shiota, H. Kiya, Super-resolution using convolutional neural networks without any checkerboard artifacts, arXiv preprint arXiv:/1806.02658 (2018).
[65] M.D. Zeiler, D. Krishnan, G.W. Taylor, R. Fergus, Deconvolutional networks (2010).

6.2 Amendments and Corrections

There are amendments or corrections to the paper[6] covered by this chapter.

Location: Page 19, text following eqn 1.
Change: "...to only 25 e^{-2} for a camera..." should say "...to only 25 $e\text{Å}^{-2}$ for a camera...".

Location: Page 21, first paragraph of performance section.
Change: "...structural similarity index (SSIM)..." should say "...structural similarity index measure (SSIM)...".

6.3 Reflection

This chapter covers our paper titled "Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder"[6] and associated research outputs[15,23,24]. Our paper presents a DNN based on Deeplabv3+ that is trained to remove Poisson noise from TEM images. My DNN is affectionately named "Fluffles" and it is the only DNN that I have named. Pretrained models and performance characterizations are provided for DNNs trained for low and high electron doses. We also show that my DNN has lower MSEs, lower MSE variance, higher SSIMs, and lower or similar SSIM variance to other popular algorithms. We also provide MSE and SSIM distributions, and visualize errors for each output pixel.

Due to limited available computational resources, DNN training was stopped after it surpassed the performance of a variety of popular denoising algorithms. However, there are many other denoising algorithms[226–228] that might achieve higher performance, some of which were developed for electron microscopy[1]. For example, we did not compare our DNN against block-matching and 3D filtering[229,230] (BM3D), which often achieves high-performance. However, an extensive comparison is complicated by source code not being available for some algorithms. In addition, we expect that further training would improve performance as validation errors did not diverge from training errors. For comparison, our DNN was trained for about ten days on two Nvidia GTX 1080 Ti GPUs whereas Xception[231], which is randomly initialized as part of our DNN, was trained for one month on 60 Nvidia K80 GPUs for ImageNet[232] image classification. Indeed, I suspect that restarting DNN training with a pretrained Xception backbone may more quickly achieve much higher performance than continuing training from my pretrained models. Finally, sufficiently deep and wide ANNs are universal approximators[233–241], so denoising DNNs can always outperform or match the accuracy of other methods developed by humans.

A few aspects of my DNN architecture and optimization are peculiar as our paper presents some of my earliest experiments with deep learning. For example, learning rates were stepwise decayed at irregular "wall clock" times. Further, large decreases in errors when learning rates were decreased may indicate that learning rates were too high. Another issue is that ReLU6[242] activation does not significantly outperform ReLU[243,244] activation, so ReLU is preferable as it requires less computation. Finally, I think that my DNN is too large for electron micrograph denoising. We justified that training can be continued and provide pretrained models; however, I doubt that training on the scale of Xception is practical insofar that most electron microscopists do not readily have access to more than a few GPUs for DNN training. I investigated smaller DNNs, which achieved lower performance. However, I expect that their performance could have been improved by further optimization of their training and architecture. In any case, I think that future DNNs for TEM denoising should be developed with automatic machine learning[245–249]

(AutoML) as AutoML can balance accuracy and training time, and can often outperform human developers[250,251].

My denoiser has higher errors near output image edges. Higher errors near image edges were also observed for compressed sensing with spiral[4] and uniformly spaced grid[19] scans (ch. 4). Indeed, the structured systematic errors of my denoiser partially motivated my investigations of structured systematic errors in compressed sensing. To avoid higher errors at output edges, I overlap parts of images that my denoiser is applied to so that edges of outputs where errors are higher can be discarded. However, discarding parts of denoiser outputs is computationally inefficient. To reduce structured systematic errors, I tried weighting contributions of output pixel errors to training losses by multiplying pixel errors by their exponential moving averages[4]. However, weighting errors did not have a significant effect. Nevertheless, I expect that higher variation of pixel weights could reduce systematic errors. Moreover, I propose that weights for output pixel errors could be optimized during DNN training to minimize structured systematic errors.

Chapter 7

Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning

7.1 Scientific Paper

This chapter covers the following paper[7] and its supplementary information[12].

> J. M. Ede, J. J. P. Peters, J. Sloan, and R. Beanland. Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning. *arXiv preprint arXiv:2001.10938 (under review by Ultramicroscopy)*, 2020
>
> J. M. Ede, J. J. P. Peters, J. Sloan, and R. Beanland. Supplementary Information: Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning. Zenodo, Online: `https://doi.org/10.5281/zenodo.4277357`, 2020

Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning

Jeffrey M. Ede[a,*], Jonathan J. P. Peters[a], Jeremy Sloan[a], Richard Beanland[a]

[a] *Department of Physics, University of Warwick, Coventry, England, CV4 7AL*

Abstract

Half of wavefunction information is undetected by conventional transmission electron microscopy (CTEM) as only the intensity, and not the phase, of an image is recorded. Following successful applications of deep learning to optical hologram phase recovery, we have developed neural networks to recover phases from CTEM intensities for new datasets containing 98340 exit wavefunctions. Wavefunctions were simulated with clTEM multislice propagation for 12789 materials from the Crystallography Open Database. Our networks can recover 224×224 wavefunctions in ∼25 ms for a large range of physical hyperparameters and materials, and we demonstrate that performance improves as the distribution of wavefunctions is restricted. Phase recovery with deep learning overcomes the limitations of traditional methods: it is live, not susceptible to distortions, does not require microscope modification or multiple images, and can be applied to any imaging regime. This paper introduces multiple approaches to CTEM phase recovery with deep learning, and is intended to establish starting points to be improved upon by future research. Source code and links to our new datasets and pre-trained models are available at `https://github.com/Jeffrey-Ede/one-shot`.

Keywords: deep learning, electron microscopy, exit wavefunction reconstruction

---

1. Introduction

Information transfer by electron microscope lenses and correctors can be described by wave optics[1] as electrons exhibit wave-particle duality[2, 3]. In a model electron microscope, a system of condenser lenses directs electrons illuminating a material into a planar wavefunction, $\psi_{\text{inc}}(\mathbf{r}, z)$, with wavevector, $\mathbf{k}$. Here, z is distance along its optical axis in the electron propagation direction, described by unit vector $\hat{\mathbf{z}}$, and $\mathbf{r}$ is the position in a plane perpendicular to the optical axis. As $\psi_{\text{inc}}(\mathbf{r}, z)$ travels through a material in fig. 1a, it is perturbed to an exit wavefunction, $\psi_{\text{exit}}(\mathbf{r}, z)$, by a material potential.

The projected potential of a material in direction $\hat{\mathbf{z}}$, $U(\mathbf{r}, z)$, and corresponding structural information can be calculated from $\psi_{\text{exit}}(\mathbf{r}, z)$[4, 5]. For example,

$$U(\mathbf{r}) \approx \frac{\text{Im}(\psi_{\text{exit}}(\mathbf{r}, z)\exp(i\varphi) - \langle \psi_{\text{exit}}(\mathbf{r}, z)\rangle_{\mathbf{r}})}{\lambda \xi \sin(\pi z/\xi)}, \tag{1}$$

for a typical crystal system well-approximated by two Bloch waves[4]. Here φ is a distance between Bloch wavevectors, λ is the electron wavelength, ξ is an extinction distance for two Bloch waves, $\langle ... \rangle_{\mathbf{r}}$ denotes an average with respect to $\mathbf{r}$, and $\text{Im}(z)$ is the imaginary part of z. Other applications of $\psi_{\text{exit}}(\mathbf{r}, z)$[6] include information storage, point spread function deconvolution, improving contrast, aberration correction[7], thickness measurement[8], and electric and magnetic structure determination[9, 10]. Exit wavefunctions can also simplify comparison with simulations as no information is lost.

In general, the intensity, $I(S)$, of a measurement with support, S, is

$$I(S) = \int_{s \in S} |\psi(s)|^2 \, \mathrm{d}s. \tag{2}$$

A support is a measurement region, such as an electron microscope camera[11, 12] element. Half of wavefunction information is lost at measurement as $|\psi|^2$ is a function of amplitude, $A > 0$, and not phase, $\theta \in [-\pi, \pi)$,

$$|\psi|^2 = |A \exp(i\theta)|^2 = A^2 |\exp(i\theta)|^2 = A^2. \tag{3}$$

---

[*] Corresponding author

Email addresses: `j.m.ede@warwick.ac.uk` (Jeffrey M. Ede), `j.peters.1@warwick.ac.uk` (Jonathan J. P. Peters), `j.sloan@warwick.ac.uk` (Jeremy Sloan), `r.beanland@warwick.ac.uk` (Richard Beanland)

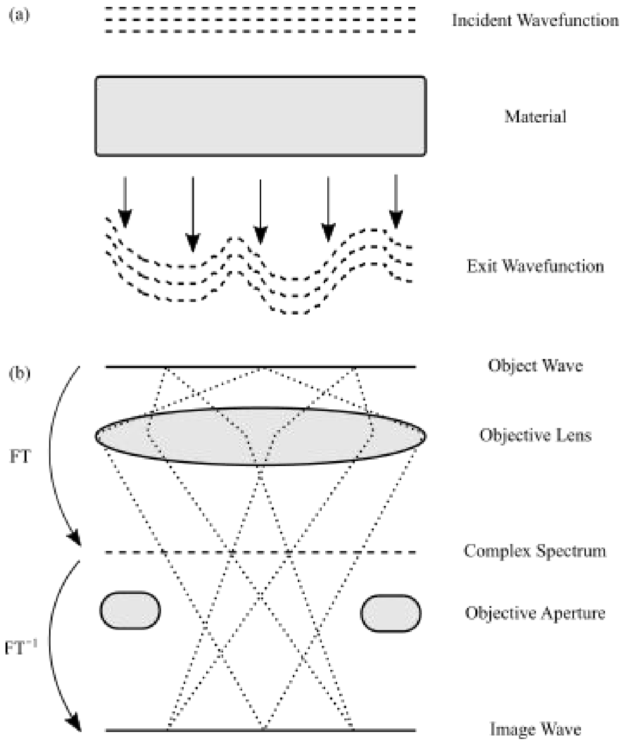


Figure 1: Wavefunction propagation. a) An incident wavefunction is perturbed by a projected potential of a material. b) Fourier transforms (FTs) can describe a wavefunction being focused by an objective lens through an objective aperture to a focal plane.

We emphasize that we define A to be positive so that $|\psi|^2 \mapsto A$ is bijective, and ψ sign information is in $\exp(i\theta)$. Phase information loss is a limitation of conventional single image approaches to electron microscopy, including transmission electron microscopy[13] (TEM), scanning transmission electron microscopy[14] (STEM), and scanning electron microscopy[15] (SEM).

In the Abbe theory of wave optics[16] in fig. 1b, the projection of ψ to a complex spectrum, $\psi_{\mathrm{dif}}(\mathbf{q})$, in reciprocal space, $\mathbf{q}$, at the back focal plane of an objective lens can be described by a Fourier transform (FT)

$$\psi_{\mathrm{dif}}(\mathbf{q}) = \mathrm{FT}[\psi_{\mathrm{exit}}(\mathbf{r})] = \int \psi_{\mathrm{exit}}(\mathbf{r}) \exp(-2\pi i \mathbf{q} \cdot \mathbf{r})\, \mathrm{d}\mathbf{r}. \tag{4}$$

In practice, $\psi_{\mathrm{dif}}(\mathbf{q})$ is perturbed to ψ_{pert} by an objective aperture, E_{ap}, coherence, E_{coh}, chromatic aberration, E_{chr}, and lens aberrations, χ, and is described in the Fourier domain[1] by

$$\psi_{\mathrm{pert}}(\mathbf{q}) = E_{\mathrm{ap}}(\mathbf{q}) E_{\mathrm{coh}}(\mathbf{q}) E_{\mathrm{chr}}(\mathbf{q}) \exp(-i\chi(\mathbf{q})) \psi_{\mathrm{dif}}(\mathbf{q}) \tag{5}$$

where

$$E_{\mathrm{ap}}(\mathbf{q}) = \begin{cases} 1, & \text{for } |\mathbf{q}| \leq k\theta_{\mathrm{max}} \\ 0, & \text{for } |\mathbf{q}| > k\theta_{\mathrm{max}} \end{cases} \tag{6}$$

$$E_{\mathrm{coh}}(\mathbf{q}) = \exp\left(-\frac{(\nabla\chi(\mathbf{q}))^2 (k\theta_{\mathrm{coh}})^2}{4\ln(2)} \right) \tag{7}$$

$$E_{\mathrm{chr}}(\mathbf{q}) = \exp\left(-\frac{1}{2}\left(\pi k C_{\mathrm{c}} \frac{\Delta E}{U_a^*} \left(\frac{q}{k}\right)^2 \right)^2 \right) \tag{8}$$

$$\chi(\theta, \phi) = \sum_{n=0}^{\infty} \sum_{m=0}^{n+1} \frac{C_{n,m,a} \theta^{n+1} \cos(m\phi)}{n+1} + \frac{C_{n,m,b} \theta^{n+1} \sin(m\phi)}{n+1} \tag{9}$$

for an objective aperture with angular extent, θ_{max}, illumination aperture with angular extent, θ_{coh}, energy spread, ΔE, chromatic aberration coefficient of the objective lens, C_{c}, relativistically corrected acceleration voltage, U_a^*, aberration coefficients, $C_{n,m,a}$ and $C_{n,m,b}$, angular inclination of perturbed wavefronts to the optical axis, ϕ, angular position in a plane perpendicular to the optical axis, θ, $m, n \in \mathbb{N}_0$, and $m + n$ is odd.

All waves emanating from points in Fourier space interfere in the image plane to produce an image wave, $\psi_{\mathrm{img}}(\mathbf{r})$, mathematically described by an inverse Fourier transform (FT^{-1})

$$\psi_{\mathrm{img}}(\mathbf{r}) = \mathrm{FT}^{-1}(\psi_{\mathrm{pert}}(\mathbf{q})) = \int \psi_{\mathrm{pert}}(\mathbf{q}) \exp(2\pi i \mathbf{q} \cdot \mathbf{r})\, \mathrm{d}\mathbf{q}. \tag{10}$$

Information transfer from ψ_{exit} to measured intensities can be modified by changing χ. Typically, by controlling the focus of the objective lens. However, half of ψ_{exit} information is missing from each measurement. To overcome this limitation, a wavefunction can be iteratively fitted to a series of aligned images with different χ[17, 18, 19, 20]. However, collecting an image series, waiting for sample drift to decay, and iterative fitting delays each ψ_{exit} measurement. As a result, aberration series reconstruction is unsuitable for live exit wavefunction reconstruction.

Electron holography[1, 18, 21] is an alternative approach to exit wavefunction reconstruction that compares ψ_{exit} to a reference wave. Typically, a hologram, I_{hol}, is created by moving a material off-axis and introducing an electrostatic biprism after the objective aperture. The Fourier transform of a

Möllenstedt biprismatic hologram is[1]

$$\begin{aligned}\mathrm{FT}(I_{\mathrm{hol}}(\mathbf{r})) = \mathrm{FT}(1 + |\psi_{\mathrm{exit}}(\mathbf{r})|^2) + \\ \mu\mathrm{FT}(\psi_{\mathrm{exit}}(\mathbf{r})) \otimes \delta(\mathbf{q} - \mathbf{q}_c) + \\ \mu\mathrm{FT}(\psi^*_{\mathrm{exit}}(\mathbf{r})) \otimes \delta(\mathbf{q} + \mathbf{q}_c),\end{aligned} \tag{11}$$

where $\psi^*_{\mathrm{exit}}(\mathbf{r})$ is the complex conjugate of $\psi_{\mathrm{exit}}(\mathbf{r})$, $|\mathbf{q}_c|$ is the carrier frequency of interference fringes, and their contrast,

$$\mu = |\mu_{\mathrm{coh}}||\mu_{\mathrm{inel}}||\mu_{\mathrm{inst}}|MTF, \tag{12}$$

is given by source spatiotemporal coherence, μ_{coh}, inelastic interactions, μ_{inst}, instabilities, μ_{inst}, and the modulation transfer function[22], MTF, of a detector. Convolutions with Dirac δ in eqn. 11 describe sidebands in Fourier space that can be cropped, centered, and inverse Fourier transformed for live exit wavefunction reconstruction. However, off-axis holograms are susceptible to distortions and require meticulous microscope alignment as phase information is encoded in interference fringes[1], and cropping Fourier space reduces resolution[21].

Artificial neural networks (ANNs) have been trained to recover phases of optical holograms from single images[23]. In general, this is not possible as there are an infinite number of physically possible θ for a given A. However, ANNs are able to leverage an understanding of the physical world to recover θ if the distribution of possible holograms is restricted, for example, to biological cells. Non-iterative methods that do not use ANNs to recover phase information from single images have also been developed. However, they are limited to defocused images in the Fresnel regime[24], or to non-planar incident wavefunctions in the Fraunhofer regime[25].

One-shot phase recovery with ANNs overcomes the limitations of traditional methods: it is live, not susceptible to off-axis holographic distortions, does not require microscope modification, and can be applied to any imaging regime. In addition, ANNs could be applied to recover phases of images in large databases, long after samples may have been lost or destroyed. In this paper, we investigate the application of deep learning to one-shot exit wavefunction reconstruction in conventional transmission electron microscopy (CTEM).

2. Exit Wavefunction Datasets

To showcase one-shot exit wavefunction reconstruction, we generated 98340 exit wavefunctions with clTEM[27, 28] multislice propagation for 12789 CIFs[29] downloaded from the Crystallography Open Database[30, 31, 32, 33, 34, 35] (COD). Complex 64 bit 512×512 wavefunctions were simulated for CTEM with acceleration voltages in $\{80, 200, 300\}$ kV, material depths along the optical axis uniformly distributed in $[5, 100)$ nm, material widths perpendicular to the optical axis in $[5, 10)$ nm, and crystallographic zone axes (h, k, l) $h, k, l \in \{0, 1, 2\}$. Materials are padded on all sides with 0.8 nm of vacuum in the image plane, and 0.3 nm along the optical axis, to reduce simulation artefacts. Finally, crystal tilts to each axis were perturbed by zero-centered Gaussian random variates with standard deviation 0.1°. We used default values for other clTEM hyperparameters.

Multislice exit wavefunction simulations with clTEM are based on [36]. Simulations start with a planar wavefunction, ψ, travelling along a TEM column

$$\psi(x, y, z) = \exp\left(\frac{2\pi i z}{\lambda}\right), \tag{13}$$

where x and y are in-plane coordinates, and z is distance travelled. After passing through a thin specimen, with thickness Δz, wavefunctions are approximated by

$$\psi(x, y, z + \Delta z) \simeq \exp(i\sigma V_z(x, y)\Delta z)\,\psi(x, y, z) \tag{14}$$

with

$$\sigma = \frac{2\pi m e\lambda}{h^2}, \tag{15}$$

where V_z is the projected potential of the specimen at z, m is relativistic electron mass, e is fundamental electron charge, and h is Planck's constant.

For electrons propagating through a thicker specimen, cumulative phase change can described by a specimen transmission function, $t(x, y, z)$, so that

$$\psi(x, y, z + \Delta z) = t(x, y, z)\,\psi(x, y, z) \tag{16}$$

with

$$t(x, y, z) = \exp\left(i\sigma \int_z^{z+\Delta z} V(x, y, z')\,\mathrm{d}z'\right). \tag{17}$$

A thin sample can be divided into multiple thin slices stacked together using a propagator function, P, to map wavefunctions between slices. A wavefunction at slice n is mapped to a wavefunction at slice $n + 1$ by

$$\psi_{n+1}(x, y) \leftarrow P(x, y, \Delta z) \otimes [t_n(x, y)\,\psi_n(x, y,)] \tag{18}$$

where ψ_0 is the incident wave in eqn. 13. Simulations with clTEM are based on OpenCL[37], and use

Dataset	n	Train	Unseen	Validation	Test	Total
Multiple Materials	1	25325	1501	3569	8563	38958
Multiple Materials	3	24530	1544	3399	8395	37868
Multiple Materials, Restricted	3	8002	-	1105	2763	11870
$In_{1.7}K_2Se_8Sn_{2.28}$	1	3856	-	963	-	4819
$In_{1.7}K_2Se_8Sn_{2.28}$	3	3861	-	964	-	4825

Table 1: New datasets containing 98340 wavefunctions simulated with clTEM are split into training, unseen, validation, and test sets. Unseen wavefunctions are simulated for training set materials with different simulation hyperparameters. Kirkland potential summations were calculated with $n = 3$ or truncated to $n = 1$ terms, and dashes (-) indicate subsets that have not been simulated. Datasets have been made publicly available at [26].

graphical processing units (GPUs) to accelerate fast Fourier transform[38] (FFT) based convolutions. The propagator is calculated in reciprocal space

$$P\left(k_x, k_y\right) = \exp\left(-i\pi\lambda k^2\Delta z\right), \tag{19}$$

where k_x, k_y are reciprocal space coordinates, and $k = (k_x^2 + k_y^2)^{1/2}$. As Fourier transforms are used to map between reciprocal and real space, propagator and transmission functions are band limited to decrease aliasing.

Projected atomic potentials are calculated using Kirkland's parameterization[36], where the projected potential of an atom at position, p, in a thin slice is

$$\begin{aligned} v_p(x,y) = 4\pi^2 e r_\text{Bohr} \sum_i^n a_i K_0\left(2\pi r_p b_i^{1/2}\right) + \\ 2\pi^2 e r_\text{Bohr} \sum_i^n \frac{c_i}{d_i} \exp\left(-\frac{\pi^2 r_p^2}{d_i}\right), \end{aligned} \tag{20}$$

where $r_p = [(x - x_p)^2 + (y - y_p)^2]^{1/2}$, x_p and y_p are the coordinates of the atom, r_Bohr is the Bohr radius, K_0 is the modified Bessel function[39], and the parameters a_i, b_i, c_i, and d_i are tabulated for each atom in [36]. Nominally, $n = 3$. However, we also use $n = 1$ to investigate robustness to alternative simulation physics. In effect, simulations with $n = 1$ are for an alternative universe where atoms have different potentials. Every atom in a slice contributes to the total projected potential

$$V_z = \sum_p v_p. \tag{21}$$

After simulation, a 320×320 region was selected from the center of each wavefunction to remove edge artefacts. Each wavefunction was divided by its magnitude to prevent an ANN from inferring information from an absolute intensity scale. In practice, it is possible to measure an absolute scale; however, it is specific to a microscope and its configuration.

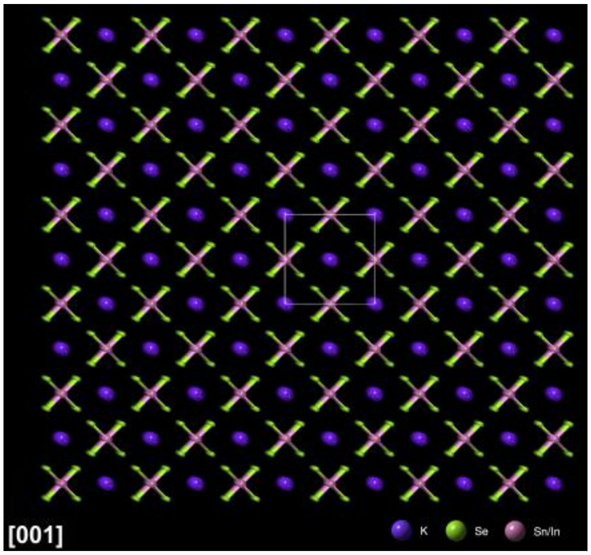


Figure 2: Crystal structure of $In_{1.7}K_2Se_8Sn_{2.28}$ projected along Miller zone axis [001]. A square outlines a unit cell.

To investigate ANN performance for multiple materials, we partitioned 12789 CIFs into training, validation, and test sets by journal of publication. There are 8639 training set CIFs: 150 New Journal of Chemistry, 1034 American Mineralogist, 1998 Journal of the American Chemical Society, and 5457 Inorganic Chemistry. In addition, there are 1216 validation set CIFs published in Physics and Chemistry of Materials, and 2927 test set CIFs published in Chemistry of Materials. Wavefunctions were simulated for three random sets of hyperparameters for each CIF, except for a small portion of examples that were discarded because CIF format or simulation hyperparameters were unsupported. Partitioning by journal helps to test the ability of an ANN to generalize given that wavefunction characteristics are expected to vary with journal.

