

Pixel Subset Super-Compression with a Generative Adversarial Network

Jeffrey M. Ede

j.m.ede@warwick.ac.uk

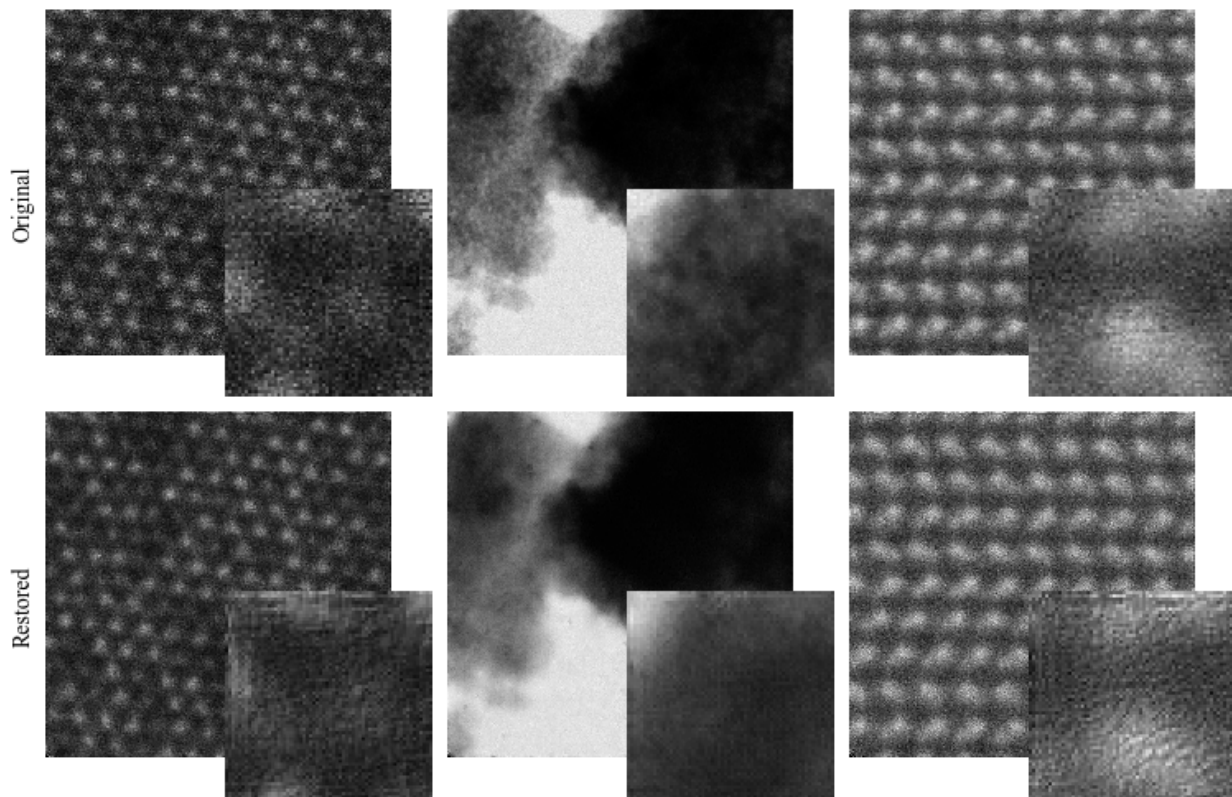


Figure 1: Example restorations of 512×512 scanning transmission electron microscopy images after $20\times$ compression using our process. Enlarged 64×64 regions from the top left of each image are inset to show fine characteristics.

Abstract: *Neural network-based image compression that uses a learned latent space suffers from information loss as information travels through the network. Instead, we propose selecting a set of pixels marked by a fixed mask and using these as the compressed representation. Spatial information implicitly encoded. By randomly distributing the marking pixels, this allows a large spatial frequency range to be encoded. We demonstrate that using a generative adversarial network with multi-scale discriminators enables the realistic restoration of 512×512 transmission and scanning*

transmission electron micrographs from $1/40$ and $1/100$ of their pixels. This includes the restoration of realistic noise characteristics. Our generative adversarial networks are compared against Euclidean loss networks that we trained end-to-end for $40\times$ compression of 512×512 TEM and STEM micrographs. They achieve final MSE performances of TODO and TODO, respectively. Noise in electron micrographs makes them difficult to encode, suggesting that our method has potential in easier domains where it may achieve even higher compression. Other contributions include the improvement of learning policies

for end-to-end training, balanced generator-discriminator learning, dynamic two-sided discriminator relaxation and application of spectral normalization to the discrimination of noisy images. We also propose a non-adversarial method to guide training and show that it outperforms natural statistics (outperforms pix2pixHD. We directly compare). Improved training with ancillary inner network trainer. Our code and pre-trained models are available at [github-repo-loc](#).

Keywords: electron microscopy, encryption, high resolution, super-compression, machine learning

1 Introduction

Data compression plays an important role in almost every scientific field. It addresses the need to store a large amount of information in a small amount of storage space, transmit a large amount of information through a limited bandwidth and more. Data compression can be financially motivated, performance motivated or both. A typical example is image compression to improve webpage loading times.

The advent of powerful general purpose graphical processing unit (GPU) acceleration[1, 2] has made the routine training of deep convolution neural networks[3, 4] (CNNs) practical in distributed settings[5, 6]. This has resulted in the widespread application of CNNs to most areas of computer vision, including image compression. Examples of neural network based image compression are autoencoders that learn latent representations[7, 8] and downsampling followed by learned upsampling[9, 10]. More involved methods compress images by enhancing existing codecs[11] or by using priors to learn a meaningful latent space[12].

In this paper, we present a new approach that uses a generative adversarial network (GAN) to restore images from a fixed random subset of their pixels. The use of a fixed random distribution; rather than a grid, of selection pixels allows information about more spatial frequencies to be encoded. Adversarial training results in the restoration of realistic images. Our method can be thought of as an extension to downsampling followed by learned upsampling.

Our method adds new capabilities to a popular image compression method and therefore has a wide range of applications where it extends its functionality. This includes any application where it is important for restored images to have realistic textures, noise characteristics or better coverage of all spatial frequencies. These are very general improvements.

We further improve beyond the state-of-the-art by developing new methods to train high-resolution GANs with multiple discriminators. These relax discriminators

to reduce greed, balance generator and discriminator learning and improve gradient stability. Our methods are especially designed to enhance the generation of noisy textures and fine detail found in electron micrographs. We demonstrate the improvement by direct comparison of our training protocols against pix2pixHD's[13] for 512×512 networks. Our training methods are also very consistent: 10/10 of the networks we present in this paper were successfully trained first-time without mode collapse. Basic method only works for TEM. Mode collapses for STEM as Jensen-Shannon divergence is a poor critic for noise. Expect Kullback-Liebler-based gradient penalized Wasserstein discriminators to change that.

2 Process

In this section, we introduce a process we have developed to compress and restore images using a fraction of their pixels. Our process is based on using a convolutional neural network (CNN) to take advantage of spatial correlations on multiple scales and is illustrated in fig. ?? . The process is

1. Resize or crop an image so that it is the same size as the CNN input. Without cropping, multiple sizes can be supported using multiple CNNs or a network trained for multiple input scales.
2. Select a fraction of pixels using a fixed mask that marks a random subset of pixels.
3. The selected fraction of pixels can be stored as a string of numbers by selecting them in a fixed order. This is possible because the mask is fixed.
4. Optionally, the string of numbers can be further compressed...
5. ...and decompressed using traditional methods.
6. To restore the image, the numbers are placed at their original locations on the fixed mask.
7. Numbers placed on the mask are processed by our neural network to restore the image. In this paper, we use a GAN generator to realistically restore images.

Compression is extremely fast as it is just the selection of a subset of pixels. Using a GAN also enables extremely high compression ratios. We demonstrate realistic restoration for $40 \times$ and $100 \times$ compression in this paper. This includes the restoration of fine detail like the noise in electron micrographs.