New simulated wavefunction datasets are tabulated in table 1 and have been made publicly available at [26]. In total, 76826 wavefunction have been simulated for multiple materials. To investigate ANN performance as the distribution of possible wavefunctions is restricted, we also simulated 11870 wavefunctions with smaller simulation hyperparameter upper bounds that reduce ranges by factors close to 1/4. In addition, we simulated 9644 wavefunctions for a randomly selected single material, $In_{1.7}K_2Se_8Sn_{2.28}$[40], shown in fig. 2. Datasets were simulated for Kirkland potential summations in eqn. 20 to $n = 3$, or truncated to $n = 1$ terms. Truncating summations allows alternative simulation physics to be investigated.

3. Artificial Neural Networks

To reconstruct an exit wavefunction, ψ_{exit}, from its amplitude, A, an ANN must recover missing phase information, θ. However, $\theta \in [-\infty, \infty]$, and restricting phase support to one period of the phase is complicated by cyclic periodicity. Instead, it is convenient to predict a periodic function of the phase with finite support. We use two output channels in fig. 3 to predict phase components, $\cos\theta$ and $\sin\theta$, where $\psi = A(\cos\theta + i\sin\theta)$.

Each convolutional layer[41, 42] is followed by batch normalization[43], then activation, except the last layer where no activation is applied. Convolutional layers in residual blocks[44] are ReLU[45] activated, whereas slope 0.1 leaky ReLU[46] activation is used after other convolutional layers to avoid dying ReLU[47, 48, 49]. In denomination, channelwise L2 normalization imposes the identity $|\exp(i\theta)| \equiv 1$ after the final convolutional layer.

In initial experiments, batch normalization was frozen halfway through training, similar to [50]. However, scale invariance before L2 normalization resulted in numerical instability. As a result, we updated batch normalization parameters throughout training. Adding a secondary objective to impose a single output scale; such as a distance between mean L2 norms and unity, slowed training. Nevertheless, L2 normalization can be removed for generators that converge to low errors if $|\exp(i\theta)| \equiv 1$ is implicitly imposed by their loss functions.

For direct prediction, generators were trained by ADAM optimized[51] stochastic gradient descent[52, 53] for $i_{\text{max}} = 5 \times 10^5$ iterations to minimize adaptive learning rate clipped[54] (ALRC) mean squared errors (MSEs) of phase components. Training losses were calculated by multiplying MSEs by 10 and ALRC layers were initialized with first raw moment $\mu_1 = 25$, second raw moment $\mu_2 = 30$, exponential decay rates $\beta_1 = \beta_2 = 0.999$, and $n = 3$ standard deviations. We used an initial learning rate $\eta_0 = 0.002$, which was stepwise exponentially decayed[55] by a factor of 0.5 every $i_{\text{max}}/7$ iterations, and a first moment of the momentum decay rate, $\beta_1 = 0.9$.

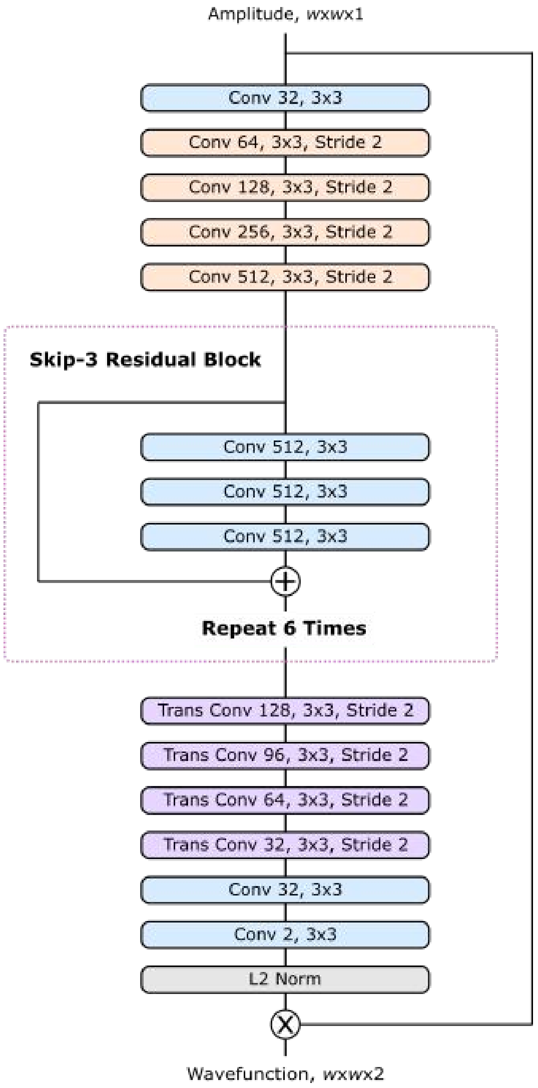


Figure 3: A convolutional neural network generates $w \times w \times 2$ channelwise concatenations of wavefunction components from their amplitudes. Training MSEs are calculated for phase components, before multiplication by input amplitudes.

In practice, wavefunctions with similar amplitudes may make output phase components ambiguous. As a result, a MSE trained generator may predict a weighted mean of multiple probable phase outputs, even if it understands that one pair of phase components is more likely. To overcome this limitation, we propose training a generative adversarial network[56] (GAN) to predict most probable outputs. Specifically, we propose training a discriminator, D, in fig. 4 for a function, f, of amplitudes, and real and generated output phase components. This will enable an adversarial generator,

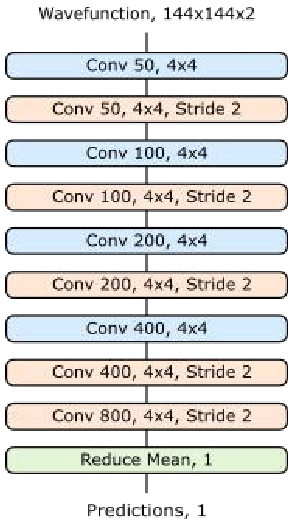


Figure 4: A discriminator predicts if wavefunction components were generated by a neural network.

G, to learn to output realistic phases in the context of their amplitudes.

There are many popular GAN loss functions and regularization mechanisms[57, 58]. Following [59], we use mean squared generator, L_G, and discriminator, L_D, losses, and apply spectral normalization to the weights of every convolutional layer in the discriminator

$$L_D = (D(f(\psi)) - 1)^2 + D(f(G(|\psi|)))^2 \qquad (22)$$

$$L_G = (D(f(G(|\psi|)) - 1)^2, \qquad (23)$$

where f is a function that parameterizes ψ as the channelwise concatenation of $\{A\cos\theta, A\sin\theta\}$. Multiplying generated phase components by inputted A conditions wavefunction discrimination on A, ensuring that the generator learns to output physically probable θ. Other parameterizations; such as the channelwise concatenation of $\{A, \cos\theta, \sin\theta\}$ could also be used. There are no biases in the discriminator.

Concatenation of conditional information to discriminator inputs and feature channels is investigated in [60, 61, 62, 63, 64, 65, 66, 67]. Projection discriminators, which calculate inner products of generator outputs and conditional embeddings, are an alternative that achieve higher performance in [68]. However, blind compression to an embedded representation would reduce wavefunction information, potentially limiting the quality of generated wavefunctions, and may encourage catastrophic forgetting[69].

Both generator and discriminator training was ADAM optimized for 5×10^5 iterations with base learning rate $\eta_G = \eta_D = 0.0002$, and first moment of the momentum decay, $\beta_1 = 0.5$. To balance generator and discriminator learning, we map the discriminator learning rate to

$$\eta'_D = \frac{\eta_D}{1 + \exp(-m(\mu_D - c))}, \qquad (24)$$

where μ_D is the running mean discrimination for generated wavefunctions, $D(f(G(|\psi|))$, tracked by an exponential moving average with a decay rate of 0.99, and $m = 20$ and $c = 0.5$ linearly transform μ_D.

To augment training data, we selected random $w \times w$ crops from 320×320 wavefunctions. Each crop was then subject to random combination of flips and $\pi/2$ rad rotations to augment our datasets by a factor of eight. We chose wavefunction size $w = 224$ for direct prediction and $w = 144$ for GANs, where w is smaller for GANs as discriminators add to GPU memory requirements. ANNs were trained with a batch size of 24.

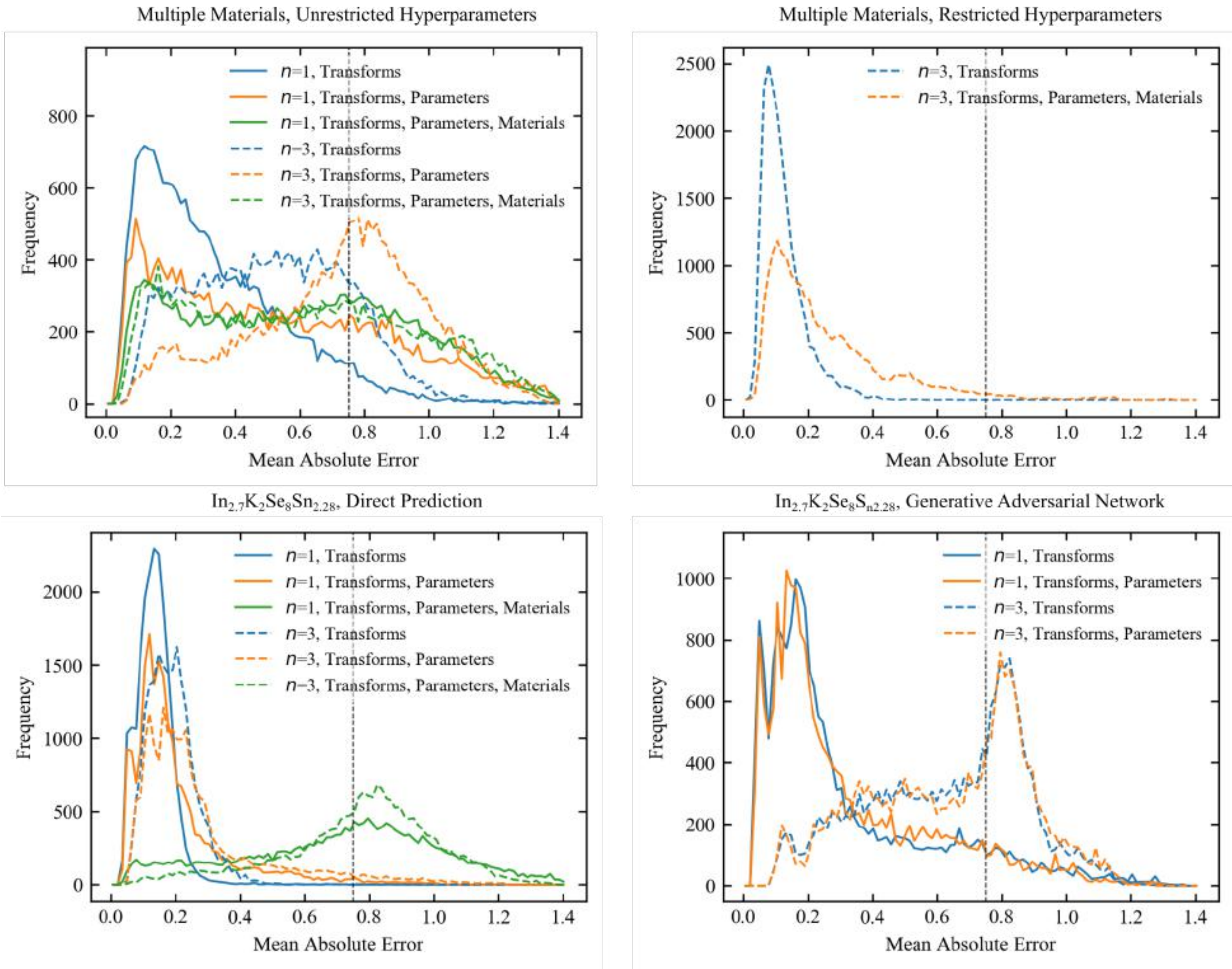
4. Experiments

In this section, we investigate phase recovery with ANNs as the distribution of wavefunctions is restricted. To directly predict θ for A, we trained ANNs for multiple materials, multiple materials with restricted simulation hyperparameters, and $In_{1.7}K_2Se_8Sn_{2.28}$. We also trained a GAN for $In_{1.7}K_2Se_8Sn_{2.28}$ wavefunctions. Experiments are repeated with the summation in eqn. 20 truncated from $n = 3$ to $n = 1$, to demonstrate robustness to simulation physics.

Distributions of generated phase component mean absolute errors (MAEs) for sets of 19992 validation examples are shown in fig. 5, and moments are tabulated in table 2. We used up to three validation sets, which cumulatively quantify the ability of a network to generalize to unseen transforms; combinations of flips, rotations and translations, simulation hyperparameters; such as thickness and voltage, and materials. In comparison, the expected error of the nth moment of phase components, $\mathrm{E}[|G(|\psi|) - f(\theta)|^n]$, where $g \in \{\cos, \sin\}$, for uniform random predictions, $x \sim U(-1, 1)$, and uniformly distributed phases, $\theta \sim U(-\pi, \pi)$, is

$$\mathrm{E}[|x - g(\theta)|^n] = \int_{-1}^{1} \int_{-\pi}^{\pi} \rho(x)\rho(\theta)|x - g(\theta)|^n \, \mathrm{d}\theta \, \mathrm{d}x, \qquad (25)$$

where $\rho(\theta) = 1/2\pi$ and $\rho(x) = 1/2$ are uniform probability density functions for θ and x, respectively.


Multiple Materials, Unrestricted Hyperparameters

Multiple Materials, Restricted Hyperparameters

$In_{2.7}K_2Se_8Sn_{2.28}$, Direct Prediction

$In_{2.7}K_2Se_8S_{n2.28}$, Generative Adversarial Network

Figure 5: Frequency distributions show 19992 validation set mean absolute errors for neural networks trained to reconstruct wavefunctions simulated for multiple materials, multiple materials with restricted simulation hyperparameters, and $In_{1.7}K_2Se_8Sn_{2.28}$. Networks for $In_{1.7}K_2Se_8Sn_{2.28}$ were trained to predict phase components directly; minimising squared errors, and as part of generative adversarial networks. To demonstrate robustness to simulation physics, some validation set errors are shown for $n = 1$ and $n = 3$ simulation physics. We used up to three validation sets, which cumulatively quantify the ability of a network to generalize to unseen transforms; combinations of flips, rotations and translations, simulation hyperparameters; such as thickness and voltage, and materials. A vertical dashed line indicates an expected error of 0.75 for random phases, and frequencies are distributed across 100 bins.

		Trans.		Trans., Param.		Trans., Param., Mater.	
Training Scope	n	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
Multiple Materials, Unrestricted Parameters	1	0.333	0.220	0.525	0.341	0.600	0.334
$In_{1.7}K_2Se_8Sn_{2.28}$, MSE	1	0.135	0.056	0.205	0.157	0.708	0.310
$In_{1.7}K_2Se_8Sn_{2.28}$, GAN	1	0.318	0.279	0.321	0.256	-	-
Multiple Materials, Unrestricted Parameters	3	0.513	0.234	0.717	0.271	0.614	0.344
Multiple Materials, Restricted Parameters	3	0.123	0.069	-	-	0.260	0.192
$In_{1.7}K_2Se_8Sn_{2.28}$, MSE	3	0.190	0.079	0.281	0.208	0.768	0.235
$In_{1.7}K_2Se_8Sn_{2.28}$, GAN	3	0.633	0.244	0.638	0.249	-	-
Uniform Random Phases (Max Entropy)	1, 3	0.750	0.520	0.750	0.520	0.750	0.520

Table 2: Means and standard deviations of 19992 validation set errors for unseen transforms (trans.), simulations hyperparameters (param.) and materials (mater.). All networks outperform a baseline uniform random phase generator for both $n = 1$ and $n = 3$ simulation physics. Dashes (-) indicate that validation set wavefunctions have not been simulated.

The first two moments are $E[|x - g(\theta)|] = 3/4$ and $E[|x - g(\theta)|^2] = 5/6$; making the expected standard deviation 0.520.

All ANN MAEs have lower means and standard deviations than a baseline random phase generator, except a $In_{1.7}K_2Se_8Sn_{2.28}$ generator applied to other materials. ANNs do not have prior understanding of propagation equations or dynamics. As a result, experiments demonstrate that ANNs are able to develop and leverage a physical understanding to recover θ. ANNs are trained for Kirkland potential summations in eqn. 20 to $n = 3$ and $n = 1$ terms, demonstrating a robustness to simulation physics. Success with different simulation physics motivates the development of ANNs for real physics; approximated by $n = 3$ simulation physics.

Validation set MAEs increase as wavefunction restrictions are cumulatively reduced from unseen transforms used for data augmentation during training, to unseen simulation parameters, and unseen materials. For example, MAEs are 0.600 and 0.614 for ANNs trained for multiple materials, increasing to 0.708 and 0.768 for ANNs trained for $In_{1.7}K_2Se_8Sn_{2.28}$. This shows that MAEs increase for materials an ANN is unfamiliar with, approaching MAEs of 0.75 expected for a uniform random phase generator where there is no familiarity.

Wavefunctions are insufficiently restricted for multiple materials. Validation MAEs of 0.333 and 0.513 for unseen transforms diverge to 0.600 and 0.614 for unseen simuation hyperparamaters and materials. In addition, a peak near 0.15 decreases, and MAE density around 0.75 increases. Taken together, this indicates that multiple material ANNs are able to recognise and generalize to some wavefunctions; however, their ability to generalize is limited. Further, frequency distribution tails exceed 0.75 for all validation sets. This may indicate that the generator struggles with material and simulation or hyperparameter combinations that produce wavefunctions with unusual or unpalatable characteristics. However, we believe the tail is mainly caused by combinations that produce different wavefunctions with similar amplitudes.

Validation divergence decreases as the distribution of wavefunctions is restricted. For example, frequency distributions have almost no tail beyond 0.75 for simulation hyperparameter ranges reduced by factors close to 1/4. Validation divergence is also reduced by training for $In_{1.7}K_2Se_8Sn_{2.28}$, a single material. Restricting the distribution of wavefunctions is an essential part of one-shot wavefunction reconstruction, otherwise there is an infinite number of possible θ for A.

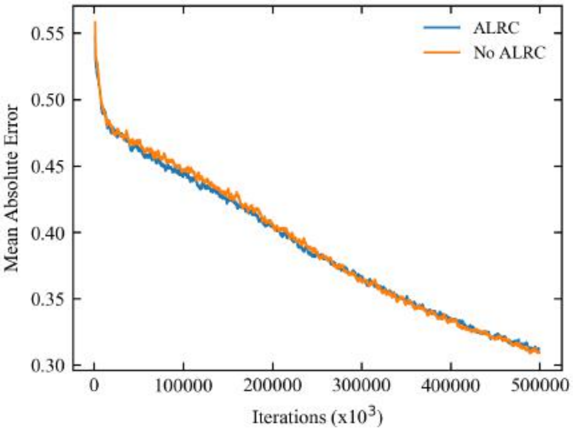


Figure 6: Training mean absolute errors are similar with and without adaptive learning rate clipping (ALRC). Learning curves are 2500 iteration boxcar averaged.

To investigate an approach to reduce prediction weighting for A with a range of probable θ, we trained GANs for $In_{1.7}K_2Se_8Sn_{2.28}$. Training as part of a GAN acts as a regularization mechanism, lowering validation divergence. However, a GAN requires a powerful discriminator to understand the distribution of possible wavefunctions and can be difficult to train. In particular, $n = 3$ wavefunctions have lower local spatial correlation than $n = 1$ wavefunctions at our simulation resolution, which made it more difficult for our $n = 3$ GAN to learn.

Training loss distributions have tails with high losses. As a result, we used ALRC to limit high errors. A comparison of training with and without ALRC is in fig. 6. Validation MAEs for unseen materials have mean 0.600 and standard deviation 0.334 with ALRC, and mean 0.602 and standard deviation 0.338 without ALRC. Differences between validation MAEs is insignificant, so ALRC is not helping for training with batch size 24. This behavior is in-line with results in the ALRC paper[54], which shows that ALRC becomes less effective as batch size increases. Nevertheless, ALRC may be help lower error if generators are trained with smaller batch sizes. In particular, if the wavefunction distribution is restricted so errors are low, removing the need for L2 normalization at the end of the generator, and therefore decreasing dependence on batch normalization.

Examples of ANN phase recovery are shown in fig. 7 alongside crystal structures highlighting the structural information producing exit wavefunctions. Results are for unseen materials and an ANN trained for multiple materials with restricted simulation hyperparameters. Wavefunctions are presented for NaCl[70] and

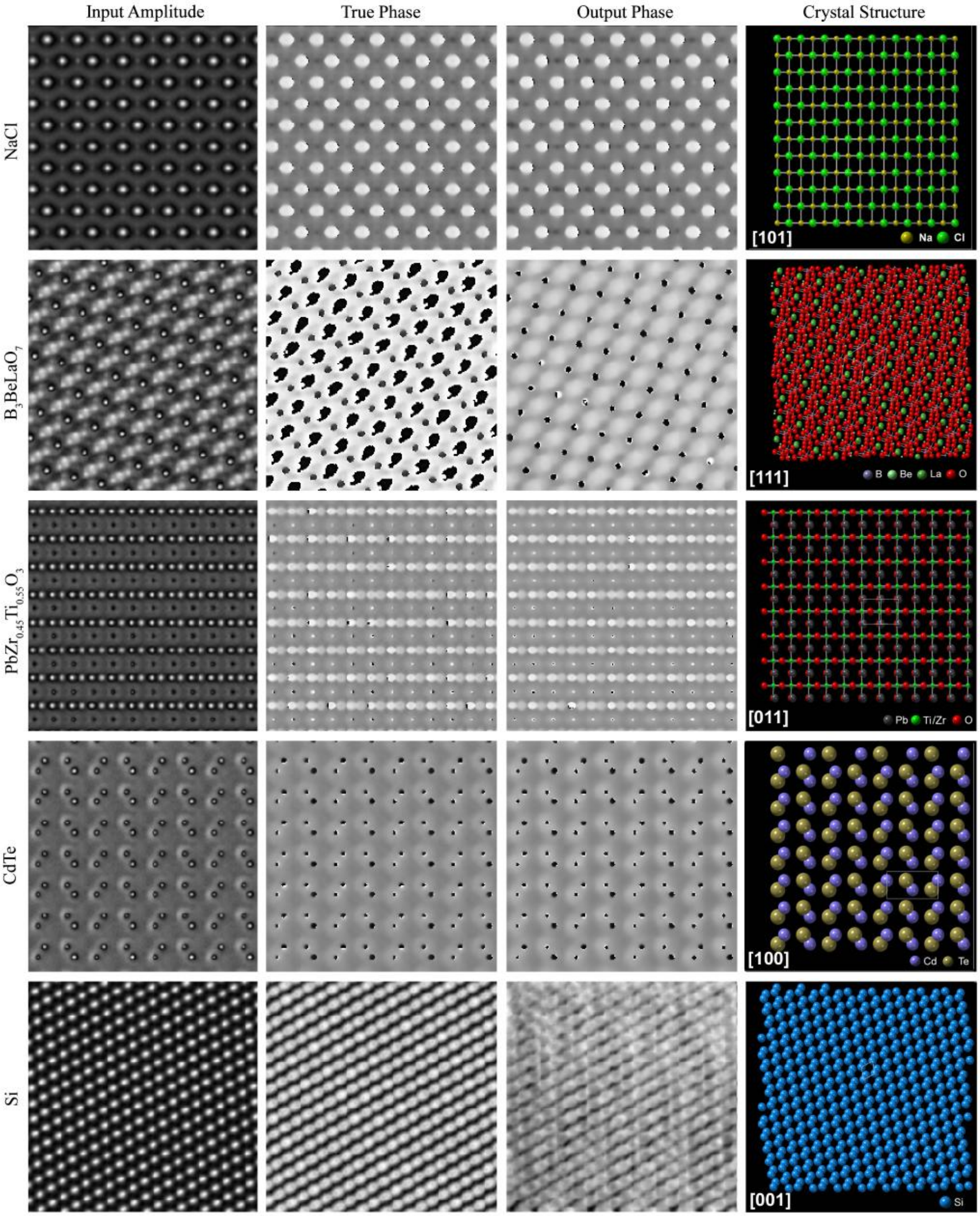


Figure 7: Exit wavefunction reconstruction for unseen NaCl, B_3BeLaO_7, $PbZr_{0.45}Ti_{0.55}0_3$, CdTe, and Si input amplitudes, and corresponding crystal structures. Phases in $[-\pi, \pi)$ rad are depicted on a linear greycale from black to white, and show that output phases are close to true phases. Wavefunctions are cyclically periodic functions of phase so distances between black and white

elemental Si as they are simple materials with widely recognised structures. Other materials belong to classes that are widely investigated: B_3BeLaO_7[71] is a non-linear optical crystal, $PbZr_{0.45}Ti_{0.55}0_3$[72] is ferroelectric used in ultrasonic transducers[73] and ceramic capacitors[74], and CdTe is a semiconductor used in solar cells[75]. The Si example is also included as typical failure case for unfamiliar examples. In this case, possibly because the Si crystal structure is unusually simple. Additional sheets of example input phases, generated phases, and true phases for each ANN will be provided as supplementary information with the published version of this preprint.