A key insight of our process is to compress images in the spatial domain rather than using a latent space. This allows spatial correlations to be implicitly encoded without the

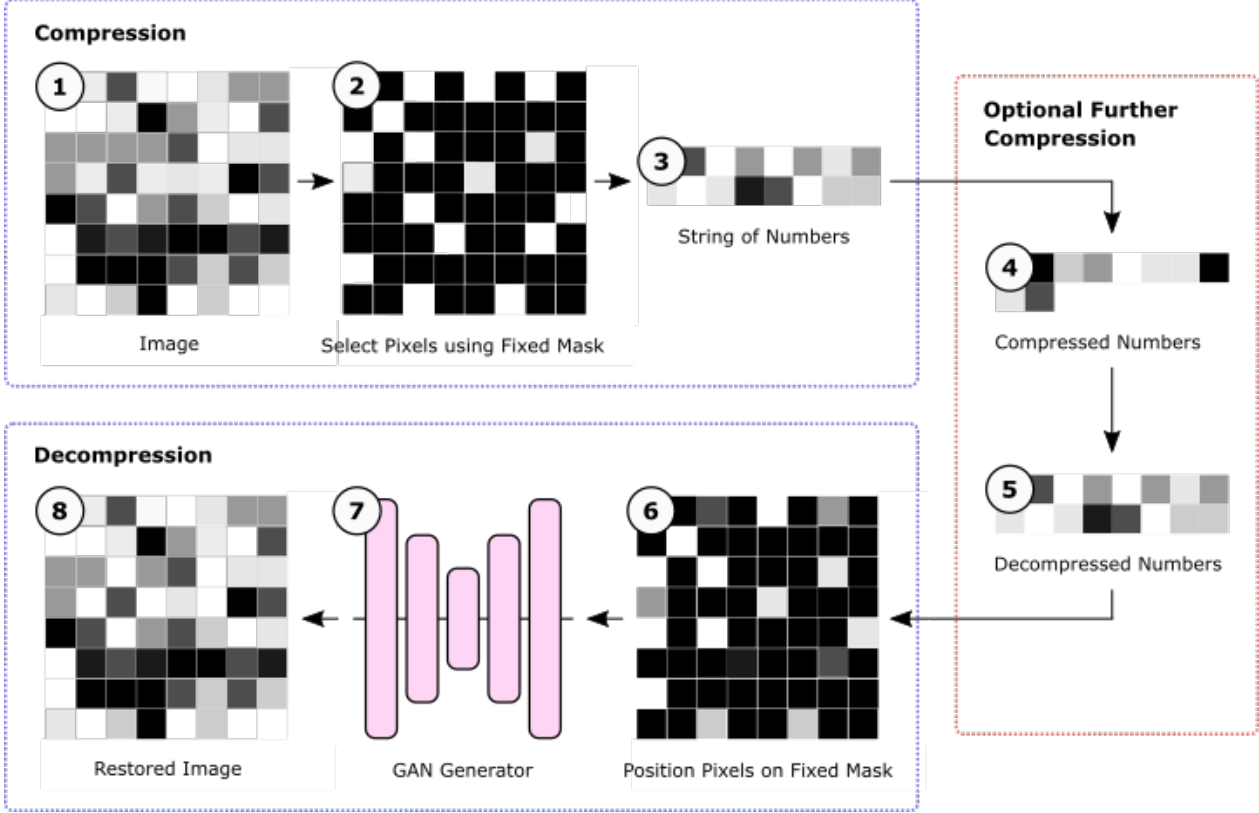


Figure 2: Image compression-decompression scheme. 1-3) A fraction of pixels are selected using a fixed mask to create a string of numbers. These can be stored as they are or 4) compressed further. Images are restored by 5) decompressing any further compression and 7) positioning the selected pixels back on the fixed mask. 7) Our generator network 8) restores the image from the positioned pixels.

signal loss that may accompany mapping to a latent space. Importantly, we also use a mask with randomly distributed; rather than grid-like, marks for pixel selection. This allows information at more spatial frequencies to be encoded.

In this paper, we will be demonstrating our process on TEM and STEM images. This will allow us to demonstrate its ability to restore fine detail such as noise characteristics. However, we expect significantly higher performance for images that are not noisy. Especially at higher compression ratios where fewer pixels are selected for compression.

3 Architecture

To restore realistic images from sparse 512×512 images of compressed data, we use a generative adversarial network. Our network is inspired by pix2pixHD[13] and uses three discriminators that act on multiple scales to train one generator. Our generator architecture is shown in fig. ??; our discriminator architecture is shown in fig. 9.

For our generator, we use an encoder-decoder that produces global-scale features in an 8 skip-3 residual block global enhancer. After upsampling, global features

are locally enhanced using 3 skip-3 residual blocks then upsampled to 512×512 for output. Importantly, a residual connection after the second convolution to the start of the local enhancer helps to overcome signal loss during global enhancement.

Multi-scale discriminators: To train our generator, we independently train three discriminators to process real and generated images on multiple scales. Specifically, we take random 512×512 , 256×256 and 128×128 crops, after reflection padding to prevent edge artefacts, and bilinearly downsample them to 70×70 images to be inputted to the discriminators. We chose multi-scale discriminators with 70×70 inputs as this is best for 256×256 generators in [15] and works for higher resolutions in [13].

We use three discriminators at different scales; rather a single discriminator, as multi-scale discriminators require fewer parameters, are more efficient computationally and produces better results[15]. Multi-scale discrimination also reduces unwanted periodic artefacts[13].

Nearest neighbour infilling: To increase input utilization, blank regions around positioned pixels are infilled with 1 nearest neighbour infilling.

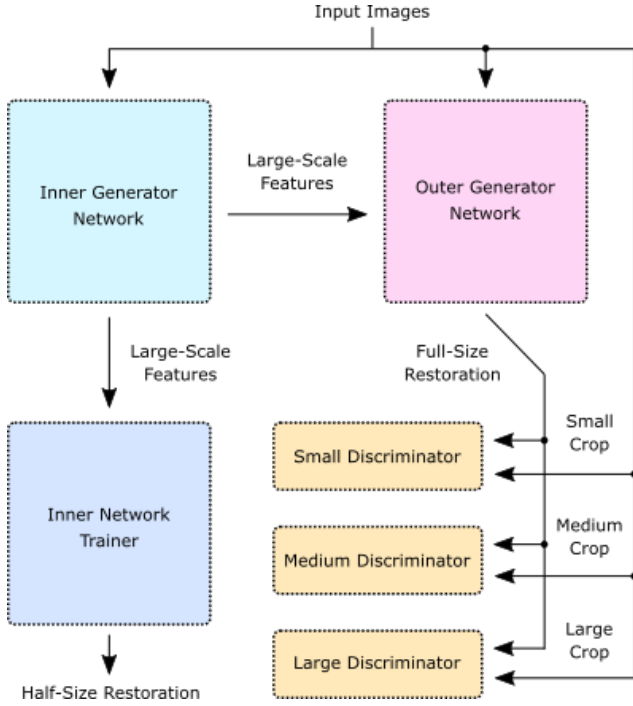


Figure 3: Simplified multi-scale generative adversarial network. An inner network generates large-scale features from inputs. These are translated to half-size restorations by a trainer network and recombined with the input to generate full-size restorations by an outer network. Multiple discriminators assess multi-scale crops from input images and full-size restorations.

Batch normalization: Batch normalization layers from [16] are applied in the generator after every convolution before activation, except after the last convolution. Batch normalization is also applied between the depthwise and pointwise convolutions of every depthwise separable convolution, similar to MobileNet[17], DeepLab3+[18] and Xception[19]. For numerical stability, we added $\varepsilon = 0.01$ to the running variances used for batch normalization.