5. Discussion

This paper describes an initial investigation into CTEM one-shot exit wavefunction reconstruction with deep learning, and is intended to be a starting point for future research. We expect that ANN architecture and learning policy can be substantially improved; possibly with AdaNet[76], Ludwig[77], or other automatic machine learning[78] algorithms, and we encourage further investigation. In this spirit, all of our source code[79] (based on TensorFlow[80]), clTEM simulation software[27], and new wavefunction datasets[26] have been made publicly available. Training for each network was stopped after a few days on an Nvidia 1080 Ti GPU, and losses were still decreasing. As a result, this paper presents lower bounds for performance.

To demonstrate robustness to simulation physics, Kirkland potential summations in eqn. 20 were calculated with $n = 3$, or truncated to $n = 1$ terms, for different datasets. For further simulations, compiled clTEM versions with $n = 1$ and $n = 3$ have been included in our project repository[79]. Source code for clTEM is also available with separate pre-releases[27]. Summations with $n = 3$ approximate experimental physics, whereas $n = 1$ is for an alternative universe with different atom potentials.

Our experiments do not include aberrations or detector noise. This restricts the distribution of wavefunctions and makes it easier for ANNs to learn. However, distributions of wavefunctions were less restricted than possible in practice, and ANNs can remove noise[81]. As a result, we expect one-shot exit wavefunction to be applicable to experimental images. A good starting point for future research may be materials where the distribution of wavefunctions is naturally restricted. For example, graphene[82] and other two-dimensional materials[83], select crystals at atomic resolution[84], or classified images; such as biological specimens[85, 86] after similar preparation.

Information about materials, expected ranges of simulation hyperparameters, and other metadata was not input to ANNs. However, this variable information is readily available and could restrict the distribution of wavefunctions; improving ANN performance. Subsequently, we suggest that metadata embedded by an ANN could be used to modulate information transfer through a convolutional neural network by conditional batch normalization[87]. However, metadata is typically high-dimensional, so this may be impractical beyond individual applications.

By default, large amounts of metadata is saved to Digital Micrograph image files (e.g. dm3 and dm4) created by Gatan Microscopy Suite[88] software. Metadata can also be saved to TIFFs[89] or other image formats preferred by electron microscopists using different software. In practice, most of this metadata describes microscope settings; such as voltage and magnification, and may not be sufficient to restrict the distribution of wavefunctions. Nevertheless, most file formats support the addition of extra metadata that is readily known to experimenters. Example information may include estimates for stoichiometry, specimen thickness, zone axis, temperature, the microscope and its likely aberration range, and phenomena exhibited by materials in scientific literature. ANNs have been developed to embed scientific literature[90], so we expect that it will become possible to include additional metadata as a lay description.

In this paper, ANNs are trained to reconstruct ψ from A, and therefore follow a history of successful deep learning applications to accelerated quantum mechanics[91, 92]. In contrast, experimental holograms are integrated over detector supports. Although probability density, $|\psi(\bar{S})|^2$, at the mean support, $\bar{S}$, can be factored outside the integral of eqn. 2 if spatial variation is small, $\nabla\chi \rightarrow 0$, and S is effectively invariant,

$$I(S) \approx |\psi(\bar{S})|^2 \int_{s \in S} \mathrm{d}s, \tag{26}$$

these restrictions are unrealistic. In practice, we do not think the distinction is important as ANNs have learned to recover optical θ from I[23].

To discourage ANNs from gaming their loss functions by predicting an average of probable phase components, we propose training GANs. However, GANs are difficult to train[93, 69], and GAN training can take longer than with MSEs. For example, our validation set GAN MAEs are lower than for MSE training after 5×10^5 iterations. We also found that GAN performance can be much lower for some wavefunctions; such as those with low local spatial correlation. High performance for large wavefunctions also requires powerful discriminators; such as [94], to understand their distribution.

Overall, we expect GANs to become less useful the more a distribution of wavefunctions is restricted. As the distribution becomes more restricted, a smaller portion of the distribution has similar amplitudes with substantially different phases. In part, we expect this effect already lowers MAEs as distributions are restricted. Another contribution is restricted physics; which makes networks less reliant on identifying features. As a result, we expect the main use of GANs in phase recovery to be improving wavefunction realism.

6. Conclusions

We have simulated five new datasets containing 98340 CTEM exit wavefunctions with clTEM. The datasets have been used to train ANNs to reconstruct wavefunctions from single images. In this initial investigation, we found that ANN performance improves as the distribution of wavefunctions is restricted. One-shot exit wavefunction reconstruction overcomes the limitations of aberration series reconstruction and holography: it is live, does not require experimental equipment, and can be applied as a post-processing step indefinitely after an image is taken. We expect our results to be generalizable to other types of electron microscopy.

7. Supplementary Information

This work is intended to establish starting points to be improved on by future research. In this spirit, our new datasets[26], clTEM simulation software[27], and source code with links to pre-trained models[79] has been made publicly available.

In appendices, we build on Abbe's theory of wave optics to propose a new approach to phase recovery with deep learning. The idea is that wavefunctions could be learned from large datasets of single images; avoiding the difficulty and expense of collecting experimental wavefunctions. Nevertheless, we also introduce a new dataset containing 1000 512×512 experimental focal series. In addition, a supplementary document will be provided with the published version of this preprint with sheets of example input amplitudes, output phases, and true phases for every ANN featured in this paper.

References

[1] M. Lehmann, H. Lichte, Tutorial on off-axis electron holography, Microscopy and Microanalysis 8 (6) (2002) 447–466.

[2] S. Frabboni, G. C. Gazzadi, G. Pozzi, Youngs double-slit interference experiment with electrons, American Journal of Physics 75 (11) (2007) 1053–1055.

[3] G. Matteucci, C. Beeli, An experiment on electron wave–particle duality including a Planck constant measurement, American Journal of Physics 66 (12) (1998) 1055–1059.

[4] M. Lentzen, K. Urban, Reconstruction of the projected crystal potential in transmission electron microscopy by means of a maximum-likelihood refinement algorithm, Acta Crystallographica Section A: Foundations of Crystallography 56 (3) (2000) 235–247.

[5] A. Auslender, M. Halabi, G. Levi, O. Diéguez, A. Kohn, Measuring the mean inner potential of Al_2O_3 sapphire using off-axis electron holography, Ultramicroscopy 198 (2019) 18–25.

[6] A. Tonomura, Applications of electron holography, Reviews of modern physics 59 (3) (1987) 639.

[7] Q. Fu, H. Lichte, E. Völkl, Correction of aberrations of an electron microscope by means of electron holography, Physical review letters 67 (17) (1991) 2319.

[8] M. McCartney, M. Gajdardziska-Josifovska, Absolute measurement of normalized thickness, t/λ_i, from off-axis electron holography, Ultramicroscopy 53 (3) (1994) 283–289.

[9] H. S. Park, X. Yu, S. Aizawa, T. Tanigaki, T. Akashi, Y. Takahashi, T. Matsuda, N. Kanazawa, Y. Onose, D. Shindo, et al., Observation of the magnetic flux and three-dimensional structure of skyrmion lattices by electron holography, Nature nanotechnology 9 (5) (2014) 337.

[10] R. E. Dunin-Borkowski, T. Kasama, A. Wei, S. L. Tripp, M. J. Hÿtch, E. Snoeck, R. J. Harrison, A. Putnis, Off-axis electron holography of magnetic nanowires and chains, rings, and planar arrays of magnetic nanoparticles, Microscopy research and technique 64 (5-6) (2004) 390–402.

[11] G. McMullan, A. Faruqi, R. Henderson, Direct electron detectors, in: Methods in enzymology, Vol. 579, Elsevier, 2016, pp. 1–17.

[12] G. McMullan, S. Chen, R. Henderson, A. Faruqi, Detective quantum efficiency of electron area detectors in electron microscopy, Ultramicroscopy 109 (9) (2009) 1126–1143.

[13] C. B. Carter, D. B. Williams, Transmission electron microscopy: Diffraction, imaging, and spectrometry, Springer, 2016.

[14] S. J. Pennycook, P. D. Nellist, Scanning transmission electron microscopy: imaging and analysis, Springer Science & Business Media, 2011.

[15] J. I. Goldstein, D. E. Newbury, J. R. Michael, N. W. Ritchie, J. H. J. Scott, D. C. Joy, Scanning electron microscopy and X-ray microanalysis, Springer, 2017.

[16] H. Köhler, On Abbe's theory of image formation in the microscope, Optica Acta: International Journal of Optics 28 (12) (1981) 1691–1701.

[17] A. Lubk, K. Vogel, D. Wolf, J. Krehl, F. Röder, L. Clark, G. Guzzinati, J. Verbeeck, Fundamentals of focal series inline electron holography, in: Advances in imaging and electron physics, Vol. 197, Elsevier, 2016, pp. 105–147.

[18] C. T. Koch, A. Lubk, Off-axis and inline electron holography: A quantitative comparison, Ultramicroscopy 110 (5) (2010) 460–471.

[19] C. T. Koch, Towards full-resolution inline electron holography, Micron 63 (2014) 69–75.

[20] S. Haigh, B. Jiang, D. Alloyeau, C. Kisielowski, A. Kirkland,

Recording low and high spatial frequencies in exit wave reconstructions, Ultramicroscopy 133 (2013) 26–34.

[21] C. Ozsoy-Keskinbora, C. Boothroyd, R. Dunin-Borkowski, P. Van Aken, C. Koch, Hybridization approach to in-line and off-axis (electron) holography for superior resolution and phase sensitivity, Scientific Reports 4 (2014) 7020.

[22] R. S. Ruskin, Z. Yu, N. Grigorieff, Quantitative characterization of electron detectors for transmission electron microscopy, Journal of structural biology 184 (3) (2013) 385–393.

[23] Y. Rivenson, Y. Zhang, H. Günaydın, D. Teng, A. Ozcan, Phase recovery and holographic image reconstruction using deep learning in neural networks, Light: Science & Applications 7 (2) (2018) 17141.

[24] A. Morgan, A. Martin, A. D'Alfonso, C. Putkunz, L. Allen, Direct exit-wave reconstruction from a single defocused image, Ultramicroscopy 111 (9-10) (2011) 1455–1460.

[25] A. Martin, L. Allen, Direct retrieval of a complex wave from its diffraction pattern, Optics Communications 281 (20) (2008) 5114–5121.

[26] J. M. Ede, J. P. P. Peters, R. Beanland, Warwick electron microscopy datasets, online: `https://warwick.ac.uk/fac/sci/physics/research/condensedmatt/microscopy/research/machinelearning` (2019).

[27] J. P. P. Peters, M. A. Dyson, clTEM, online: `https://github.com/JJPPeters/clTEM` (2019).

[28] M. A. Dyson, Advances in computational methods for transmission electron microscopy simulation and image processing, Ph.D. thesis, University of Warwick (2014).

[29] S. R. Hall, F. H. Allen, I. D. Brown, The crystallographic information file (CIF): a new standard archive file for crystallography, Acta Crystallographica Section A: Foundations of Crystallography 47 (6) (1991) 655–685.

[30] M. Quirós, S. Gražulis, S. Girdzijauskaitė, A. Merkys, A. Vaitkus, Using SMILES strings for the description of chemical connectivity in the Crystallography Open Database, Journal of Cheminformatics 10 (1) (May 2018). `doi:10.1186/s13321-018-0279-6`.

[31] A. Merkys, A. Vaitkus, J. Butkus, M. Okulič-Kazarinas, V. Kairys, S. Gražulis, COD::CIF::Parser: an error-correcting CIF parser for the Perl language, Journal of Applied Crystallography 49 (1) (Feb 2016). `doi:10.1107/S1600576715022396`.
URL `http://dx.doi.org/10.1107/S1600576715022396`

[32] S. Gražulis, A. Merkys, A. Vaitkus, M. Okulič-Kazarinas, Computing stoichiometric molecular composition from crystal structures, Journal of Applied Crystallography 48 (1) (2015) 85–91. `doi:10.1107/S1600576714025904`.
URL `http://dx.doi.org/10.1107/S1600576714025904`

[33] S. Gražulis, A. Daškevič, A. Merkys, D. Chateigner, L. Lutterotti, M. Quirós, N. R. Serebryanaya, P. Moeck, R. T. Downs, A. Le Bail, Crystallography Open Database (COD): an open-access collection of crystal structures and platform for world-wide collaboration, Nucleic Acids Research 40 (D1) (2012) D420–D427. `arXiv:http://nar.oxfordjournals.org/content/40/D1/D420.full.pdf+html`, `doi:10.1093/nar/gkr900`.
URL `http://nar.oxfordjournals.org/content/40/D1/D420.abstract`

[34] S. Gražulis, D. Chateigner, R. T. Downs, A. F. T. Yokochi, M. Quirós, L. Lutterotti, E. Manakova, J. Butkus, P. Moeck, A. Le Bail, Crystallography Open Database – an open-access collection of crystal structures, Journal of Applied Crystallography 42 (4) (2009) 726–729. `doi:10.1107/S0021889809016690`.
URL `http://dx.doi.org/10.1107/S0021889809016690`

[35] R. T. Downs, M. Hall-Wallace, The American Mineralogist crystal structure database, American Mineralogist 88 (2003) 247–250.

[36] E. J. Kirkland, Advanced computing in electron microscopy, Springer Science & Business Media, 2010.

[37] J. E. Stone, D. Gohara, G. Shi, OpenCL: A parallel programming standard for heterogeneous computing systems, Computing in science & engineering 12 (3) (2010) 66.

[38] K. Moreland, E. Angel, The FFT on a GPU, in: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, Eurographics Association, 2003, pp. 112–119.

[39] M. Abramowitz, I. A. Stegun, Handbook of mathematical functions. 1965 (1964).

[40] S.-J. Hwang, R. G. Iyer, P. N. Trikalitis, A. G. Ogden, M. G. Kanatzidis, Cooling of melts: Kinetic stabilization and polymorphic transitions in the $\mathrm{KInSnSe_4}$ system, Inorganic chemistry 43 (7) (2004) 2237–2239.

[41] M. T. McCann, K. H. Jin, M. Unser, Convolutional neural networks for inverse problems in imaging: A review, IEEE Signal Processing Magazine 34 (6) (2017) 85–95.

[42] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[43] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv:1502.03167 (2015).

[44] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition. CoRR abs/1512.03385 (2015).

[45] V. Nair, G. E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Proceedings of the 27th international conference on machine learning (ICML-10), 2010, pp. 807–814.

[46] A. L. Maas, A. Y. Hannun, A. Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: Proc. ICML, Vol. 30, 2013, p. 3.

[47] L. Lu, Y. Shin, Y. Su, G. E. Karniadakis, Dying ReLU and initialization: Theory and numerical examples, arXiv preprint arXiv:1903.06733 (2019).

[48] S. C. Douglas, J. Yu, Why ReLU units sometimes die: Analysis of single-unit error backpropagation in neural networks, in: 2018 52nd Asilomar Conference on Signals, Systems, and Computers, IEEE, 2018, pp. 864–868.

[49] B. Xu, N. Wang, T. Chen, M. Li, Empirical evaluation of rectified activations in convolutional network, arXiv preprint arXiv:1505.00853 (2015).

[50] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, Encoder-decoder with atrous separable convolution for semantic image segmentation, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 801–818.

[51] D. P. Kingma, J. Ba, ADAM: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[52] S. Ruder, An overview of gradient descent optimization algorithms, arXiv preprint arXiv:1609.04747 (2016).

[53] D. Zou, Y. Cao, D. Zhou, Q. Gu, Stochastic gradient descent optimizes over-parameterized deep relu networks, arXiv preprint arXiv:1811.08888 (2018).

[54] J. M. Ede, R. Beanland, Adaptive learning rate clipping stabilizes learning, arXiv preprint arXiv:1906.09060 (2019).

[55] R. Ge, S. M. Kakade, R. Kidambi, P. Netrapalli, The step decay schedule: A near optimal, geometrically decaying learning rate procedure, arXiv preprint arXiv:1904.12838 (2019).

[56] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in neural information processing systems, 2014, pp. 2672–2680.

[57] Z. Wang, Q. She, T. E. Ward, Generative adversarial networks: A survey and taxonomy, arXiv preprint arXiv:1906.01529 (2019).

[58] H.-W. Dong, Y.-H. Yang, Towards a deeper understanding of adversarial losses, arXiv preprint arXiv:1901.08753 (2019).

[59] T. Miyato, T. Kataoka, M. Koyama, Y. Yoshida, Spectral normalization for generative adversarial networks, arXiv preprint arXiv:1802.05957 (2018).

[60] M. Mirza, S. Osindero, Conditional generative adversarial nets, arXiv preprint arXiv:1411.1784 (2014).

[61] E. L. Denton, S. Chintala, R. Fergus, et al., Deep generative image models using a Laplacian pyramid of adversarial networks, in: Advances in neural information processing systems, 2015, pp. 1486–1494.

[62] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, H. Lee, Generative adversarial text to image synthesis, arXiv preprint arXiv:1605.05396 (2016).

[63] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, D. N. Metaxas, Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 5907–5915.

[64] G. Perarnau, J. Van De Weijer, B. Raducanu, J. M. Álvarez, Invertible conditional GANs for image editing, arXiv preprint arXiv:1611.06355 (2016).

[65] M. Saito, E. Matsumoto, S. Saito, Temporal generative adversarial nets with singular value clipping, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2830–2839.

[66] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, A. Courville, Adversarially learned inference, arXiv preprint arXiv:1606.00704 (2016).

[67] K. Sricharan, R. Bala, M. Shreve, H. Ding, K. Saketh, J. Sun, Semi-supervised conditional gans, arXiv preprint arXiv:1708.05789 (2017).

[68] T. Miyato, M. Koyama, cgans with projection discriminator, arXiv preprint arXiv:1802.05637 (2018).

[69] K. J. Liang, C. Li, G. Wang, L. Carin, Generative adversarial network training is a continual learning problem, arXiv preprint arXiv:1811.11083 (2018).

[70] S. Abrahams, J. Bernstein, Accuracy of an automatic diffractometer. Measurement of the sodium chloride structure factors, Acta Crystallographica 18 (5) (1965) 926–932.

[71] X. Yan, S. Luo, Z. Lin, Y. Yue, X. Wang, L. Liu, C. Chen, $LaBeB_3O_7$: A new phase-matchable nonlinear optical crystal exclusively containing the tetrahedral XO_4 (X=B and Be) anionic groups, Journal of Materials Chemistry C 1 (22) (2013) 3616–3622.

[72] Y. Idemoto, H. Yoshikoshi, N. Koura, K. Takeuchi, J. W. Richardson, C. K. Loong, Relation between the crystal structure, physical properties and ferroelectric properties of $PbZr_xTi_{1-x}O_3$ (x=0.40, 0.45, 0.53) ferroelectric material by heat treatment, Journal of the Ceramic Society of Japan 112 (1301) (2004) 40–45.

[73] Y. Chen, X. Bao, C.-M. Wong, J. Cheng, H. Wu, H. Song, X. Ji, S. Wu, PZT ceramics fabricated based on stereolithography for an ultrasound transducer array application, Ceramics International 44 (18) (2018) 22725–22730.

[74] M. Hikam, I. Irzaman, H. DarmasetiawanArifin, P. Arifin, M. Budiman, M. Barmawi, Pyroelectric properties of lead zirconium titanate ($PbZr_{0.525}Ti_{0.475}O_3$) metal-ferroelectric-metal capacitor and its application for IR sensor, Jurnal Sains Materi Indonesia 6 (3) (2018) 23–27.

[75] J. M. Burst, J. N. Duenow, D. S. Albin, E. Colegrove, M. O. Reese, J. A. Aguiar, C.-S. Jiang, M. Patel, M. M. Al-Jassim, D. Kuciauskas, et al., CdTe solar cells with open-circuit voltage breaking the 1 V barrier, Nature Energy 1 (3) (2016) 1–8.

[76] C. Weill, J. Gonzalvo, V. Kuznetsov, S. Yang, S. Yak, H. Mazzawi, E. Hotaj, G. Jerfel, V. Macko, B. Adlam, M. Mohri, C. Cortes, AdaNet: A scalable and flexible framework for automatically learning ensembles (2019). arXiv:1905.00080.

[77] P. Molino, Y. Dudin, S. S. Miryala, Ludwig: a type-based declarative deep learning toolbox, arXiv preprint arXiv:1909.07930 (2019).

[78] X. He, K. Zhao, X. Chu, AutoML: A survey of the state-of-the-art, arXiv preprint arXiv:1908.00709 (2019).

[79] J. M. Ede, One shot exit wavefunction reconstruction, online: https://github.com/Jeffrey-Ede/one-shot (2019).

[80] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: A system for large-scale machine learning., in: OSDI, Vol. 16, 2016, pp. 265–283.

[81] J. M. Ede, R. Beanland, Improving electron micrograph signal-to-noise with an atrous convolutional encoder-decoder, Ultramicroscopy 202 (2019) 18–25.

[82] C. Wang, C. Luo, X. Wu, Characterization and dynamic manipulation of graphene by in situ transmission electron microscopy at atomic scale, Handbook of Graphene: Physics, Chemistry, and Biology (2019) 291.

[83] R. G. Mendes, J. Pang, A. Bachmatiuk, H. Q. Ta, L. Zhao, T. Gemming, L. Fu, Z. Liu, M. H. Rümmelii, Electron-driven in situ transmission electron microscopy of 2D transition metal dichalcogenides and their 2D heterostructures, ACS nano 13 (2) (2019) 978–995.

[84] D. Zhang, Y. Zhu, L. Liu, X. Ying, C.-E. Hsiung, R. Sougrat, K. Li, Y. Han, Atomic-resolution transmission electron microscopy of electron beam–sensitive crystalline materials, Science 359 (6376) (2018) 675–679.

[85] M. Lakshman, Application of conventional electron microscopy in aquatic animal disease diagnosis: A review, Journal of Entomology and Zoology Studies 7 (2019) 470–475.

[86] Y. Ogawa, J.-L. Putaux, Transmission electron microscopy of cellulose. Part 2: technical and practical aspects, Cellulose 26 (1) (2019) 17–34.

[87] E. Perez, H. de Vries, F. Strub, V. Dumoulin, A. Courville, Learning visual reasoning without strong priors, arXiv preprint arXiv:1707.03017 (2017).

[88] Gatan, Gatan microscopy suite, online: www.gatan.com/products/tem-analysis/gatan-microscopy-suite-software (2019).

[89] A. D. Association, et al., TIFF revision 6.0, online: www.adobe.io/content/dam/udp/en/open/standards/tiff/TIFF6.pdf (1992).

[90] V. Tshitoyan, J. Dagdelen, L. Weston, A. Dunn, Z. Rong, O. Kononova, K. A. Persson, G. Ceder, A. Jain, Unsupervised word embeddings capture latent knowledge from materials science literature, Nature 571 (7763) (2019) 95–98.

[91] M. J. Beach, I. De Vlugt, A. Golubeva, P. Huembeli, B. Kulchytskyy, X. Luo, R. G. Melko, E. Merali, G. Torlai, QuCumber: Wavefunction reconstruction with neural networks, arXiv preprint arXiv:1812.09329 (2018).

[92] G. Carleo, K. Choo, D. Hofmann, J. E. Smith, T. Westerhout, F. Alet, E. J. Davis, S. Efthymiou, I. Glasser, S.-H. Lin, et al., NetKet: A machine learning toolkit for many-body quantum

systems, arXiv preprint arXiv:1904.00031 (2019).

[93] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, Improved techniques for training GANs, in: Advances in neural information processing systems, 2016, pp. 2234–2242.

[94] A. Brock, J. Donahue, K. Simonyan, Large scale GAN training for high fidelity natural image synthesis, arXiv preprint arXiv:1809.11096 (2018).

[95] H. Research, FTSR software, online: www.hremresearch.com/Eng/plugin/FTSREng.html (2019).

[96] MATLAB, version 9.5 (R2018b), The MathWorks Inc., Natick, Massachusetts, 2018.

8. Acknowledgements

Thanks go to Christoph T. Koch for software used to collect experimental focal series, to David Walker for suggesting materials in fig. 7, and to Jessica Marshall for feedback on fig. 7.

Funding: J.M.E. acknowledges EPSRC grant EP/N035437/1 and EPSRC Studentship 1917382 for financial support, R.B. acknowledge EPSRC grant EP/N035437/1 for financial support, J.J.P.P. acknowledges EPSRC grant EP/P031544/1 for financial support, and J.S. acknowledges EPSRC grant EP/R019428/1 for financial support.

Appendix A. Sharded Deep Holography

Collecting experimental CTEM holograms with a biprism or focal series reconstruction is expensive: Measuring a large number of representative holograms is time-intensive, and requires skilled electron microscopists to align and operate microscopes. In this context, we propose a new method to reconstruct holograms by extracting information from a large image database with deep learning. It is based on the idea that individual images are fragments of aberration series sampled from an aberration series distribution. To be clear, this section summarizes an idea and is intended to be a starting point for future work.

Let $\psi_{\text{exit}} \sim \Psi_{\text{exit}}$ denote an unknown exit wavefunction, ψ_{exit}, sampled from a distribution, Ψ_{exit}, $c \sim C$ denote an unknown contrast transfer function (CTF), $c = \psi_{\text{pert}}(\mathbf{q})/\psi_{\text{dif}}(\mathbf{q})$, sampled from a distribution, C, and $m \sim M$ denote metadata, m, sampled from a distribution, M, that restricts Ψ_{exit}. The image wave is

$$\psi_{\text{img}} = \text{FT}^{-1}(c\text{FT}(\psi_{\text{exit}})). \tag{27}$$

We propose introducing a faux CTF, $c' \sim C'$, to train a cycle-consistent generator, G, and discriminator, D, to predict the exit wave,

$$\hat{\psi}_{\text{exit}} = G(|\psi_{\text{img}}|, m). \tag{28}$$

The faux CTF can be used to generate an image wavefunction

$$\hat{\psi}'_{\text{img}} = \text{FT}^{-1}(c'\text{FT}(\hat{\psi}_{\text{exit}})). \tag{29}$$

If the faux distribution is realistic, D can be trained to discriminate between $|\hat{\psi}'_{\text{img}}|$ and $|\psi_{\text{img}}|$. For example, by minimizing the expected value of

$$L_D = D(|\psi_{\text{img}}|, m)^2 + (D(|\hat{\psi}'_{\text{img}}|, m') - 1)^2, \tag{30}$$

where $m' \neq m$ if metadata describes different CTFs. A cycle-consistent adversarial generator can then be trained to minimize the expected value of

$$\begin{aligned} L_G = D(|\hat{\psi}'_{\text{img}}|, m)^2 + \\ \lambda ||G(|\psi_{\text{img}}|, m) - G(|\hat{\psi}'_{\text{img}}|, m')||_2^2, \end{aligned} \tag{31}$$

where λ weights the contribution of the adversarial and cycle-consistency losses. The adversarial loss trains the generator to produce realistic wavefunctions, whereas the cycle-consistency loss trains the generator to learn unique solutions.

Alternatively, CTFs could be preserved by mapping

$$G(|\hat{\psi}'_{\text{img}}|, m) \rightarrow \text{FT}^{-1}(\text{FT}(G(|\hat{\psi}'_{\text{img}}|, m))/c'), \tag{32}$$

when calculating the L2 norm in eqn. 31. If CTFs are preserved by this mapping, c' is a relative; rather than absolute, CTF and cc' is the CTF of $\hat{\psi}'_{\text{img}}$.

Two of our experimental datasets containing 17267 TEM and 16227 STEM images are available with our new wavefunction datasets[26]. However, the images are unlabelled to anonymise contributors; limiting metadata available to restrict a distribution of wavefunctions.

Appendix B. Experimental Focal Series

As a potential starting point for experimental one-shot exit wavefunction reconstruction, we have made 1000 focal series publicly available[26]. We have also made simple focal series reconstruction code available at [79]. Alternatively, refined focal and tilt series reconstruction (FTSR) software is commercially available[95]. Each series consists of 14 32-bit 512×512 TIFFs, area downsampled from 4096×4096 with MATLAB[96] and default antialiasing. All series were created with a common, quadratically increasing[20] defocus series. However, spatial scales vary and must be fitted as part of reconstruction.

Supplementary Information: Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning

Jeffrey M. Ede, Jonathan J. P. Peters, Jeremy Sloan, and Richard Beanland

{j.m.ede, j.peters.1, j.sloan, r.beanland}@warwick.ac.uk

S1. ADDITIONAL EXAMPLES

Example applications of ANNs are shown in figs. S1-S18, and source code for every ANN is available in [1]. Phases in $[-\pi, \pi)$ rad are depicted on a linear greycale from black to white. Wavefunctions are cyclically periodic functions of phase so distances between black and white pixels are small.

REFERENCES

[1] J. M. Ede, "One shot exit wavefunction reconstruction." online: `https://github.com/Jeffrey-Ede/one-shot`, 2019.

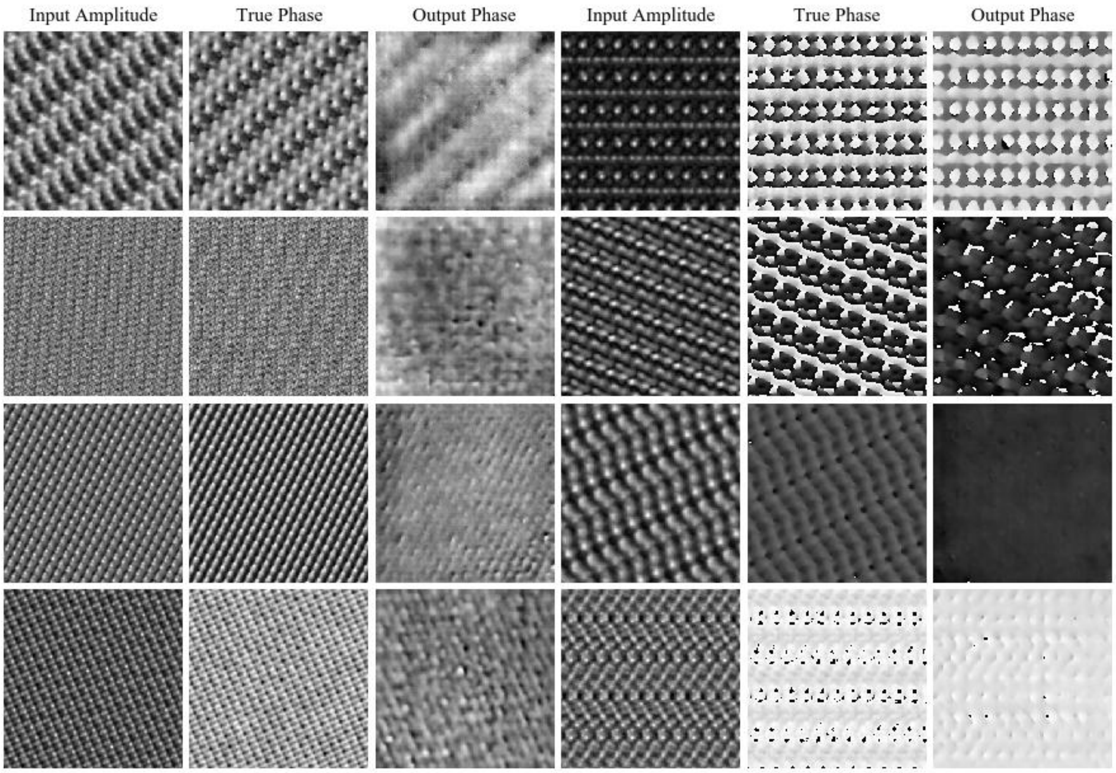


Fig. S1: Input amplitudes, target phases and output phases of 224×224 multiple material training set wavefunctions for unseen flips, rotations and translations, and $n = 1$ simulation physics.

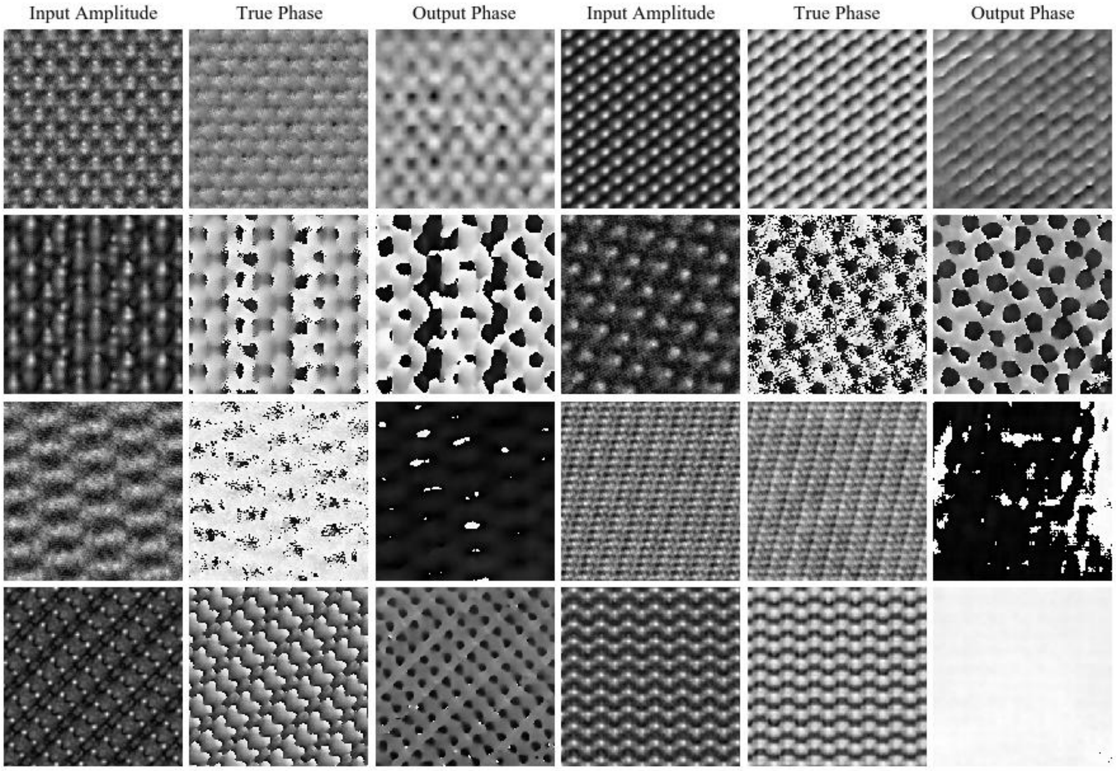


Fig. S2: Input amplitudes, target phases and output phases of 224×224 multiple material validation set wavefunctions for seen materials, unseen simulation hyperparameters, and $n = 1$ simulation physics.

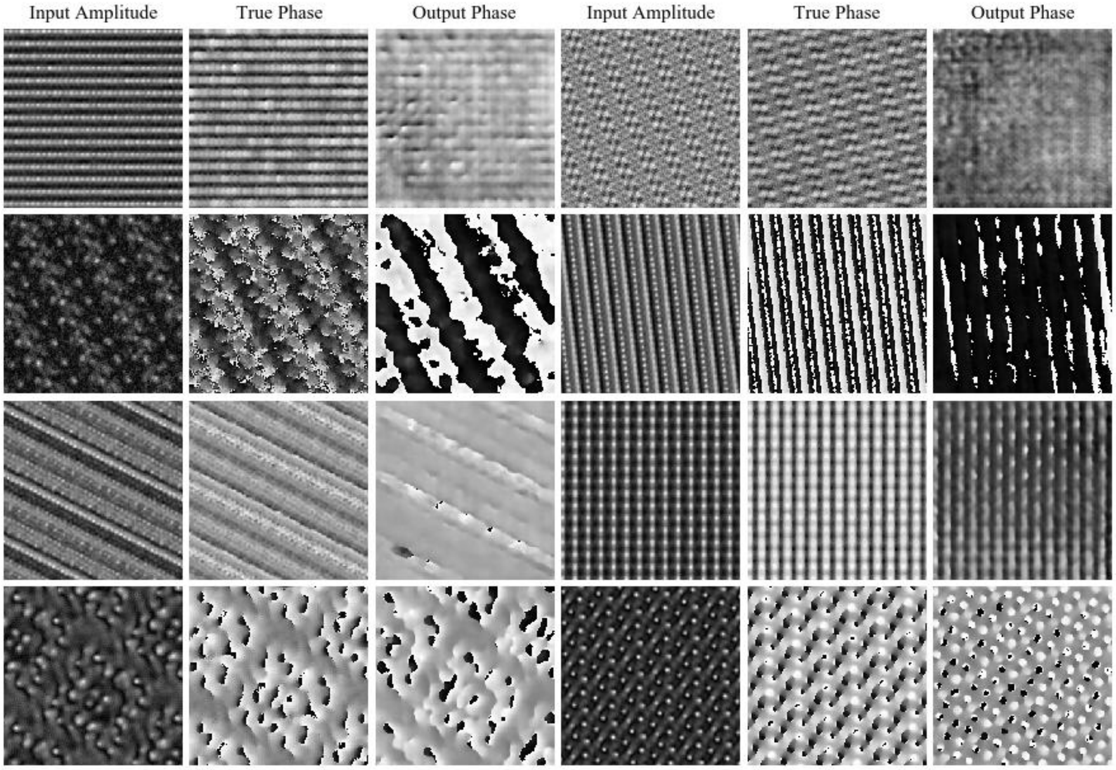


Fig. S3: Input amplitudes, target phases and output phases of 224×224 multiple material validation set wavefunctions for unseen materials,

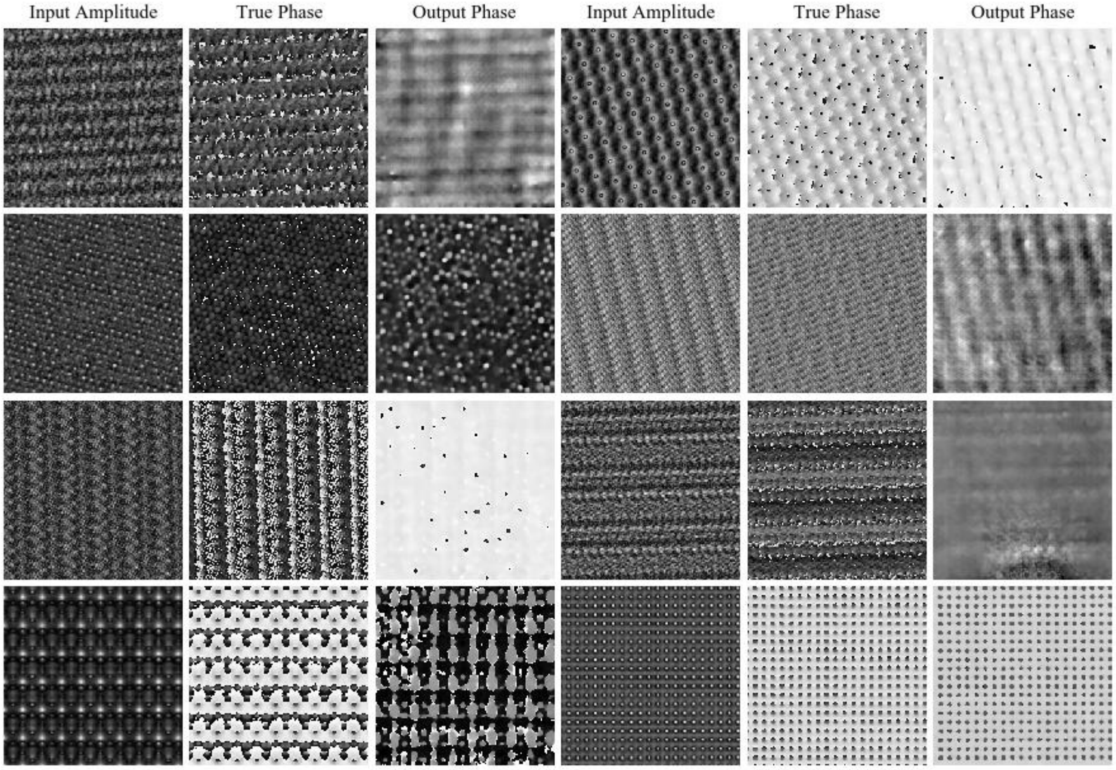


Fig. S4: Input amplitudes, target phases and output phases of 224×224 multiple material training set wavefunctions for unseen flips, rotations and translations, and $n = 3$ simulation physics.

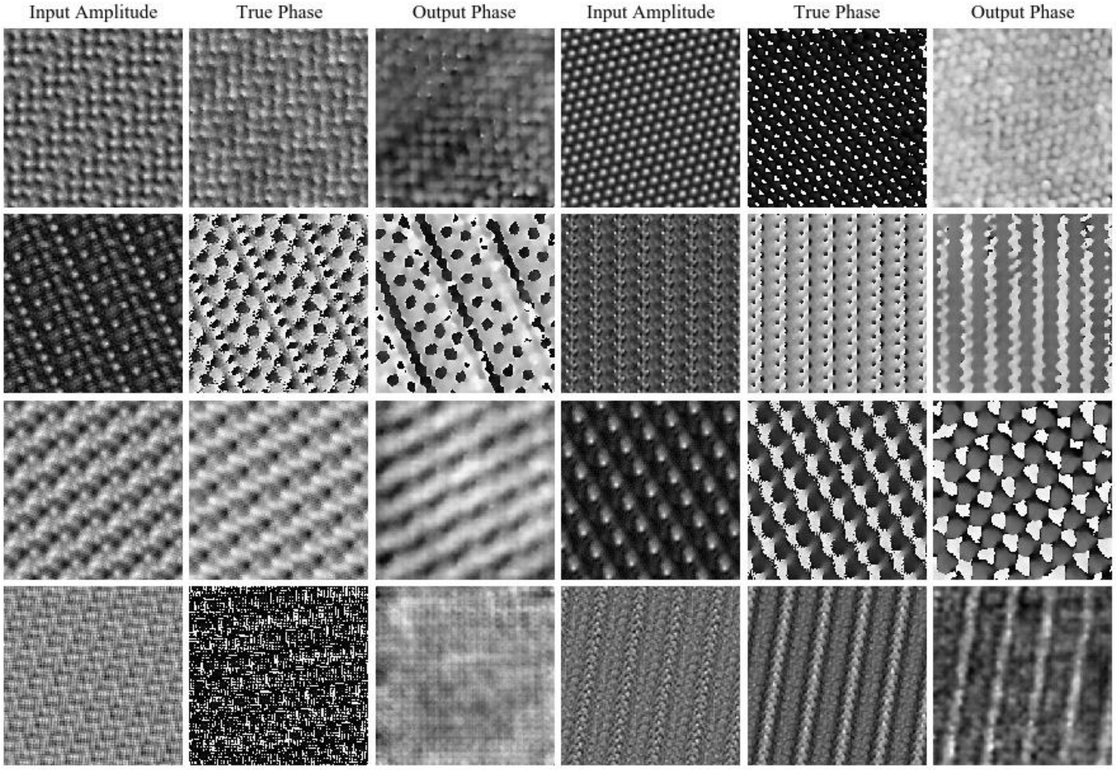


Fig. S5: Input amplitudes, target phases and output phases of 224×224 multiple material validation set wavefunctions for seen materials,

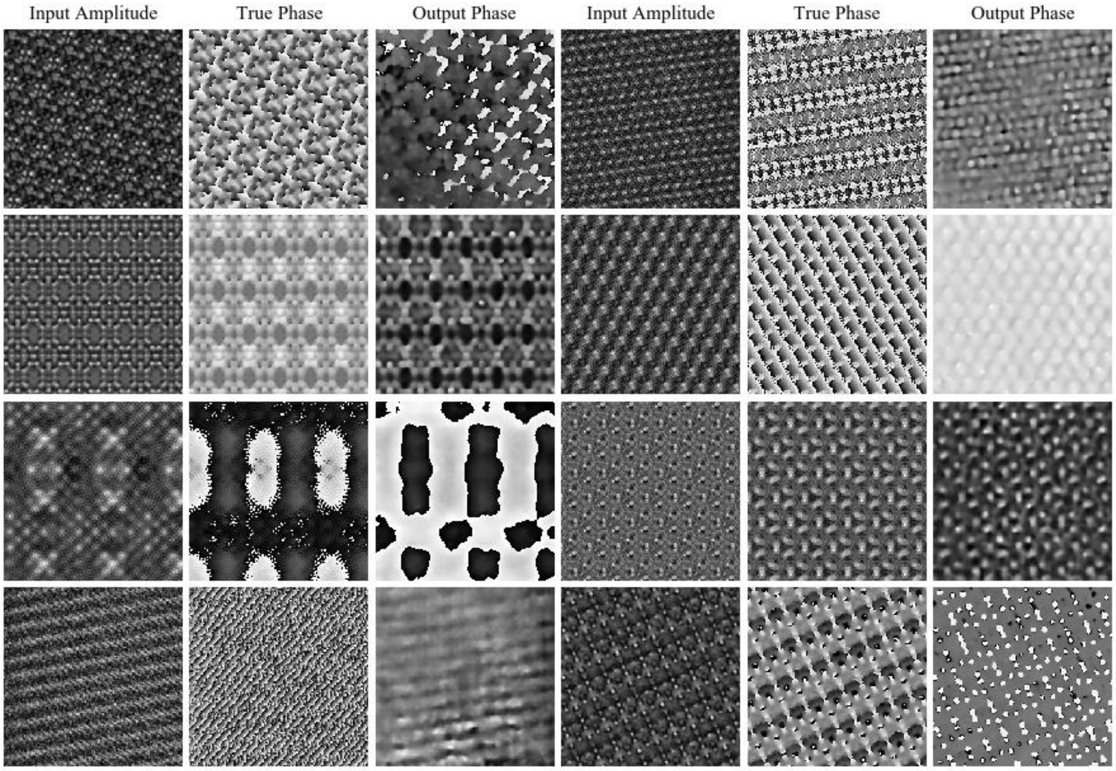


Fig. S6: Input amplitudes, target phases and output phases of 224×224 multiple material validation set wavefunctions for unseen materials, unseen simulation hyperparameters are unseen, and $n = 3$ simulation physics.