Low batch sizes limit the convergence of GAN generators[20] and result in blurry images. Nevertheless, we trained all our networks with batch size 1 as it significantly improved the rate of convergence in the early stages of training. Importantly, this means that our batch size 1 results give a lower bound on performance. Weight normalization[21] typically gives higher performance at small batch sizes; however, it is outperformed by batch normalization[22] at typical batch sizes. For training with larger batch sizes, we recommend freezing batch normalization in the final third of training so that it does not limit convergence.

Input normalization: Images, I , were normalized to have values $I_N \in [-1, 1]$ before applying the fixed mask. Images input to the networks were linearly transformed $I_N \rightarrow I'_N = (I_N + 1)/2$, $I'_N \in [0, 1]$, before the first convolutions.

Activation: Leaky ReLU[23] activation is used in the generator and discriminators. In the generator we use slope 0.01; in the discriminators we use slope 0.2. The high slope in the discriminators encourages the use of more features.

Initialization: Generator weights were Xavier uniform initialized[24]; biases were zero initialized. Discriminators' weights were random normal initialized with mean 0.00 and standard deviation 0.03; biases were zero initialized.

Interpolation: Time-variance of interpolative systems; such as our GAN generator, can produce periodic artefacts. Even a good discriminator can struggle to correct these as they are a consequence of the time-variance of the generator, rather than a straightforward limitation of the loss function. To reduce periodic artefacts, we upsample using bilinear upsampling before; rather transpositional convolutions[25]. Transpositional convolution; sometimes called deconvolution after [26], is a faster alternative if periodic artefacts can be overlooked.

Padding: Generator feature spaces are reflection padded before convolutions to maintain feature space sizes. Discriminator convolutions are not pre-padded. The generator output was also reflection padded before taking random crops to input to the discriminators to reduce edge artefacts.

First kernel size: For compression by a factor of r , we chose a kernel of size

$$w = \text{ceil} \left[\sqrt{r} \right] \quad (1)$$

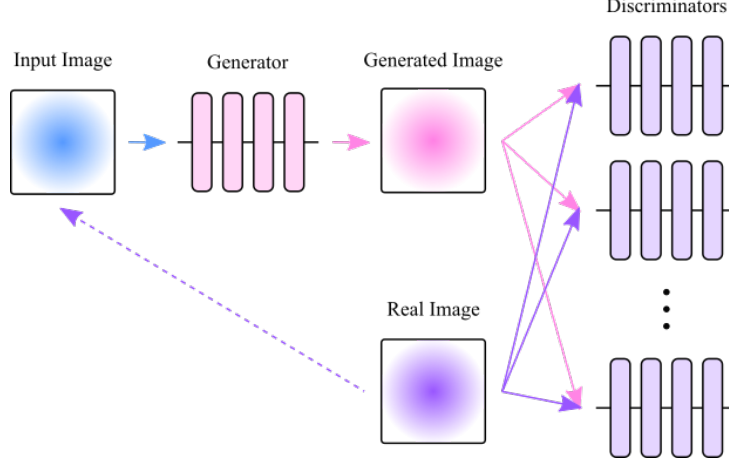


Figure 4: Setup for multi-discriminator generative adversarial image-to-image translation. Discriminators study real images and images translated by the generator at multiple scales to predict their realism.