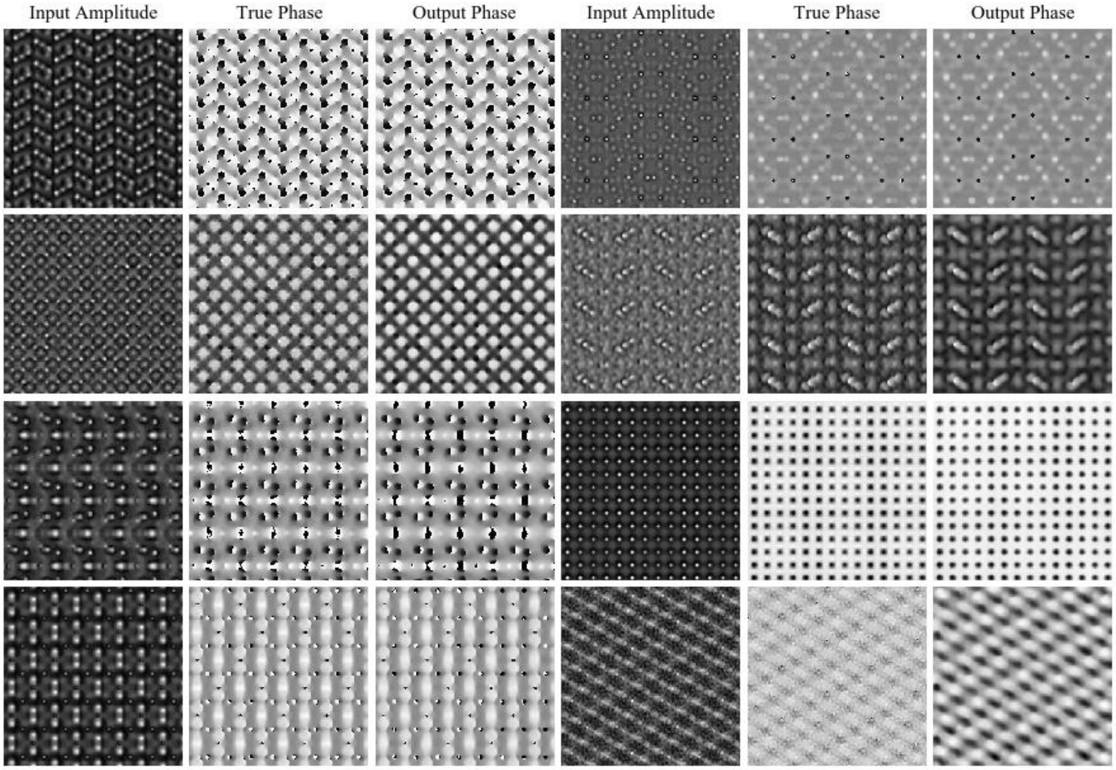


Fig. S7: Input amplitudes, target phases and output phases of 224×224 validation set wavefunctions for restricted simulation

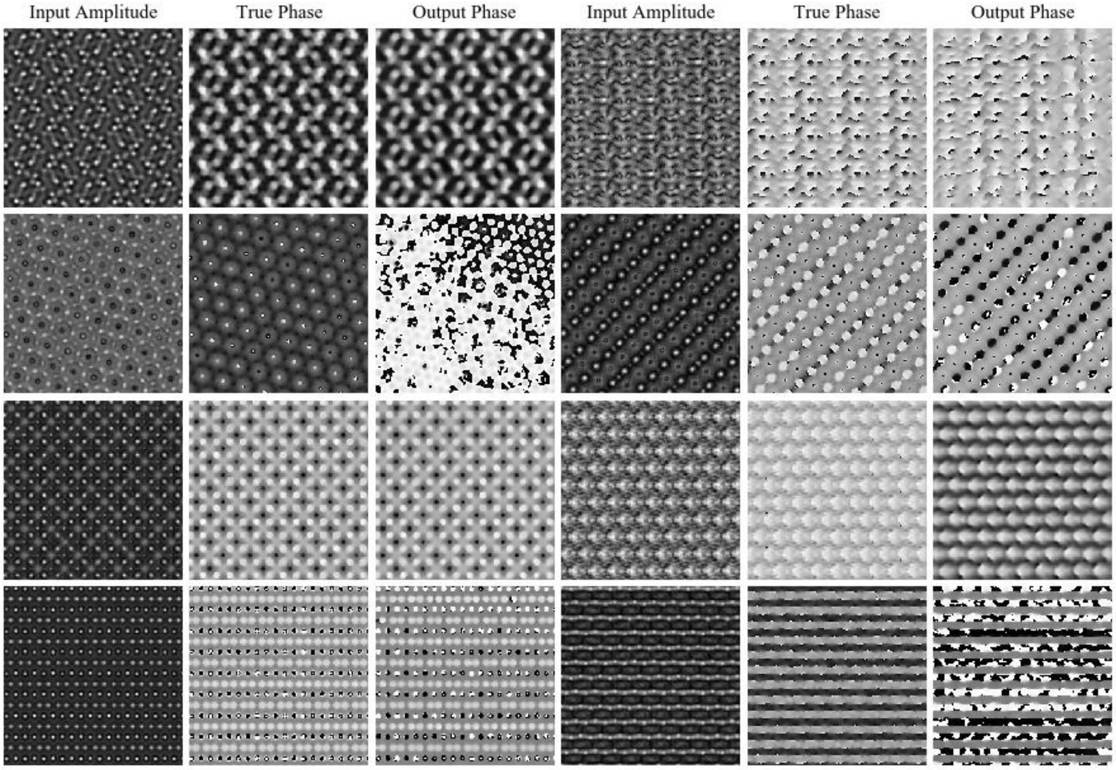


Fig. S8: Input amplitudes, target phases and output phases of 224×224 validation set wavefunctions for restricted simulation hyperparameters, and $n = 3$ simulation physics.

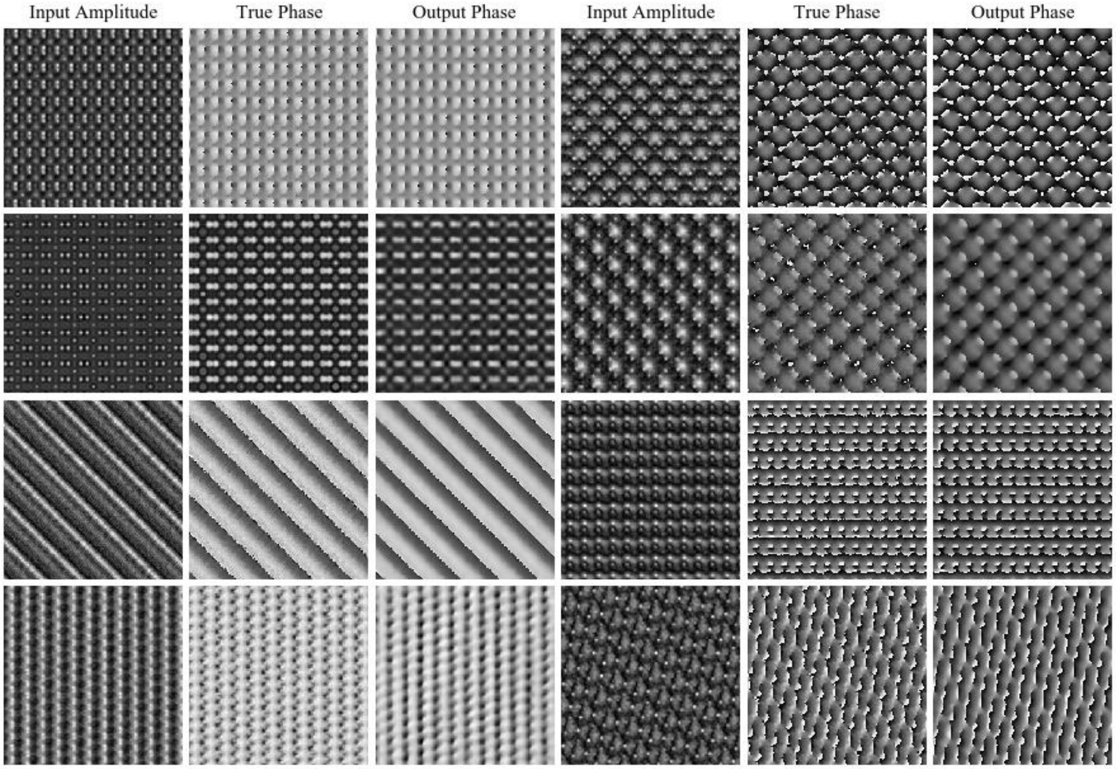


Fig. S9: Input amplitudes, target phases and output phases of 224×224 $\mathrm{In}_{1.7}\mathrm{K}_2\mathrm{Se}_8\mathrm{Sn}_{2.28}$ training set wavefunctions for unseen flips,

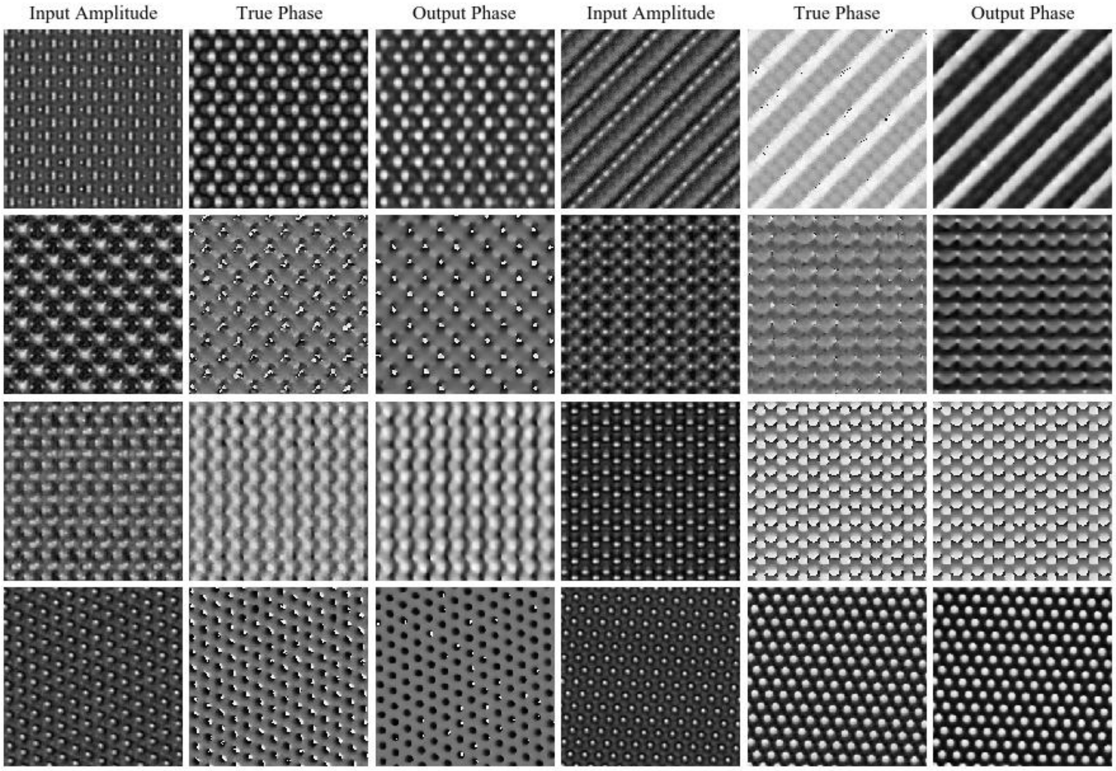


Fig. S10: Input amplitudes, target phases and output phases of 224×224 $In_{1.7}K_2Se_8Sn_{2.28}$ validation set wavefunctions for unseen simulation hyperparameters, and $n = 1$ simulation physics.

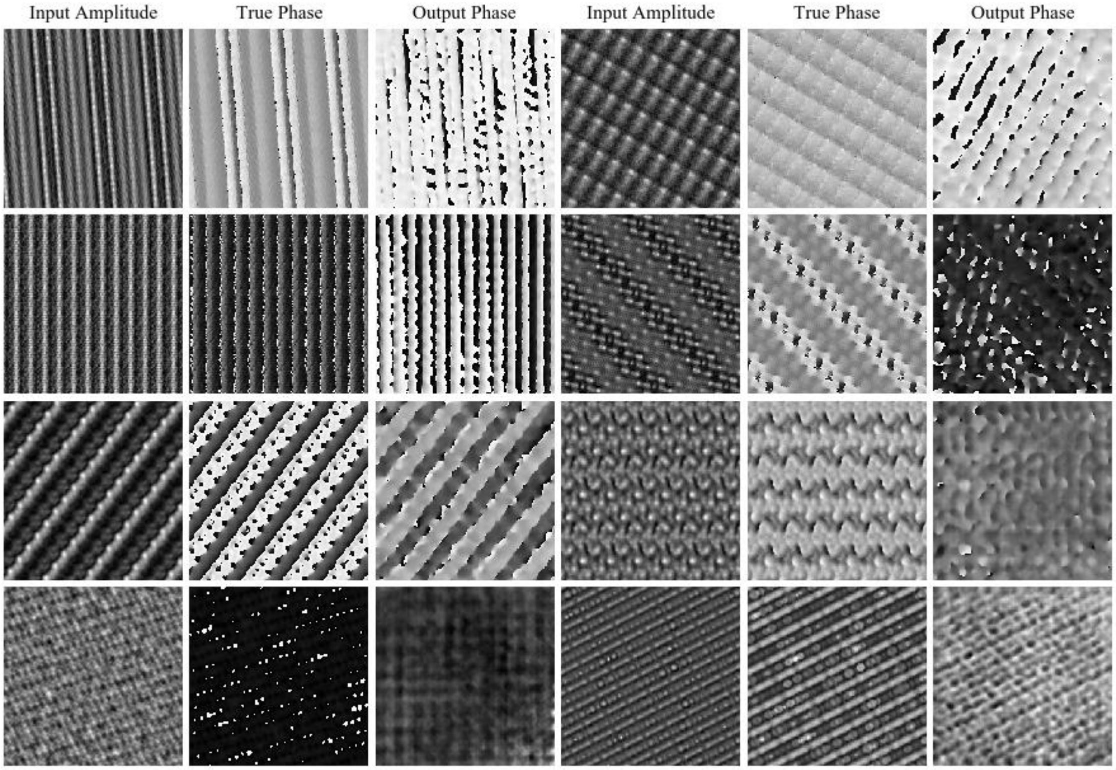


Fig. S11: Input amplitudes, target phases and output phases of 224×224 validation set wavefunctions for unseen simulation hyperparameters

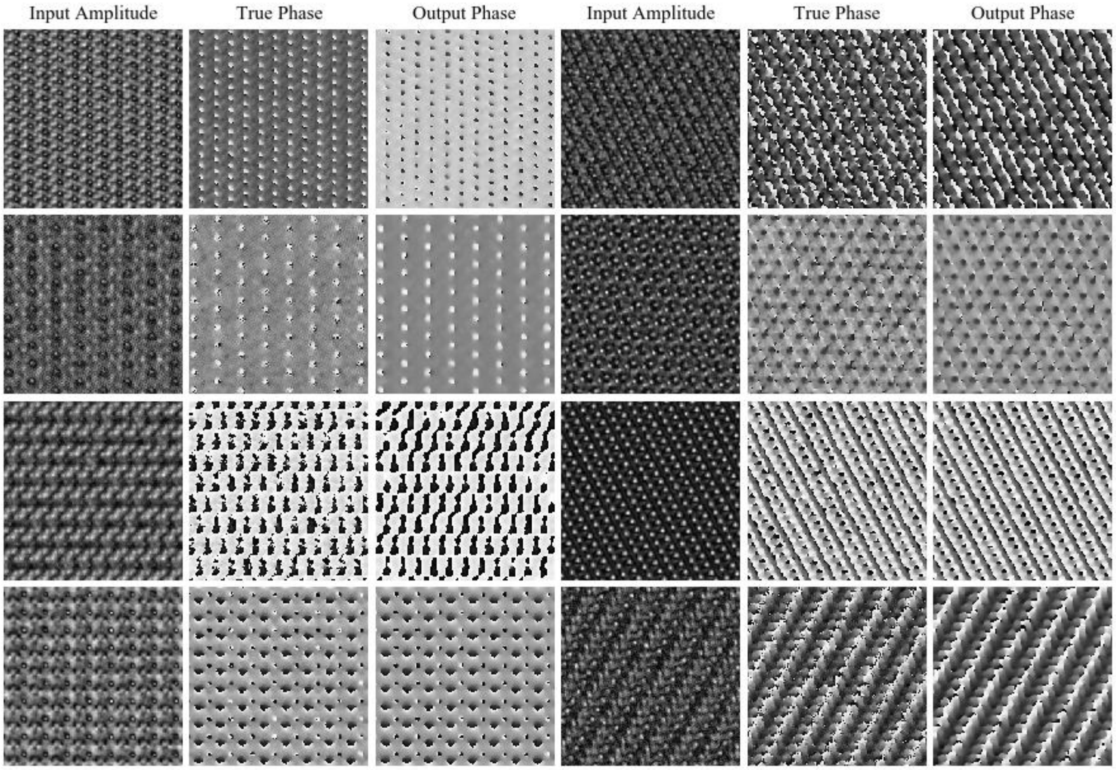


Fig. S12: Input amplitudes, target phases and output phases of 224×224 $In_{1.7}K_2Se_8Sn_{2.28}$ training set wavefunctions for unseen flips, rotations and translations, and $n = 1$ simulation physics.

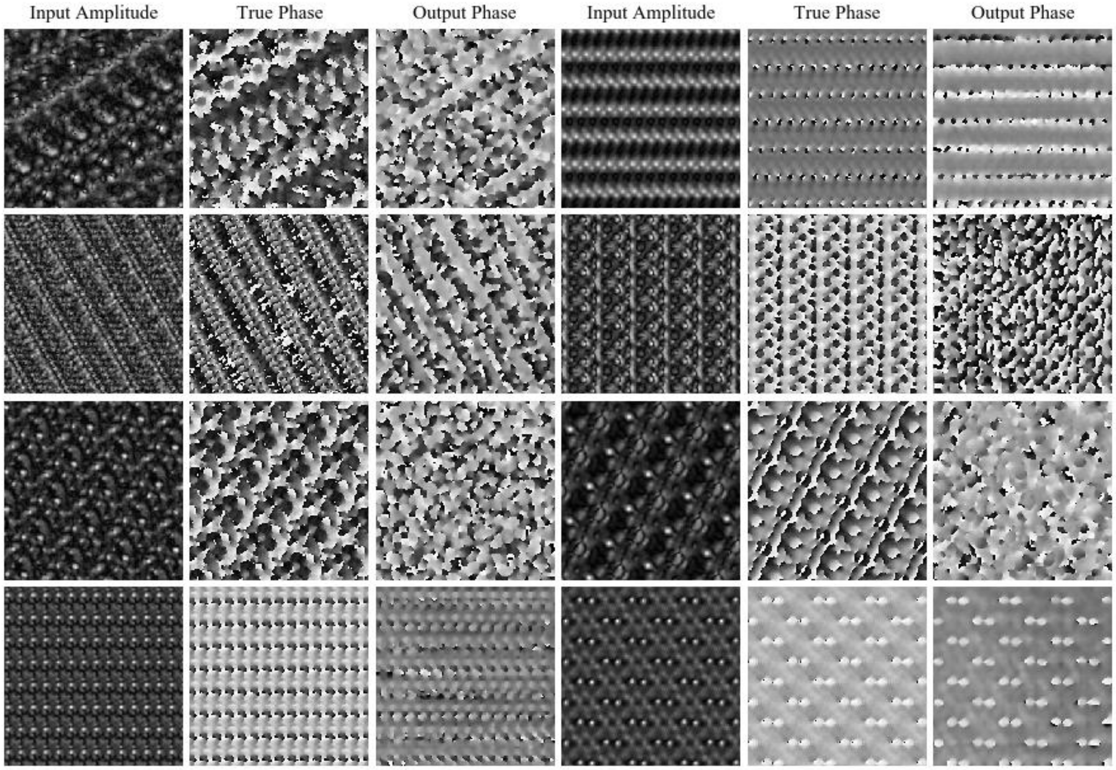


Fig. S13: Input amplitudes, target phases and output phases of 224×224 $In_{1.7}K_2Se_8Sn_{2.28}$ validation set wavefunctions for unseen

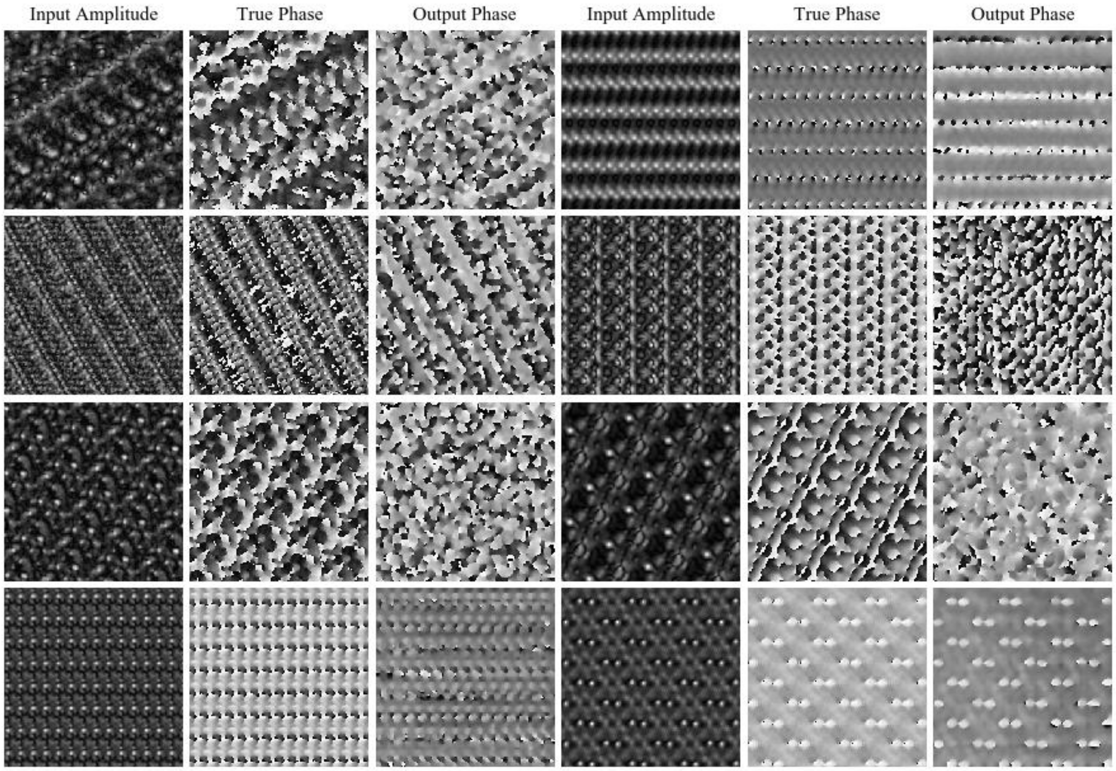


Fig. S14: Input amplitudes, target phases and output phases of 224×224 validation set wavefunctions for unseen simulation hyperparameters and materials, and $n = 3$ simulation physics. The generator was trained with $\mathrm{In}_{1.7}\mathrm{K}_2\mathrm{Se}_8\mathrm{Sn}_{2.28}$ wavefunctions.

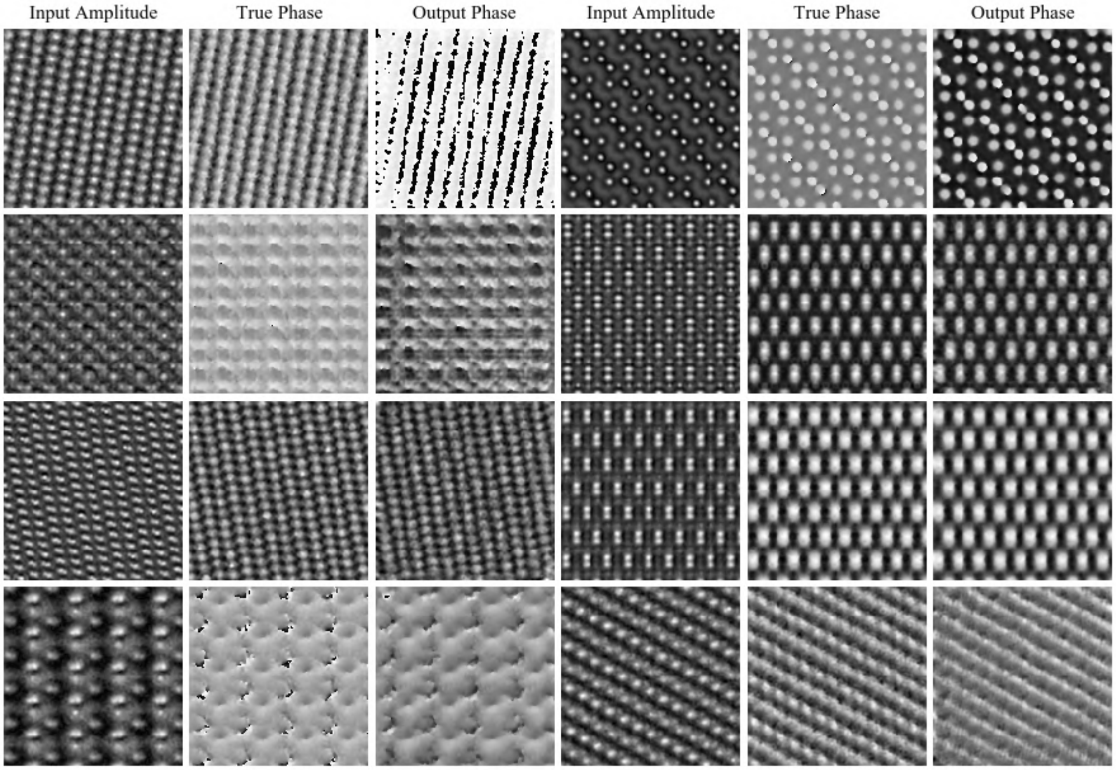


Fig. S15: GAN input amplitudes, target phases and output phases of 144×144 $\mathrm{In}_{1.7}\mathrm{K}_2\mathrm{Se}_8\mathrm{Sn}_{2.28}$ validation set wavefunctions for unseen

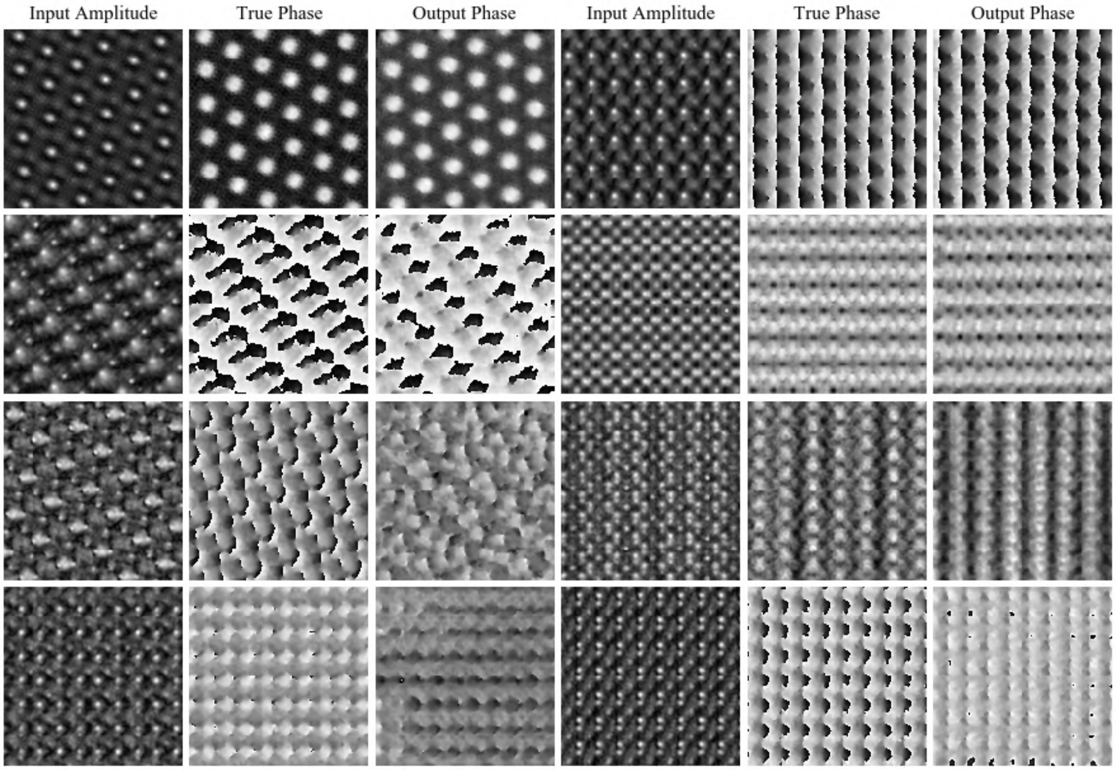


Fig. S16: GAN input amplitudes, target phases and output phases of 144×144 $In_{1.7}K_2Se_8Sn_{2.28}$ validation set wavefunctions for unseen simulation hyperparameters, and $n = 1$ simulation physics.

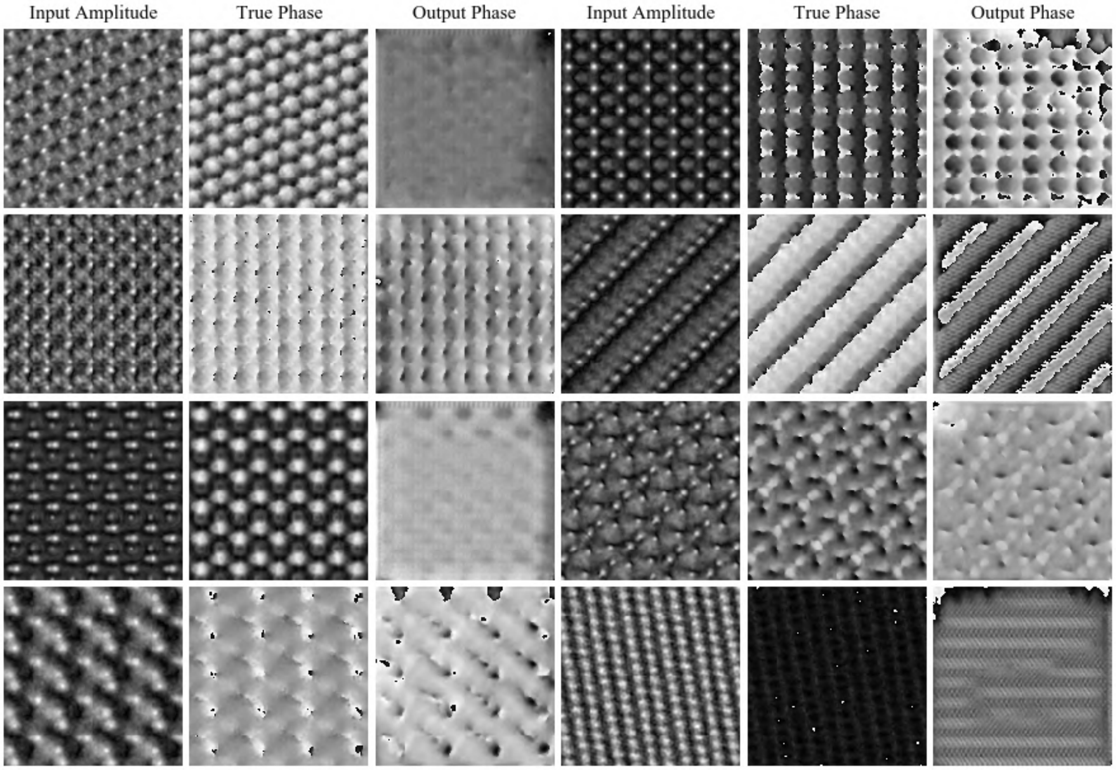


Fig. S17: GAN input amplitudes, target phases and output phases of 144×144 $In_{1.7}K_2Se_8Sn_{2.28}$ validation set wavefunctions for unseen

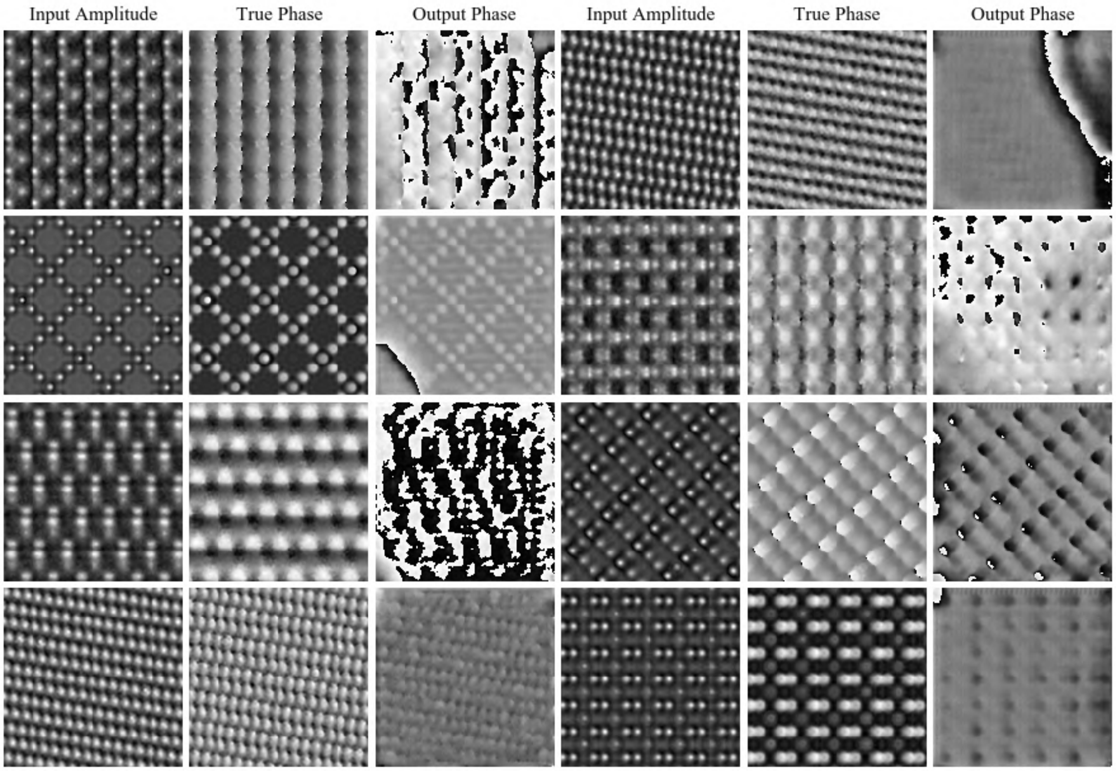


Fig. S18: GAN input amplitudes, target phases and output phases of 144×144 $\mathrm{In}_{1.7}\mathrm{K}_2\mathrm{Se}_8\mathrm{Sn}_{2.28}$ validation set wavefunctions for unseen simulation hyperparameters, and $n = 3$ simulation physics.

7.2 Reflection

This chapter covers our paper titled "Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning"[7] and associated research outputs[15,25]. At the University of Warwick, EWR is usually based on iterative focal and tilt series reconstruction (FTSR), so a previous PhD student, Mark Dyson, GPU-accelerated FTSR[252]. However, both recording a series of electron micrographs and FTSR usually take several seconds, so FTSR is unsuitable for live EWR. We have an electrostatic biprism that can be used for live in-line holography[253–255]; however, it is not used as we find that in-line holography is more difficult than FTSR. In addition, in-line holography can require expensive microscope modification if a microscope is not already equipped for it. Thus, I was inspired by applications of DNNs to predict missing information for low-light vision[256,257] to investigate live application of DNNs to predict missing phases of exit wavefunctions from single TEM images.

A couple of years ago, it was shown that DNNs can recover phases of exit wavefunctions from single optical micrographs if wavefunctions are constrained by limiting input variety[258–260]. Similarly, electron propagation can be described by wave optics[261], and optical and electron microscopes have similar arrangements of optical and electromagnetic lenses, respectively[262]. Thus, it might be expected that DNNs can recover phases of exit wavefunctions from single TEM images. However, earlier experiments with optical micrographs were unbeknownst to us when we started our investigation. Thus, whether DNNs could reconstruct phase information from single TEM images was contentions as there are infinite possible phases for a given amplitude. Further, previous non-iterative approaches to TEM EWR were limited to defocused images in the Fresnel regime[263] or non-planar incident wavefunctions in the Fraunhofer regime[264].

We were not aware of any large openly accessible datasets containing experimental TEM exit wavefunctions. Consequently, we simulated exit wavefunctions with clTEM[252,265] for a preliminary investigation. Similar to optical EWR[258–260], we found that DNNs can recover the phases of TEM exit wavefunctions if wavefunction variety is restricted. Limitingly, our simulations are unrealistic insofar they do not include aberrations, specimen drift, statistical noise, and higher-order simulation physics. However, we have demonstrated that DNNs can learn to remove noise[6] (ch. 6), specimen drifted can be reduced by sample holders[266], and aberrations can be minimized by aberration correctors[261,267–269]. Moreover, our results present lower bounds for performance as our inputs were far less restricted than possible in practice.

Curating a dataset of experimental exit wavefunctions to train DNNs to recover their phases is time-consuming and expensive. Further, data curation became impractical due to a COVID-19 national lockdown in the United Kingdom[196]. Instead, we propose a new approach to EWR that uses metadata to inform DNN training with single images. Our TEM (ch. 6) and STEM (ch. 4) images in WEMD[2] are provided as a possible resource to investigate our proposal. However, metadata is not included in WEMD, which is problematic as performance is expected to increase with increasing metadata as increasing metadata increasingly restricts probable exit wavefunctions. Nevertheless, DNNs can reconstruct some metadata from unlabelled electron micrographs[270]. Another issue is that experimental WEMD contain images for a range of electron microscope configurations, which would complicate DNN training. For example, experimental TEM images include bright field, dark field, diffraction and CBED images. However, data clustering could be applied to partially automate labelling of electron microscope configurations. For example, I provide pretrained VAEs to embed images for tSNE[2] (ch. 2).

Chapter 8

Conclusions

This thesis covers a subset of my papers on advances in electron microscopy with deep learning. My review paper (ch. 1) offers a substantial introduction that sets my work in context. Ancillary chapters then introduce new machine learning datasets for electron microscopy (ch. 2) and an algorithm to prevent learning instabilty when training large neural networks with limited computational resources (ch. 3). Finally, we report applications of deep learning to compressed sensing in STEM with static (ch. 4) and dynamic (ch. 5) scans, improving TEM signal-to-noise (ch. 6), and TEM exit wavefunction reconstruction (ch. 7). This thesis therefore presents a substantial original contribution to knowledge which is, in practice, worthy of peer-reviewed publication. This thesis adds to my existing papers by presenting their relationships, reflections, and holistic conclusions. To encourage further investigation, source code, pretrained models, datasets, and other research outputs associated with this thesis are openly accessible.

Experiments presented in this thesis are based on unlabelled electron microscopy image data. Thus, this thesis demonstrates that large machine learning datasets can be valuable without needing to add enhancements, such as image-level or pixel-level labels, to data. Indeed, this thesis can be characterized as an investigation into applications of large unlabelled electron microscopy datasets. However, I expect that tSNE clustering based on my pretrained VAE encodings[2] (ch. 2) could ease image-level labelling for future investigations. Most areas of science are facing a reproducibility crisis[115], including artificial intelligence[271], which I think is partly due to a perceived lack of value in archiving data that has not been enhanced. However, this thesis demonstrates that unlabelled data can readily enable new applications of deep learning in electron microscopy. Thus, I hope that my research will encourage more extensive data archiving by the electron microscopy community.

My DNNs were developed with TensorFlow[272,273] and Python. In addition, recent versions of Gatan Microscopy Suite (GMS) software[274], which is often used to drive electron microscopes, support Python[275]. Thus, my pretrained models and source code can be readily integrated into existing GMS software. If a microscope is operated by alternative software or an older version of GMS that does not support Python, TensorFlow supports many other programming languages[1] which can also interface with my pretrained models, and which may be more readily integrated. Alternatively, Python code can often be readily embedded in or executed by other programming languages. To be clear, my DNNs were developed as part of an initial investigation of deep learning in electron microscopy. Thus, this thesis presents lower bounds for performance that may be improved upon by refining ANN architecture and learning policy. Nevertheless, my pretrained models can be the initial basis of deep learning software for electron microscopy.

This thesis includes a variety of experiments to refine ANN architecture and learning policy. As AutoML[245–249] has improved since the start of my PhD, I expect that human involvement can be reduced in future investigations of standard architecture and learning policy variations. However, AutoML is yet to be able to routinely develop new approaches to machine learning, such as VAE encoding normalization and regularization[2] (ch. 2) and ALRC[3] (ch. 3). Most machine learning experts do not think that a technological singularity, where machines outrightly surpasses human developers, is likely for at least a couple of decades[276]. Nonetheless, our increasingly creative machines are already automating some aspects of software development[277,278] and can programmatically describe ANNs[279]. Subsequently, I encourage adoption of creative software, like AutoML, to ease development.

Perhaps the most exciting aspect of ANNs is their scalability[280,281]. Once an ANN has been trained, clones of the ANN and supporting software can be deployed on many electron microscopes at little or no additional cost to the developer. All machine learning software comes with technical debt[282,283]; however, software maintenance costs are usually far lower than the cost of electron microscopes. Thus, machine learning may be a promising means to cheaply enhance electron microscopes. As an example, my experiments indicate that compressed sensing ANNs[4] (ch. 4) can increase STEM and other electron microscopy resolution by up to 10× with minimal information loss. Such a resolution increase could greatly reduce the cost of electron microscopes while maintaining similar capability. Further, I anticipate that multiple ANNs offering a variety of functionality can be combined into a single- or multiple-ANN system that simultaneously offers a variety of enhancements, including increased resolution, decreased noise[6] (ch. 6), and phase information[7] (ch. 6).

I think the main limitation of this thesis, and deep learning, is that it is difficult to fairly compare different approaches to DNN development. As an example, I found that STEM compressed sensing with regularly spaced scans outperforms contiguous scans for the same ANN architecture and learning policy[4] (ch. 4). However, such a performance comparison is complicated by sensitivity of performance to training data, architecture, and learning policy. As a case in point, I argued that contiguous scans could outperform spiral scans if STEM images were not oversampled[4], which could be the case if partial STEM ANNs are also trained to increase image resolution. In part, I think ANN development is an art: Most ANN architecture and learning policy is guided by heuristics, and best approaches to maximize performance are chosen by natural selection[284]. Due to the complicated nature of most data, maximum performances that can be achieved with deep learning are not known. However, it follows from the universal approximator theorem[233–241] that minimum errors can, in principle, be achieved by DNNs.

Applying an ANN to a full image usually requires less computation than applying an ANN to multiple image crops. Processing full images avoids repeated calculations if crops overlap[6] (ch. 6) or lower performance near crop edges where there is less information[4,6,19] (ch. 4 and ch. 6). However, it is usually impractical to train large DNNs to process full electron microscopy images, which are often 1024×1024 or larger, due to limited memory in most GPUs. This was problematic as one of my original agreements about my research was that I would demonstrate that DNNs could be applied to large electron microscopy images, which Richard Beanland and I decided were at least 512×512. As a result, most of my DNNs were developed for 512×512 crops from electron micrographs, especially near the start of my PhD. The combination of large input images and limited available GPU memory restricted training batch sizes to few examples for large ANNs, so I often trained ANNs with a batch size of 1 and either weight[285] or spectral[286] normalization, rather than batch normalization[287].

Most of my DNNs leverage an understanding of physics to add extra information to electron microscopy images. Overt examples include predicting unknown pixels for compressed sensing with static[4] (ch. 4) or adaptive[5] (ch. 5) sparse scans, and unknown phase information from image intensities[7] (ch. 7). More subtly, improving image signal-to-noise with an DNN[6] (ch. 6) is akin to improving signal-to-noise by increasing numbers of intensity measurements. Arguably, even search engines based on VAEs[2] (ch. 2) add information to images insofar that VAE encodings can be compared to quantify semantic similarities between images. Ultimately, my DNNs add information to data that could already be understood from physical laws and observations. However, high-dimensional datasets can be difficult to utilize. Deep learning offers an effective and timely means to both understand high-dimensional data and leverage that understanding to produce results in a useable format. Thus, I both anticipate and encourage further investigation of deep learning in electron microscopy.

References

[1] J. M. Ede. Review: Deep Learning in Electron Microscopy. *arXiv preprint arXiv:2009.08328 (accepted by Machine Learning: Science and Technology – https://doi.org/10.1088/2632-2153/abd614)*, 2020.

[2] J. M. Ede. Warwick Electron Microscopy Datasets. *Machine Learning: Science and Technology*, 1(4):045003, 2020.

[3] J. M. Ede and R. Beanland. Adaptive Learning Rate Clipping Stabilizes Learning. *Machine Learning: Science and Technology*, 1:015011, 2020.

[4] J. M. Ede and R. Beanland. Partial Scanning transmission Electron Microscopy with Deep Learning. *Scientific Reports*, 10(1):1–10, 2020.

[5] J. M. Ede. Adaptive Partial Scanning Transmission Electron Microscopy with Reinforcement Learning. *arXiv preprint arXiv:2004.02786 (under review by Machine Learning: Science and Technology)*, 2020.

[6] J. M. Ede and R. Beanland. Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder. *Ultramicroscopy*, 202:18–25, 2019.

[7] J. M. Ede, J. J. P. Peters, J. Sloan, and R. Beanland. Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning. *arXiv preprint arXiv:2001.10938 (under review by Ultramicroscopy)*, 2020.

[8] J. M. Ede. Resume of Jeffrey Mark Ede. Zenodo, Online: `https://doi.org/10.5281/zenodo.4429077`, 2021.

[9] J. M. Ede. Supplementary Information: Warwick Electron Microscopy Datasets. Zenodo, Online: `https://doi.org/10.5281/zenodo.3899740`, 2020.

[10] J. M. Ede. Supplementary Information: Partial Scanning Transmission Electron Microscopy with Deep Learning. Online: `https://static-content.springer.com/esm/art%3A10.1038%2Fs41598-020-65261-0/MediaObjects/41598_2020_65261_MOESM1_ESM.pdf`, 2020.

[11] J. M. Ede. Supplementary Information: Adaptive Partial Scanning Transmission Electron Microscopy with Reinforcement Learning. Zenodo, Online: `https://doi.org/10.5281/zenodo.4384708`, 2020.

[12] J. M. Ede, J. J. P. Peters, J. Sloan, and R. Beanland. Supplementary Information: Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning. Zenodo, Online: `https://doi.org/10.5281/zenodo.4277357`, 2020.

[13] J. M. Ede. Warwick Electron Microscopy Datasets. *arXiv preprint arXiv:2003.01113*, 2020.

[14] J. M. Ede. Source Code for Warwick Electron Microscopy Datasets. Online: https://github.com/Jeffrey-Ede/datasets, 2020.

[15] J. M. Ede. Warwick Electron Microscopy Datasets Archive. Online: https://github.com/Jeffrey-Ede/datasets/wiki, 2020.

[16] J. M. Ede and R. Beanland. Adaptive Learning Rate Clipping Stabilizes Learning. *arXiv preprint arXiv:1906.09060*, 2019.

[17] J. M. Ede. Source Code for Adaptive Learning Rate Clipping Stabilizes Learning. Online: https://github.com/Jeffrey-Ede/ALRC, 2020.

[18] J. M. Ede and R. Beanland. Partial Scanning Transmission Electron Microscopy with Deep Learning. *arXiv preprint arXiv:1910.10467*, 2020.

[19] J. M. Ede. Deep Learning Supersampled Scanning Transmission Electron Microscopy. *arXiv preprint arXiv:1910.10467*, 2019.

[20] J. M. Ede. Source Code for Partial Scanning Transmission Electron Microscopy. Online: https://github.com/Jeffrey-Ede/partial-STEM, 2019.

[21] J. M. Ede. Source Code for Deep Learning Supersampled Scanning Transmission Electron Microscopy. Online: https://github.com/Jeffrey-Ede/DLSS-STEM, 2019.

[22] J. M. Ede. Source Code for Adaptive Partial Scanning Transmission Electron Microscopy with Reinforcement Learning. Online: https://github.com/Jeffrey-Ede/adaptive-scans, 2020.

[23] J. M. Ede. Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder. *arXiv preprint arXiv:1807.11234*, 2018.

[24] J. M. Ede. Source Code for Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder. Online: https://github.com/Jeffrey-Ede/Electron-Micrograph-Denoiser, 2019.

[25] J. M. Ede. Source Code for Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning. Online: https://github.com/Jeffrey-Ede/one-shot, 2019.

[26] J. M. Ede. Progress Reports of Jeffrey Mark Ede: 0.5 Year Progress Report. Zenodo, Online: https://doi.org/10.5281/zenodo.4094750, 2020.

[27] J. M. Ede. Source Code for Beanland Atlas. Online: https://github.com/Jeffrey-Ede/Beanland-Atlas, 2018.