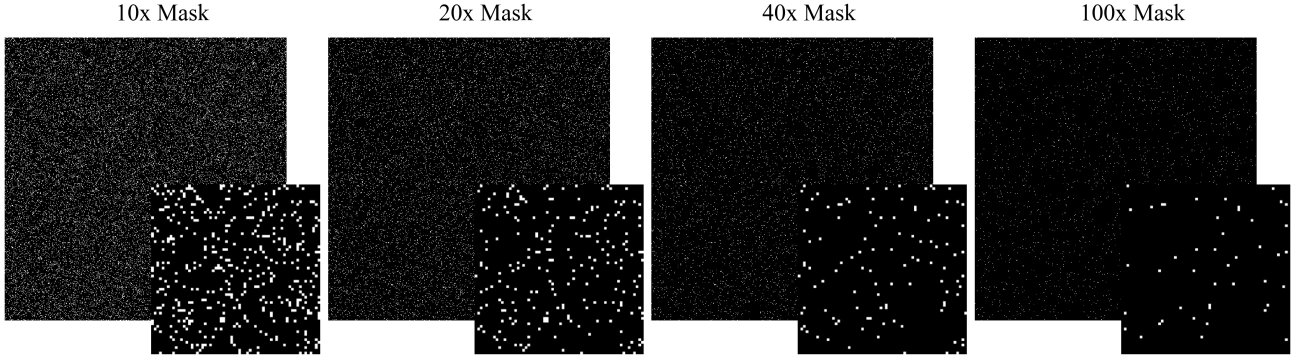


Figure 5: Fixed random 512×512 masks used for image compression. Pixels were marked by choosing numpy[14] random seed 1, sampling a 512×512 array from a numpy variate uniformly distributed in $[0, 1]$ and selecting the random numbers less than the reciprocal of the compression ratio. Enlarged 64×64 regions from the top-left of each mask are inset to ease comparison.

for the first generator convolution. This is the minimum integer-sized kernel with an area larger than the mean area per selected pixel. For $20\times$, $40\times$ and $100\times$ compression, the minimum kernel sizes, $w \times w$, are 5×5 , 7×7 and 11×11 ¹, respectively.

TODO: The instability at small compression rates for small first kernel sizes is probably not a quirk. Without pre-infilling, e.g. KNN, this was necessary to ensure spatially correlated flows into future convolutions.

Data augmentation: Example images were subject to a random combination of flips and 90deg rotations to augment the dataset by a factor of 8.

Multiscale Cropping: To reduce edge and periodic patterns in GAN outputs, images are often reflection padded

before being cropped for the discriminators. However, this biases the discriminators towards rewarding the generation of images with mirror symmetry. This is problematic when restoring images of atom columns, which often have high local mirror symmetry, when it is important to restore lower-symmetry regions, e.g. that show defects, accurately.

We propose the use of a dedicated probability distribution that weights the probability of each part of the image being cropped so that each pixel has the same probability of being selected as for reflection padding. Consider an n -dimensional example with dimensions $d = \{d_0, d_1, \dots, d_{n-1}, d_n\}$ that a crop of size $c = \{c_0, c_1, \dots, c_{n-1}, c_n\}$ is to be taken from. Traditionally, the image would be reflection padded by $c - 1$ so that every n -dimensional cell has equal probability of being sampled.

An alternative is to use the distribution.

¹We also experimented with 9×9 and 7×7 kernels for $100\times$ compression. There was no obvious difference for 9×9 ; however, the generator was less stable at the start of training for 7×7 .