[28] J. M. Ede. Thesis Word Counting. Zenodo, Online: https://doi.org/10.5281/zenodo.4321429, 2020.

[29] J. M. Ede. Posters and Presentations. Zenodo, Online: https://doi.org/10.5281/zenodo.4041574, 2020.

[30] J. M. Ede. Autoencoders, Kernels, and Multilayer Perceptrons for Electron Micrograph Restoration and Compression. *arXiv preprint arXiv:1808.09916*, 2018.

[31] J. M. Ede. Source Code for Autoencoders, Kernels, and Multilayer Perceptrons for Electron Micrograph Restoration and Compression. Online: https://github.com/Jeffrey-Ede/Denoising-Kernels-MLPs-Autoencoders, 2018.

[32] J. M. Ede. Source Code for Simple Webserver. Online: https://github.com/Jeffrey-Ede/simple-webserver, 2019.

[33] Guide to Examinations for Higher Degrees by Research. University of Warwick Doctoral College, Online: https://warwick.ac.uk/services/dc/pgrassessments/gtehdr, 2020.

[34] Regulation 38: Research Degrees. University of Warwick Calendar, Online: https://warwick.ac.uk/services/gov/calendar/section2/regulations/reg38pgr, 2020.

[35] Thesis Writing and Submission. University of Warwick Department of Physics, Online: https://warwick.ac.uk/fac/sci/physics/current/postgraduate/regs/thesis, 2020.

[36] A Warwick Thesis Template. University of Warwick Department of Physics, Online: https://warwick.ac.uk/fac/sci/physics/staff/academic/mhadley/wthesis, 2020.

[37] J. M. Ede. Advances in Electron Microscopy with Deep Learning. *arXiv preprint arXiv:2101.01178*, 2021.

[38] EPSRC Studentship 1917382: Application of Novel Computing and Data Analysis Methods in Electron Microscopy. UK Research and Innovation, Online: https://gtr.ukri.org/projects?ref=studentship-1917382, 2020.

[39] EPSRC Grant EP/N035437/1: ADEPT – Advanced Devices by ElectroPlaTing. EPSRC, Online: https://gow.epsrc.ukri.org/NGBOViewGrant.aspx?GrantRef=EP/N035437/1, 2016.

[40] A. J. M. Hubert, R. Römer, and R. Beanland. Structure Refinement from 'Digital' Large Angle Convergent Beam Electron Diffraction Patterns. *Ultramicroscopy*, 198:1–9, 2019.

[41] J. M. Ede. Beanland Atlas Repository. Towards Data Science, Online: https://github.com/Jeffrey-Ede/Beanland-Atlas, 2018.

[42] J. L. Hart, S. Liu, A. C. Lang, A. Hubert, A. Zukauskas, C. Canalias, R. Beanland, A. M. Rappe, M. Arredondo, and M. L. Taheri. Electron-Beam-Induced Ferroelectric Domain Behavior in the Transmission Electron Microscope: Toward Deterministic Domain Patterning. *Physical Review B*, 94(17):174104, 2016.

[43] D. Ha. Neural Network Generative Art in Javascript. Online: https://blog.otoro.net/2015/06/19/neural-network-generative-art, 2015.

[44] T. Le. Generate Abstract Random Art with A Neural Network. Medium, Online: https://medium.com/@tuanle618/generate-abstract-random-art-with-a-neural-network-ecef26f3dd5f, 2019.

[45] L. A. Gatys, A. S. Ecker, and M. Bethge. Image Style Transfer Using Convolutional Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016.

[46] L. A. Gatys, A. S. Ecker, and M. Bethge. A Neural Algorithm of Artistic Style. *Nature Communications*, 2015.

[47] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[48] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8798–8807, 2018.

[49] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[50] F. Jiang, W. Tao, S. Liu, J. Ren, X. Guo, and D. Zhao. An End-to-End Compression Framework Based on Convolutional Neural Networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(10): 3007–3018, 2017.

[51] C. Guerin. Connecting the Dots: Writing a Doctoral Thesis by Publication. In *Research Literacies and Writing Pedagogies for Masters and Doctoral Writers*, pages 31–50. Brill, 2016.

[52] S. Mason, J. E. Morris, and M. K. Merga. Institutional and Supervisory Support for the Thesis by Publication. *Australian Journal of Education*, page 0004944120929065, 2020.

[53] V. Larivière, A. Zuccala, and É. Archambault. The Declining Scientific Impact of Theses: Implications for Electronic Thesis and Dissertation Repositories and Graduate Studies. *Scientometrics*, 74(1):109–121, 2008.

[54] 2018 Global State of Peer Review. Publons, Online: https://publons.com/static/Publons-Global-State-Of-Peer-Review-2018.pdf, 2018.

[55] J. P. Tennant. The State of the Art in Peer Review. *FEMS Microbiology Letters*, 365(19), 2018.

[56] R. Walker and P. Rocha da Silva. Emerging Trends in Peer Review – A Survey. *Frontiers in Neuroscience*, 9: 169, 2015.

[57] I. Vesper. Peer Reviewers Unmasked: Largest Global Survey Reveals Trends. *Nature*, 2018.

[58] Z.-Y. Tan, N. Cai, J. Zhou, and S.-G. Zhang. On Performance of Peer Review for Academic Journals: Analysis Based on Distributed Parallel System. *IEEE Access*, 7:19024–19032, 2019.

[59] T. Ferreras-Fernández, F. García-Peñalvo, J. A. Merlo-Vega, and H. Martín-Rodero. Providing Open Access to PhD Theses: Visibility and Citation Benefits. *Program*, 2016.

[60] M. Kettler. Ways of Disseminating, Tracking Usage and Impact of Electronic Theses and Dissertations (ETDs). In *Conference on Grey Literature and Repositories*, page 37, 2016.

[61] B. M. Miller. *The Making of Knowledge-Makers in Composition: A Distant Reading of Dissertations*. PhD thesis, City University of New York, 2015.

[62] University of Warwick Physics PhD Theses. Online: `https://wrap.warwick.ac.uk/view/theses/Department_of_Physics.html`, 2020.

[63] About arXiv. Online: `https://arxiv.org/about`, 2020.

[64] P. Ginsparg. ArXiv at 20. *Nature*, 476(7359):145–147, 2011.

[65] G. Pignalberi and M. Dominici. Introduction to LATEX and to Some of its Tools. *ArsTEXnica*, page 8, 2019.

[66] M. Bransen and G. Schulpen. Pimp Your Thesis: A Minimal Introduction to LATEX. IC/TC, U.S.S. Proton, Online: `https://ussproton.nl/files/careerweeks/20180320-pimpyourthesis.pdf`, 2018.

[67] L. Lamport. *LATEX: A Document Preparation System: User's Guide and Reference Manual*. Addison-Wesley, 1994.

[68] Creative Commons Attribution 4.0 International (CC BY 4.0). Online: `https://creativecommons.org/licenses/by/4.0`, 2020.

[69] M. B. Hoy. Rise of the Rxivs: How Preprint Servers are Changing the Publishing Process. *Medical Reference Services Quarterly*, 39(1):84–89, 2020.

[70] N. K. Fry, H. Marshall, and T. Mellins-Cohen. In Praise of Preprints. *Microbial Genomics*, 5(4), 2019.

[71] E. G. Rodríguez. Preprints and Preprint Servers as Academic Communication Tools. *Revista Cubana de Información en Ciencias de la Salud*, 30(1), 2019.

[72] G. C. Banks, J. G. Field, F. L. Oswald, E. H. O'Boyle, R. S. Landis, D. E. Rupp, and S. G. Rogelberg. Answers to 18 Questions About Open Science Practices. *Journal of Business and Psychology*, 34(3):257–270, 2019.

[73] N. Fraser, F. Momeni, P. Mayr, and I. Peters. The Relationship Between bioRxiv Preprints, Citations and Altmetrics. *Quantitative Science Studies*, 1(2):618–638, 2020.

[74] Z. Wang, W. Glänzel, and Y. Chen. The Impact of Preprints in Library and Information Science: An Analysis of Citations, Usage and Social Attention Indicators. *Scientometrics*, pages 1–21, 2020.

[75] A. C. Furnival and B. Hubbard. Open Access to Scholarly Communications: Advantages, Policy and Advocacy. *Acceso Abierto a la información en las Bibliotecas Académicas de América Latina y el Caribe*, pages 101–120, 2020.

[76] D. Y. Fu and J. J. Hughey. Meta-Research: Releasing a Preprint is Associated with More Attention and Citations for the Peer-Reviewed Article. *Elife*, 8:e52646, 2019.

[77] Y. Niyazov, C. Vogel, R. Price, B. Lund, D. Judd, A. Akil, M. Mortonson, J. Schwartzman, and M. Shron. Open Access Meets Discoverability: Citations to Articles Posted to Academia.edu. *PLOS ONE*, 11(2): e0148257, 2016.

[78] M. Klein, P. Broadwell, S. E. Farb, and T. Grappone. Comparing Published Scientific Journal Articles to Their Pre-Print Versions. *International Journal on Digital Libraries*, 20(4):335–350, 2019.

[79] C. F. Carneiro, V. G. Queiroz, T. C. Moulin, C. A. Carvalho, C. B. Haas, D. Rayêe, D. E. Henshall, E. A. De-Souza, F. Espinelli, F. Z. Boos, et al. Comparing Quality of Reporting Between Preprints and Peer-Reviewed Articles in the Biomedical Literature. *BioRxiv*, page 581892, 2019.

[80] Elsevier Language Editing Services. Online: `https://webshop.elsevier.com/language-editing-services`, 2020.

[81] IOP Editing Services. Online: `https://editing.iopscience.iop.org`, 2020.

[82] Springer Nature Author Services. Online: `https://authorservices.springernature.com`, 2020.

[83] Wiley Editing Services. Online: `https://wileyeditingservices.com/en`, 2020.

[84] R. Roth. Understanding the Importance of Copyediting in Peer-Reviewed Manuscripts. *Science Editor*, 42(2): 51, 2019.

[85] ISO 32000-2:2017 Document management — Portable document format — Part 2: PDF 2.0. International Organization for Standardization, Online: `https://www.iso.org/standard/51502.html`, 2017.

[86] ISO 32000-2:2008 Document management — Portable document format — Part 1: PDF 1.7. Adobe Systems, Online: `http://wwwimages.adobe.com/www.adobe.com/content/dam/acom/en/devnet/pdf/pdfs/PDF32000_2008.pdf`, 2008.

[87] arXiv License Information. Online: `https://arxiv.org/help/license`, 2020.

[88] C. B. Clement, M. Bierbaum, K. P. O'Keeffe, and A. A. Alemi. On the Use of ArXiv as a Dataset. *arXiv preprint arXiv:1905.00075*, 2019.

[89] S. Eger, C. Li, F. Netzer, and I. Gurevych. Predicting Research Trends from ArXiv. *arXiv preprint arXiv:1903.02831*, 2019.

[90] T. Ross-Hellauer. What is Open Peer Review? A Systematic Review. *F1000Research*, 6, 2017.

[91] About OpenReview. Online: `https://openreview.net/about`, 2020.

[92] D. Soergel, A. Saunders, and A. McCallum. Open Scholarship and Peer Review: a Time for Experimentation. In *International Conference on Machine Learning (ICML 2013) Peer Review Workshop*, 2013.

[93] L. Wang and Y. Zhan. A Conceptual Peer Review Model for arXiv and Other Preprint Databases. *Learned Publishing*, 32(3):213–219, 2019.

[94] GitHub Profile of Jeffrey Mark Ede. Online: `https://github.com/Jeffrey-Ede`, 2020.

[95] Zenodo. Online: `https://about.zenodo.org`, 2020.

[96] R. Lalli. A Brief History of Physics Reviews. *Nature Reviews Physics*, 1(1):12, 2019.

[97] S. Sengupta, S. Basak, P. Saikia, S. Paul, V. Tsalavoutis, F. Atiah, V. Ravi, and A. Peters. A Review of Deep Learning with Special Emphasis on Architectures, Applications and Recent Trends. *Knowledge-Based Systems*, page 105596, 2020.

[98] A. Shrestha and A. Mahmood. Review of Deep Learning Algorithms and Architectures. *IEEE Access*, 7: 53040–53065, 2019.

[99] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. S. Awwal, and V. K. Asari. A State-of-the-Art Survey on Deep Learning Theory and Architectures. *Electronics*, 8(3):292, 2019.

[100] Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. *Nature*, 521(7553):436–444, 2015.

[101] J. Schmidhuber. Deep Learning in Neural Networks: An Overview. *Neural Networks*, 61:85–117, 2015.

[102] G. Barbastathis, A. Ozcan, and G. Situ. On the Use of Deep Learning for Computational Imaging. *Optica*, 6 (8):921–943, 2019.

[103] M. Ge, F. Su, Z. Zhao, and D. Su. Deep Learning Analysis on Microscopic Imaging in Materials Science. *Materials Today Nano*, page 100087, 2020.

[104] J. Wei, X. Chu, X.-Y. Sun, K. Xu, H.-X. Deng, J. Chen, Z. Wei, and M. Lei. Machine Learning in Materials Science. *InfoMat*, 1(3):338–358, 2019.

[105] G. R. Schleder, A. C. Padilha, C. M. Acosta, M. Costa, and A. Fazzio. From DFT to Machine Learning: Recent Approaches to Materials Science – A Review. *Journal of Physics: Materials*, 2(3):032001, 2019.

[106] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová. Machine Learning and the Physical Sciences. *Reviews of Modern Physics*, 91(4):045002, 2019.

[107] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*, 2014.

[108] D. P. Kingma and M. Welling. An Introduction to Variational Autoencoders. *arXiv preprint arXiv:1906.02691*, 2019.

[109] C. Doersch. Tutorial on Variational Autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[110] L. v. d. Maaten and G. Hinton. Visualizing Data Using t-SNE. *Journal of Machine Learning Research*, 9 (Nov):2579–2605, 2008.

[111] G. C. Linderman and S. Steinerberger. Clustering with t-SNE, Provably. *SIAM Journal on Mathematics of Data Science*, 1(2):313–332, 2019.

[112] L. van der Maaten. Accelerating t-SNE Using Tree-Based Algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

[113] L. van der Maaten. Barnes-Hut-SNE. *arXiv preprint arXiv:1301.3342*, 2013.

[114] M. Wattenberg, F. Viégas, and I. Johnson. How to Use t-SNE Effectively. *Distill*, 1(10):e2, 2016.

[115] M. Baker. Reproducibility Crisis. *Nature*, 533(26):353–66, 2016.

[116] C. A. of Cyberspace Studies. Development of World Internet. *World Internet Development Report 2017: Translated by Peng Ping*, pages 1–19, 2019.

[117] T. Berners-Lee, L. Masinter, and M. McCahill. RFC1738: Uniform Resource Locators (URL). *RFC*, 1994. doi: 10.17487/RFC1738.

[118] P. Kaushik, D. P. Singh, and S. Rajpoot. Fibre Optic Communication in 21 st Century. In *2020 International Conference on Intelligent Engineering and Management (ICIEM)*, pages 125–129. IEEE, 2020.

[119] E. Mack. The History of Broadband. *Geographies of the Internet*, pages 63–76, 2020.

[120] L. Abrardi and C. Cambini. Ultra-Fast Broadband Investment and Adoption: A Survey. *Telecommunications Policy*, 43(3):183–198, 2019.

[121] M. Graydon and L. Parks. ’Connecting the Unconnected’: A Critical Assessment of US Satellite Internet Services. *Media, Culture & Society*, 42(2):260–276, 2020.

[122] C. Kaufmanna, H.-P. Huthb, F. Zeigerb, and M. Schmidta. Performance Evaluation of Internet over Geostationary Satellite for Industrial Applications. In *70th International Astronautical Congress*. International Astronautical Federation, 2019.

[123] D. Castelvecchi. Google Unveils Search Engine for Open Data. *Nature*, 561(7722):161–163, 2018.

[124] N. Noy. Discovering Millions of Datasets on the Web. The Keyword, Online: `https://blog.google/products/search/discovering-millions-datasets-web`, 2020.

[125] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 843–852, 2017.

[126] T. Hey, K. Butler, S. Jackson, and J. Thiyagalingam. Machine Learning and Big Scientific Data. *Philosophical Transactions of the Royal Society A*, 378(2166):20190054, 2020.

[127] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The Kinetics Human Action Video Dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[128] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[129] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. YouTube-8M: A Large-Scale Video Classification Benchmark. *arXiv preprint arXiv:1609.08675*, 2016.

[130] C. Sohrabi, Z. Alsafi, N. O'Neill, M. Khan, A. Kerwan, A. Al-Jabir, C. Iosifidis, and R. Agha. World Health Organization Declares Global Emergency: A Review of the 2019 Novel Coronavirus (COVID-19). *International Journal of Surgery*, 2020.

[131] W. J. Wiersinga, A. Rhodes, A. C. Cheng, S. J. Peacock, and H. C. Prescott. Pathophysiology, Transmission, Diagnosis, and Treatment of Coronavirus Disease 2019 (COVID-19): A Review. *Jama*, 324(8):782–793, 2020.

[132] T. Singhal. A Review of Coronavirus Disease-2019 (COVID-19). *The Indian Journal of Pediatrics*, pages 1–6, 2020.

[133] L. O. Teixeira, R. M. Pereira, D. Bertolini, L. S. Oliveira, L. Nanni, and Y. M. Costa. Impact of Lung Segmentation on the Diagnosis and Explanation of COVID-19 in Chest X-Ray Images. *arXiv preprint arXiv:2009.09780*, 2020.

[134] A. Ghosal, A. Nandy, A. K. Das, S. Goswami, and M. Panday. A Short Review on Different Clustering Techniques and Their Applications. In *Emerging Technology in Modelling and Graphics*, pages 69–83. Springer, 2020.

[135] M. Z. Rodriguez, C. H. Comin, D. Casanova, O. M. Bruno, D. R. Amancio, L. d. F. Costa, and F. A. Rodrigues. Clustering Algorithms: A Comparative Approach. *PlOS ONE*, 14(1):e0210236, 2019.

[136] K. Djouzi and K. Beghdad-Bey. A Review of Clustering Algorithms for Big Data. In *2019 International Conference on Networking and Advanced Systems (ICNAS)*, pages 1–6. IEEE, 2019.

[137] M. Mittal, L. M. Goyal, D. J. Hemanth, and J. K. Sethi. Clustering Approaches for High-Dimensional Databases: A Review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(3):e1300, 2019.

[138] M. Y. Ansari, A. Ahmad, S. S. Khan, G. Bhushan, et al. Spatiotemporal Clustering: A Review. *Artificial Intelligence Review*, pages 1–43, 2019.

[139] A. K. Jain, M. N. Murty, and P. J. Flynn. Data Clustering: A Review. *ACM Computing Surveys (CSUR)*, 31 (3):264–323, 1999.

[140] J. A. Lee and M. Verleysen. *Nonlinear Dimensionality Reduction.* Springer Science & Business Media, 2007.

[141] G. E. Hinton and S. T. Roweis. Stochastic Neighbor Embedding. In *Advances in Neural Information Processing Systems*, pages 857–864, 2003.

[142] W. Li, J. E. Cerise, Y. Yang, and H. Han. Application of t-SNE to Human Genetic Data. *Journal of Bioinformatics and Computational Biology*, 15(04):1750017, 2017.

[143] I. Wallach and R. Lilien. The Protein–Small-Molecule Database, A Non-Redundant Structural Resource for the Analysis of Protein-Ligand Binding. *Bioinformatics*, 25(5):615–620, 2009.

[144] B. M. Devassy and S. George. Dimensionality Reduction and Visualisation of Hyperspectral Ink Data Using t-SNE. *Forensic Science International*, page 110194, 2020.

[145] B. Melit Devassy, S. George, and P. Nussbaum. Unsupervised Clustering of Hyperspectral Paper Data Using t-SNE. *Journal of Imaging*, 6(5):29, 2020.

[146] P. Gang, W. Zhen, W. Zeng, Y. Gordienko, Y. Kochura, O. Alienin, O. Rokovyi, and S. Stirenko. Dimensionality Reduction in Deep Learning for Chest X-Ray Analysis of Lung Cancer. In *2018 Tenth International Conference on Advanced Computational Intelligence (ICACI)*, pages 878–883. IEEE, 2018.

[147] J. Birjandtalab, M. B. Pouyan, and M. Nourani. Nonlinear Dimension Reduction for EEG-Based Epileptic Seizure Detection. In *2016 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, pages 595–598. IEEE, 2016.

[148] W. M. Abdelmoula, B. Balluff, S. Englert, J. Dijkstra, M. J. Reinders, A. Walch, L. A. McDonnell, and B. P. Lelieveldt. Data-Driven Identification of Prognostic Tumor Subpopulations Using Spatially Mapped t-SNE of Mass Spectrometry Imaging Data. *Proceedings of the National Academy of Sciences*, 113(43):12244–12249, 2016.

[149] F. Psihas, E. Niner, M. Groh, R. Murphy, A. Aurisano, A. Himmel, K. Lang, M. D. Messier, A. Radovic, and A. Sousa. Context-Enriched Identification of Particles with a Convolutional Network for Neutrino Events. *Physical Review D*, 100(7):073005, 2019.

[150] E. Racah, S. Ko, P. Sadowski, W. Bhimji, C. Tull, S.-Y. Oh, P. Baldi, et al. Revealing Fundamental Physics from the Daya Bay Neutrino Experiment Using Deep Neural Networks. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 892–897. IEEE, 2016.

[151] F. Gong, F. Bu, Y. Zhang, Y. Yan, R. Hu, and M. Dong. Visual Clustering Analysis of Electricity Data Based on t-SNE. In *2020 IEEE 5th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*, pages 234–240. IEEE, 2020.

[152] K. McDonald, M. Tan, and Y. Mann. The Infinite Drum Machine. Experiments with Google, Online: https://experiments.withgoogle.com/drum-machine, 2018.

[153] E. Schubert and M. Gertz. Intrinsic t-Stochastic Neighbor Embedding for Visualization and Outlier Detection. In *International Conference on Similarity Search and Applications*, pages 188–203. Springer, 2017.

[154] I. T. Jolliffe and J. Cadima. Principal Component Analysis: A Review and Recent Developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065): 20150202, 2016.

[155] M. E. Wall, A. Rechtsteiner, and L. M. Rocha. Singular Value Decomposition and Principal Component Analysis. In *A Practical Approach to Microarray Data Analysis*, pages 91–109. Springer, 2003.

[156] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Review*, 53(2):217–288, 2011.

[157] P.-G. Martinsson, V. Rokhlin, and M. Tygert. A Randomized Algorithm for the Decomposition of Matrices. *Applied and Computational Harmonic Analysis*, 30:47–68, 2011.

[158] N. Pezzotti, J. Thijssen, A. Mordvintsev, T. Höllt, B. Van Lew, B. P. Lelieveldt, E. Eisemann, and A. Vilanova. GPGPU Linear Complexity t-SNE Optimization. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1172–1181, 2019.

[159] D. M. Chan, R. Rao, F. Huang, and J. F. Canny. t-SNE-CUDA: GPU-Accelerated t-SNE and its Applications to Modern Data. In *2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pages 330–338. IEEE, 2018.

[160] N. Pezzotti, B. P. Lelieveldt, L. van der Maaten, T. Höllt, E. Eisemann, and A. Vilanova. Approximated and User Steerable tSNE for Progressive Visual Analytics. *IEEE Transactions on Visualization and Computer Graphics*, 23(7):1739–1752, 2016.

[161] A. C. Belkina, C. O. Ciccolella, R. Anno, R. Halpert, J. Spidlen, and J. E. Snyder-Cappione. Automated Optimized Parameters for t-Distributed Stochastic Neighbor Embedding Improve Visualization and Analysis of Large Datasets. *Nature Communications*, 10(1):1–12, 2019.

[162] P. Alfeld. A Trivariate Clough—Tocher Scheme for Tetrahedral Data. *Computer Aided Geometric Design*, 1 (2):169–181, 1984.

[163] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. *arXiv preprint arXiv:1711.02257*, 2017.

[164] I. Malkiel, S. Ahn, V. Taviani, A. Menini, L. Wolf, and C. J. Hardy. Conditional WGANs with Adaptive Gradient Balancing for Sparse MRI Reconstruction. *arXiv preprint arXiv:1905.00985*, 2019.