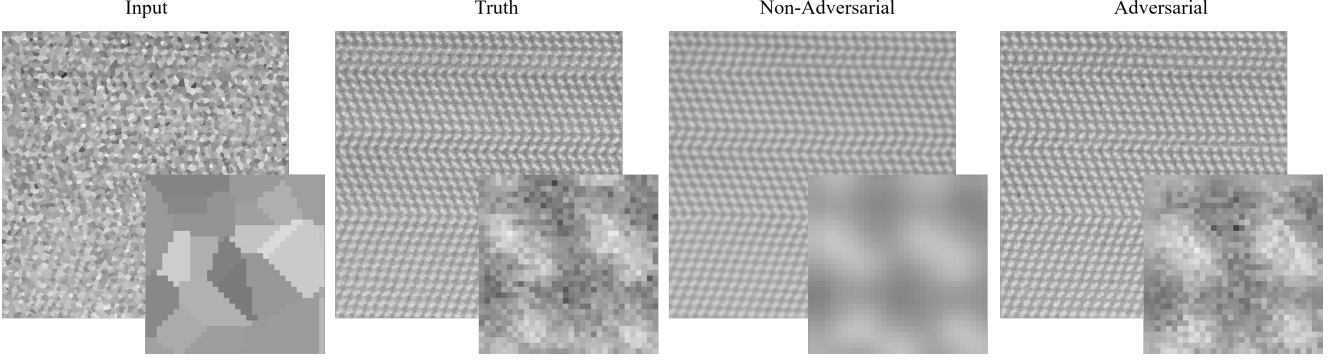


Figure 6: Example adversarial and non-adversarial decomposition of nearest neighbour infilled positioned pixels for a $40\times$ compressed image. Adversarial training produces an image with realistic noise characteristics whereas non-adversarial training produces a blurred image. Enlarged 32×32 regions from the top left of each image are inset to ease comparison.

4 Single-Stage Training

In this section, we present our training hyperparameters and learning policies used to train our generators to restore images. These generators were trained as part of generative adversarial networks by three discriminators acting at multiple scales and were guided by either natural statistics or mean squared errors (MSEs). We also trained some of our generators with a non-adversarial MSE for comparison.

Our networks were trained using ADAM optimized [27] stochastic gradient descent with one Nvidia GTX 1080 Ti GPU. We used the TensorFlow[6] deep learning framework.

4.1 GAN Training with Natural Statistics

Iterations: For high-dose TEM image compression, our generator and discriminators were trained for 210000 iterations. For STEM image compression, they were trained for 300000 iterations. Our generator and discriminators were trained once per iteration. These numbers are in-line with the couple of thousand iterations other GAN trainers have used to generate 512×512 images with multi-scale discriminators e.g. [13].

Discriminator crops: Three discriminators acting on random 512×512 , 256×256 and 128×128 crops acted as adversaries to the generator.

Optimization: For the generator, we used a constant learning rate of 0.0002 for the first two-thirds of training. This was linearly decayed stepwise to zero in the final third of training after every one-fifth of the remaining third of training. Training for two-thirds before decaying the learning rate is recommended in [?]. To speed up training, we used a high first moment of the momentum, $\beta_1 = 0.9$, at the start of training and linearly decay β_1 to $\beta_1 = \beta_{\text{final}}$ in the first third of training. Here we chose $\beta_{\text{final}} = 0.75$ for

$100\times$ compression² and $\beta_{\text{final}} = 0.5$ for $40\times$ compression. We kept $\beta_1 = \beta_{\text{final}}$ for the final two-thirds to training so that the generator would be responsive to individual examples. Low β_1 ; specifically $\beta_1 = 0.5$, is shown to produce higher-quality generations than the recommended[27] $\beta_1 = 0.9$ in other experiments e.g. [28].

The choice of β_{final} for the generator is a compromise between stability and responsiveness to individual examples. We use a high $\beta_{\text{final}} = 0.9$ for $100\times$ compression of electron micrographs as the sparsity of input pixels coupled with their noisiness resulted in varying outputs. This results in natural statistics varying and makes their addition to the generator loss function less meaningful at low momentums. For $40\times$ compression there is more information, making natural statistics more stable. Consequently, we use a low $\beta_{\text{final}} = 0.5$ for increased responsiveness to individual examples.

For the discriminators, we used a maximum learning rate of 0.00005 at the start of training and linearly increased it to 0.00025 in the first twentieth of training. The lower learning rate at the start is a precaution against any numerical stability that may be produced by the untrained discriminator weights. It also helps avoid real and fake predictions collapsing to single modes as a result of the discriminators being able to discriminate between the generated and real distributions too easily in the early stages of training. We used a constant $\beta_1 = 0.5$ for the discriminators.

For high-dose TEM, the generator and discriminators were trained for 210000 iterations. For STEM; which had much noisier examples, the generator and discriminators were trained for 300000 iterations.

Balanced learning: If the discriminators become too good, the predictions for real and generated examples will

²Except in our pre-trained model for $100\times$ TEM compression where we used $\beta_{\text{final}} = 0.5$ as we had not made this refinement yet.