[165] W. McIlhagga. Estimates of Edge Detection Filters in Human Vision. *Vision Research*, 153:30–36, 2018.

[166] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding Beyond Pixels Using a Learned Similarity Metric. In *International Conference on Machine Learning*, pages 1558–1566, 2016.

[167] G. Grund Pihlgren, F. Sandin, and M. Liwicki. Improving Image Autoencoder Embeddings with Perceptual Loss. In *International Joint Conference on Neural Networks*, 2020.

[168] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[169] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. Loss Functions for Image Restoration with Neural Networks. *IEEE Transactions on Computational Imaging*, 3(1):47–57, 2016.

[170] S. Z. Dadaneh, S. Boluki, M. Yin, M. Zhou, and X. Qian. Pairwise Supervised Hashing with Bernoulli Variational Auto-Encoder and Self-Control Gradient Estimator. *arXiv preprint arXiv:2005.10477*, 2020.

[171] N. Patterson and Y. Wang. Semantic Hashing with Variational Autoencoders, 2016.

[172] G. Jin, Y. Zhang, and K. Lu. Deep Hashing Based on VAE-GAN for Efficient Similarity Retrieval. *Chinese Journal of Electronics*, 28(6):1191–1197, 2019.

[173] F. Mena and R. Ñanculef. A Binary Variational Autoencoder for Hashing. In *Iberoamerican Congress on Pattern Recognition*, pages 131–141. Springer, 2019.

[174] D. Shen, Q. Su, P. Chapfuwa, W. Wang, G. Wang, L. Carin, and R. Henao. Nash: Toward End-to-End Neural Architecture for Generative Semantic Hashing. *arXiv preprint arXiv:1805.05361*, 2018.

[175] S. Chaidaroon and Y. Fang. Variational Deep Semantic Hashing for Text Documents. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 75–84, 2017.

[176] K. J. Liang, C. Li, G. Wang, and L. Carin. Generative Adversarial Network Training is a Continual Learning Problem. *arXiv preprint arXiv:1811.11083*, 2018.

[177] M. Li, M. Soltanolkotabi, and S. Oymak. Gradient Descent with Early Stopping is Provably Robust to Label Noise for Overparameterized Neural Networks. In *International Conference on Artificial Intelligence and Statistics*, pages 4313–4324, 2020.

[178] T. Flynn, K. M. Yu, A. Malik, N. D'Imperio, and S. Yoo. Bounding the Expected Run-Time of Nonconvex Optimization with Early Stopping. *arXiv preprint arXiv:2002.08856*, 2020.

[179] D. Hendrycks, K. Lee, and M. Mazeika. Using Pre-Training can Improve Model Robustness and Uncertainty. *arXiv preprint arXiv:1901.09960*, 2019.

[180] R. Beanland, K. Evans, and R. A. Roemer. Felix. Online: `https://github.com/RudoRoemer/Felix`, 2020.

[181] G. P. Meyer. An Alternative Probabilistic Interpretation of the Huber Loss. *arXiv preprint arXiv:1911.02088*, 2019.

[182] P. J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, pages 73–101, 1964.

[183] P. Seetharaman, G. Wichern, B. Pardo, and J. L. Roux. AutoClip: Adaptive Gradient Clipping for Source Separation Networks. *arXiv preprint arXiv:2007.14469*, 2020.

[184] R. Pascanu, T. Mikolov, and Y. Bengio. On the Difficulty of Training Recurrent Neural Networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[185] E. Gorbunov, M. Danilova, and A. Gasnikov. Stochastic Optimization with Heavy-Tailed Noise via Accelerated Gradient Clipping. *arXiv preprint arXiv:2005.10785*, 2020.

[186] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han. On the Variance of the Adaptive Learning Rate and Beyond. *arXiv preprint arXiv:1908.03265*, 2019.

[187] D. P. Kingma and J. Ba. ADAM: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[188] J. M. Ede. Pixel Subset Super-Compression with a Generative Adversarial Network (Unfinished Manuscript). Zenodo, Online: `https://doi.org/10.5281/zenodo.4072946`, 2020.

[189] B. Connolly. Atomic Scale Deep Learning. Towards Data Science, Online: `https://towardsdatascience.com/atomic-scale-deep-learning-34238feda632`, 2020.

[190] J. M. Ede. Pixel Subset Super-Compression of STEM Images. Online: `https://zenodo.org/record/4072946#.X37gMWhKiCo`, 2020.

[191] X. Sang, A. R. Lupini, R. R. Unocic, M. Chi, A. Y. Borisevich, S. V. Kalinin, E. Endeve, R. K. Archibald, and S. Jesse. Dynamic Scan Control in STEM: Spiral Scans. *Advanced Structural and Chemical Imaging*, 2(1): 1–8, 2016.

[192] X. Sang, A. R. Lupini, J. Ding, S. V. Kalinin, S. Jesse, and R. R. Unocic. Precision Controlled Atomic Resolution Scanning Transmission Electron Microscopy Using Spiral Scan Pathways. *Scientific Reports*, 7: 43585, 2017.

[193] S. Gandhare and B. Karthikeyan. Survey on FPGA Architecture and Recent Applications. In *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, pages 1–4. IEEE, 2019.

[194] C. Zhang, B. Berkels, B. Wirth, and P. M. Voyles. Joint Denoising and Distortion Correction for Atomic Column Detection in Scanning Transmission Electron Microscopy Images. *Microscopy and Microanalysis*, 23(S1):164–165, 2017.

[195] P. Jin and X. Li. Correction of Image Drift and Distortion in a Scanning Electron Microscopy. *Journal of Microscopy*, 260(3):268–280, 2015.

[196] E. Aspinall. COVID-19 Timeline. British Foreign Policy Group, Online: `https://bfpg.co.uk/2020/04/covid-19-timeline`, 2020.

[197] J. Caldeira and B. Nord. Deeply Uncertain: Comparing Methods of Uncertainty Quantification in Deep Learning Algorithms. *arXiv preprint arXiv:2004.10710*, 2020.

[198] A. M. Alaa. Uncertainty Quantification in Deep Learning: Literature Survey. Towards Data Science, Online: `https://github.com/ahmedmalaa/deep-learning-uncertainty`, 2020.

[199] N. Ståhl, G. Falkman, A. Karlsson, and G. Mathiason. Evaluation of Uncertainty Quantification in Deep Learning. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 556–568. Springer, 2020.

[200] A. Loquercio, M. Segu, and D. Scaramuzza. A General Framework for Uncertainty Estimation in Deep Learning. *IEEE Robotics and Automation Letters*, 5(2):3153–3160, 2020.

[201] A. G. Kendall. *Geometry and Uncertainty in Deep Learning for Computer Vision*. PhD thesis, University of Cambridge, 2019.

[202] Y. Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[203] C. Rudin. Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and use Interpretable Models Instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[204] J. van Amersfoort, L. Smith, Y. W. Teh, and Y. Gal. Simple and Scalable Epistemic Uncertainty Estimation Using a Single Deep Deterministic Neural Network. *arXiv preprint arXiv:2003.02037*, 2020.

[205] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

[206] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson. A Simple Baseline for Bayesian Uncertainty in Deep Learning. In *Advances in Neural Information Processing Systems*, pages 13153–13164, 2019.

[207] M. Teye, H. Azizpour, and K. Smith. Bayesian Uncertainty Estimation for Batch Normalized Deep Networks. *arXiv preprint arXiv:1802.06455*, 2018.

[208] A. Kendall and Y. Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In *Advances in Neural Information Processing Systems*, pages 5574–5584, 2017.

[209] Y. Gal and Z. Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

[210] L. Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, 1996.

[211] C. N. d. Santos, Y. Mroueh, I. Padhi, and P. Dognin. Learning Implicit Generative Models by Matching Perceptual Features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4461–4470, 2019.

[212] N. Saldi, S. Yüksel, and T. Linder. Asymptotic Optimality of Finite Model Approximations for Partially Observed Markov Decision Processes With Discounted Cost. *IEEE Transactions on Automatic Control*, 65 (1):130–142, 2019.

[213] T. Jaakkola, S. P. Singh, and M. I. Jordan. Reinforcement Learning Algorithm for Partially Observable Markov Decision Problems. In *Advances in Neural Information Processing Systems*, pages 345–352, 1995.

[214] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver. Memory-Based Control with Recurrent Neural Networks. *arXiv preprint arXiv:1512.04455*, 2015.

[215] B. D. Earp. The Need for Reporting Negative Results - A 90 Year Update. *Journal of clinical and translational research*, 3(Suppl 2):344, 2018.

[216] A. Mlinarić, M. Horvat, and V. Šupak Smolčić. Dealing with the Positive Publication Bias: Why You Should Really Publish Your Negative Results. *Biochemia Medica*, 27(3):447–452, 2017.

[217] S. B. Nissen, T. Magidson, K. Gross, and C. T. Bergstrom. Publication Bias and the Canonization of False Facts. *eLife*, 5:e21451, 2016.

[218] I. Andrews and M. Kasy. Identification of and Correction for Publication Bias. *American Economic Review*, 109(8):2766–94, 2019.

[219] I. Buvat and F. Orlhac. The Dark Side of Radiomics: On the Paramount Importance of Publishing Negative Results. *Journal of Nuclear Medicine*, 60(11):1543–1544, 2019.

[220] H. Sharma and S. Verma. Is Positive Publication Bias Really a Bias, or an Intentionally Created Discrimination Toward Negative Results? *Saudi Journal of Anaesthesia*, 13(4):352, 2019.

[221] N. Matosin, E. Frank, M. Engel, J. S. Lum, and K. A. Newell. Negativity Towards Negative Results: A Discussion of the Disconnect Between Scientific Worth and Scientific Culture, 2014.

[222] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is All You Need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[223] J. Alammar. The Illustrated Transformer. GitHub Blog, Online: `http://jalammar.github.io/illustrated-transformer`, 2018.

[224] S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, R. Yamamoto, X. Wang, et al. A Comparative Study on Transformer vs RNN in Speech Applications. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 449–456. IEEE, 2019.

[225] A. Zeyer, P. Bahar, K. Irie, R. Schlüter, and H. Ney. A Comparison of Transformer and LSTM Encoder Decoder Models for ASR. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 8–15. IEEE, 2019.

[226] B. Goyal, A. Dogra, S. Agrawal, B. Sohi, and A. Sharma. Image Denoising Review: From Classical to State-of-the-Art Approaches. *Information Fusion*, 55:220–244, 2020.

[227] A. Girdher, B. Goyal, A. Dogra, A. Dhindsa, and S. Agrawal. Image Denoising: Issues and Challenges. *Available at SSRN 3446627*, 2019.

[228] L. Fan, F. Zhang, H. Fan, and C. Zhang. Brief Review of Image Denoising Techniques. *Visual Computing for Industry, Biomedicine, and Art*, 2(1):7, 2019.

[229] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007.

[230] M. Lebrun. An Analysis and Implementation of the BM3D Image Denoising Method. *Image Processing On Line*, 2:175–213, 2012.

[231] F. Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1251–1258, 2017.

[232] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[233] P. Kidger and T. Lyons. Universal Approximation with Deep Narrow Networks. *arXiv preprint arXiv:1905.08539*, 2019.

[234] H. Lin and S. Jegelka. ResNet with One-Neuron Hidden Layers is a Universal Approximator. In *Advances in Neural Information Processing Systems*, pages 6169–6178, 2018.

[235] B. Hanin and M. Sellke. Approximating Continuous Functions by ReLU Nets of Minimal Width. *arXiv preprint arXiv:1710.11278*, 2017.

[236] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang. The Expressive Power of Neural Networks: A View from the Width. In *Advances in Neural Information Processing Systems*, pages 6231–6239, 2017.

[237] A. Pinkus. Approximation Theory of the MLP Model in Neural Networks. *Acta Numerica*, 8(1):143–195, 1999.

[238] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer Feedforward Networks with a Nonpolynomial Activation Function can Approximate any Function. *Neural Networks*, 6(6):861–867, 1993.

[239] K. Hornik. Approximation Capabilities of Multilayer Feedforward Networks. *Neural Networks*, 4(2):251–257, 1991.

[240] K. Hornik, M. Stinchcombe, and H. White. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, 2(5):359–366, 1989.

[241] G. Cybenko. Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[242] A. Krizhevsky. Convolutional Deep Belief Networks on CIFAR-10. *Technical Report*, 40(7):1–9, 2010.

[243] V. Nair and G. E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[244] X. Glorot, A. Bordes, and Y. Bengio. Deep Sparse Rectifier Neural Networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[245] X. He, K. Zhao, and X. Chu. AutoML: A Survey of the State-of-the-Art. *arXiv preprint arXiv:1908.00709*, 2019.

[246] E. Malekhosseini, M. Hajabdollahi, N. Karimi, and S. Samavi. Modeling Neural Architecture Search Methods for Deep Networks. *arXiv preprint arXiv:1912.13183*, 2019.

[247] Y. Jaafra, J. L. Laurent, A. Deruyver, and M. S. Naceur. Reinforcement Learning for Neural Architecture Search: A Review. *Image and Vision Computing*, 89:57–66, 2019.

[248] T. Elsken, J. H. Metzen, and F. Hutter. Neural Architecture Search: A Survey. *arXiv preprint arXiv:1808.05377*, 2018.

[249] J. Waring, C. Lindvall, and R. Umeton. Automated Machine Learning: Review of the State-of-the-Art and Opportunities for Healthcare. *Artificial Intelligence in Medicine*, page 101822, 2020.

[250] M. Hanussek, M. Blohm, and M. Kintz. Can AutoML Outperform Humans? An Evaluation on Popular OpenML Datasets Using AutoML Benchmark. *arXiv preprint arXiv:2009.01564*, 2020.

[251] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning Transferable Architectures for Scalable Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2018.

[252] M. A. Dyson. *Advances in Computational Methods for Transmission Electron Microscopy Simulation and Image Processing*. PhD thesis, University of Warwick, 2014.

[253] M. Lehmann and H. Lichte. Tutorial on Off-Axis Electron Holography. *Microscopy and Microanalysis*, 8(6): 447–466, 2002.

[254] C. T. Koch and A. Lubk. Off-Axis and Inline Electron Holography: A Quantitative Comparison. *Ultramicroscopy*, 110(5):460–471, 2010.

[255] C. Ozsoy-Keskinbora, C. B. Boothroyd, R. Dunin-Borkowski, P. A. Van Aken, and C. T. Koch. Hybridization Approach to In-Line and Off-Axis (Electron) Holography for Superior Resolution and Phase Sensitivity. *Scientific Reports*, 4:7020, 2014.

[256] F. Almasri and O. Debeir. Robust Perceptual Night Vision in Thermal Colorization. *arXiv preprint arXiv:2003.02204*, 2020.

[257] C. Chen, Q. Chen, J. Xu, and V. Koltun. Learning to See in the Dark. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3291–3300, 2018.

[258] Y. Rivenson, Y. Zhang, H. Günaydın, D. Teng, and A. Ozcan. Phase Recovery and Holographic Image Reconstruction Using Deep Learning in Neural Networks. *Light: Science & Applications*, 7(2):17141–17141, 2018.

[259] Y. Wu, Y. Rivenson, Y. Zhang, Z. Wei, H. Günaydin, X. Lin, and A. Ozcan. Extended Depth-of-Field in Holographic Imaging Using Deep-Learning-Based AutofocUsing and Phase Recovery. *Optica*, 5(6):704–710, 2018.

[260] A. Sinha, J. Lee, S. Li, and G. Barbastathis. Lensless Computational Imaging Through Deep Learning. *Optica*, 4(9):1117–1125, 2017.

[261] H. H. Rose. Optics of High-Performance Electron Microscopes. *Science and Technology of Advanced Materials*, 9(1):014107, 2008.

[262] X. Chen, B. Zheng, and H. Liu. Optical and Digital Microscopic Imaging Techniques and Applications in Pathology. *Analytical Cellular Pathology*, 34(1, 2):5–18, 2011.

[263] A. J. Morgan, A. V. Martin, A. J. D'Alfonso, C. T. Putkunz, and L. J. Allen. Direct Exit-Wave Reconstruction from a Single Defocused Image. *Ultramicroscopy*, 111(9-10):1455–1460, 2011.

[264] A. V. Martin and L. J. Allen. Direct Retrieval of a Complex Wave from its Diffraction Pattern. *Optics Communications*, 281(20):5114–5121, 2008.

[265] J. J. P. Peters and M. A. Dyson. clTEM. Online: `https://github.com/JJPPeters/clTEM`, 2019.

[266] B. H. Goodge, E. Bianco, and H. W. Kourkoutis. Atomic-Resolution Cryo-STEM Across Continuously Variable Temperature. *arXiv preprint arXiv:2001.11581*, 2020.

[267] S. J. Pennycook. The Impact of STEM Aberration Correction on Materials Science. *Ultramicroscopy*, 180: 22–33, 2017.

[268] Q. M. Ramasse. Twenty Years After: How "Aberration Correction in the STEM" Truly Placed a "A Synchrotron in a Microscope". *Ultramicroscopy*, 180:41–51, 2017.

[269] P. Hawkes. Aberration Correction Past and Present. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1903):3637–3664, 2009.

[270] G. H. Weber, C. Ophus, and L. Ramakrishnan. Automated Labeling of Electron Microscopy Images Using Deep Learning. In *Proceedings of MLHPC 2018: Machine Learning in HPC Environments, Held in conjunction with SC 2018: The International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 26–36. IEEE, 2018.

[271] M. Hutson. Artificial Intelligence Faces Reproducibility Crisis. *Science*, 359(6377):725–726, 2018.

[272] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. TensorFlow: A System for Large-Scale Machine Learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[273] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv preprint arXiv:1603.04467*, 2016.

[274] Gatan Microscopy Suite Software. Online: `www.gatan.com/products/tem-analysis/gatan-microscopy-suite-software`, 2020.

[275] B. Miller and S. Mick. Real-Time Data Processing Using Python in DigitalMicrograph. *Microscopy and Microanalysis*, 25(S2):234–235, 2019.

[276] V. C. Müller and N. Bostrom. Future Progress in Artificial Intelligence: A Survey of Expert Opinion. In *Fundamental Issues of Artificial Intelligence*, pages 555–572. Springer, 2016.

[277] A. Sarkar and S. Cooper. Towards Game Design via Creative Machine Learning (GDCML). In *2020 IEEE Conference on Games (CoG)*, pages 744–751. IEEE, 2020.

[278] M. Guzdial, N. Liao, and M. Riedl. Co-Creative Level Design via Machine Learning. *arXiv preprint arXiv:1809.09420*, 2018.

[279] A. Sethi, A. Sankaran, N. Panwar, S. Khare, and S. Mani. DLPaper2Code: Auto-Generation of Code from Deep Learning Research Papers. *arXiv preprint arXiv:1711.03543*, 2017.

[280] L. E. Lwakatare, A. Raj, I. Crnkovic, J. Bosch, and H. H. Olsson. Large-Scale Machine Learning Systems in Real-World Industrial Settings: A Review of Challenges and Solutions. *Information and Software Technology*, 127:106368, 2020.

[281] P. Gupta, A. Sharma, and R. Jindal. Scalable Machine-Learning Algorithms for Big Data Analytics: A Comprehensive Review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 6(6): 194–214, 2016.

[282] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley. The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1123–1132. IEEE, 2017.

[283] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison. Hidden Technical Debt in Machine Learning Systems. In *Advances in Neural Information Processing Systems*, pages 2503–2511, 2015.

[284] B. R. Johnson and S. K. Lam. Self-Organization, Natural Selection, and Evolution: Cellular Hardware and Genetic Software. *Bioscience*, 60(11):879–885, 2010.

[285] T. Salimans and D. P. Kingma. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. In *Advances in Neural Information Processing Systems*, pages 901–909, 2016.

[286] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral Normalization for Generative Adversarial Networks. *arXiv preprint arXiv:1802.05957*, 2018.

[287] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv preprint arXiv:1502.03167*, 2015.

Vita

This vita covers the following resume[8].

> J. M. Ede. Resume of Jeffrey Mark Ede. Zenodo, Online: `https://doi.org/10.5281/zenodo.4429077`, 2021

Last updated on 6th Jan 2021

Jeffrey Ede

Staff Page: Warwick://Jeffrey Ede
Email: j.m.ede@warwick.ac.uk **Phone:** +44 (0) 7593 883091

EDUCATION

University of Warwick

Doctor of Philosophy (PhD) in Physics:
From Oct 2017 (thesis is finished)
Master of Physics (MPhys) and
Bachelor of Science (BSc) in Physics:
Aug 2013 - Jul 2017, First Class with Honours

PERSONAL

Date of Birth: 13th Jul 1995
Nationality: English
Salary: Best Offer
Willing to Relocate: Yes
Remote Work: Prefer On-Site

LINKS

arXiv:// **Jeffrey Ede**
GitHub:// **Jeffrey Ede**
LinkedIn:// **Jeffrey Ede**
StackOverflow:// **Jeffrey Ede**

INTERESTS

Data curation and processing
Parallel and distributed computing
Automation
Machine learning

SKILLS

Programming

Over 10k lines:
Python • C/C++ • MATLAB • LaTex
Over 1k lines:
DigitalMicrograph • Java • R
Familiar:
Arduino • LabVIEW • Mathematica • Verilog • OpenCL • MySQL

Machine Learning

Training:
adversarial • reinforcement • supervised
Architectures:
actor-critic • AE • CNN • DNC • encoder-decoder • GAN • MLP • RNN • VAE • VAE-GAN
Miscellaneous:
dataset curation • style transfer • tSNE

SYNOPSIS

I am about to submit my finished doctoral thesis and want to arrange a job as soon as possible. My start date is flexible. I have four years of programming experience and a background in physics, machine learning, and automation.

EXPERIENCE

Researcher – Machine Learning / Electron Microscopy

From Oct 2017 at the University of Warwick
My doctoral thesis titled ”Advances in Electron Microscopy with Deep Learning” was completed under the supervision of Jeremy Sloan. Highlights include:

- Search engines based on variational autoencoders.
- Reinforcement learning to train recurrent neural networks to piecewise adapt sparse scans to specimens for compressed sensing.
- Generative adversarial networks for quantum mechanics and compressed sensing.
- Signal denoising for low electron dose imaging.
- Curation, management, and processing of large new machine learning datasets.

In addition, I was a teaching assistant in undergraduate labs for quantum conduction and electronics experiments.

Summer Internship – Atomic Force Microscopy

Jul - Sep 2017 at the University of Warwick
Programmatic automation of an atomic force microscope, lock-in amplifiers, and superconducting magnets in Marin Alexe's research group.

Summer Internship – Ultrafast Spectroscopy

Jul - Sep 2015 at the University of Warwick
Programmatic Fourier analysis and wavelet decomposition of broadband optical spectra to determine material properties in James Lloyd-Hughes' research group.

PUBLICATIONS

[1] J. M. Ede, "Review: Deep Learning in Electron Microscopy," *arXiv preprint arXiv:2009.08328*, 2020.

[2] J. M. Ede, "Warwick Electron Microscopy Datasets," *Machine Learning: Science and Technology*, vol. 1, no. 045003, 2020.

[3] J. M. Ede and R. Beanland, "Partial Scanning Transmission Electron Microscopy with Deep Learning," *Scientific Reports*, vol. 10, no. 1, pp. 1–10, 2020.

[4] J. M. Ede and R. Beanland, "Adaptive Learning Rate Clipping Stabilizes Learning," *Machine Learning: Science and Technology*, vol. 1, no. 1, p. 015011, 2020.

[5] J. M. Ede and R. Beanland, "Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder," *Ultramicroscopy*, vol. 202, pp. 18–25, 2019.

[6] J. M. Ede, J. J. P. Peters, J. Sloan, and R. Beanland, "Exit Wavefunction Reconstruction from Single Transmission Electron Micrographs with Deep Learning," *arXiv preprint arXiv:2001.10938*, 2020.

[7] J. M. Ede, "Adaptive Partial Scanning Transmission Electron microscopy with Reinforcement Learning," *arXiv preprint arXiv:2004.02786*, 2020.

[8] J. M. Ede, "Deep Learning Supersampled Scanning Transmission Electron Microscopy," *arXiv preprint arXiv:1910.10467*, 2019.