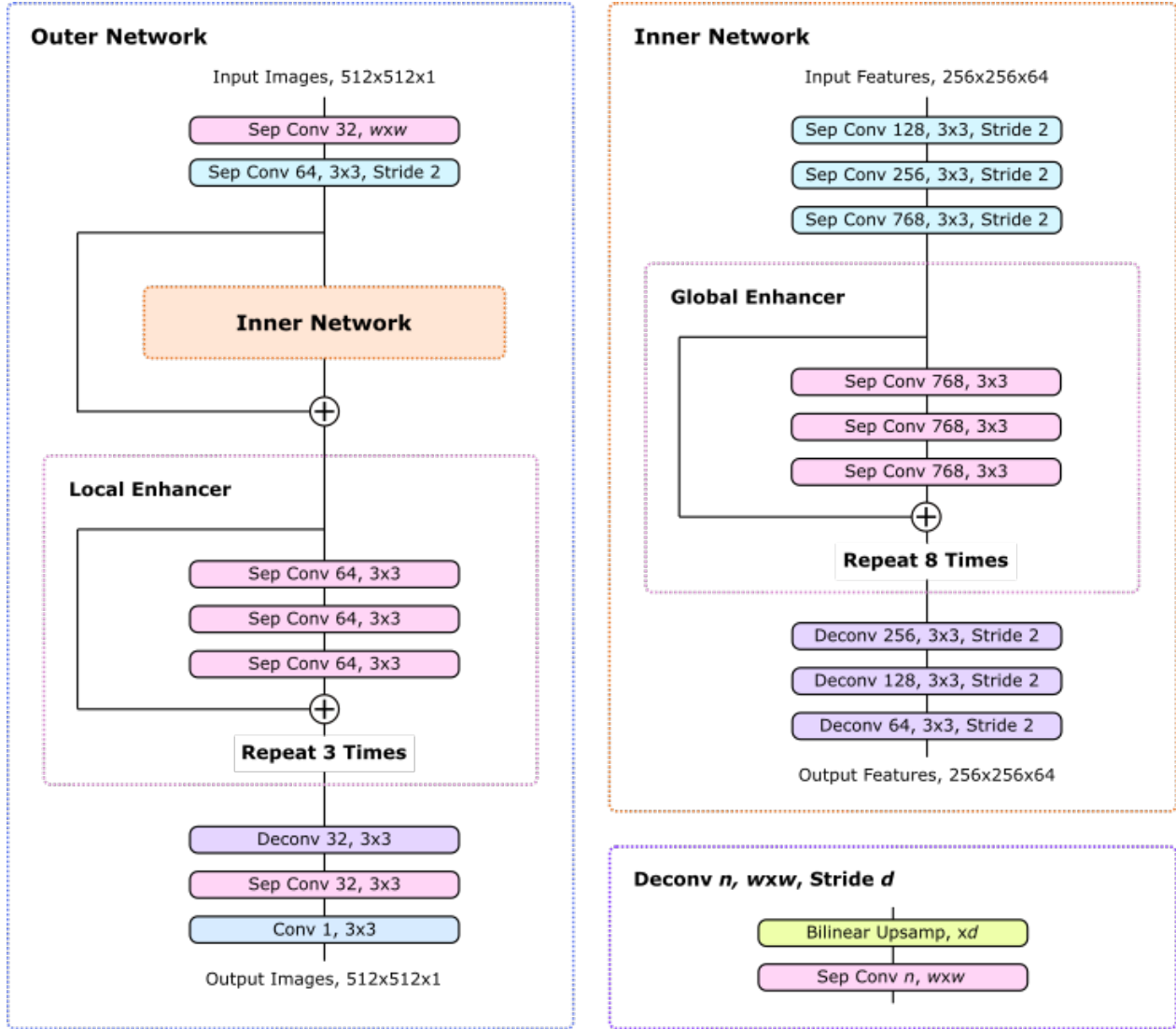


Figure 7: One-stage generator that restores images from a fixed subset of their pixels. Large scale features developed by the inner network are locally enhanced by the outer network and turned into images. A residual connection across the inner network reduces signal attenuation.

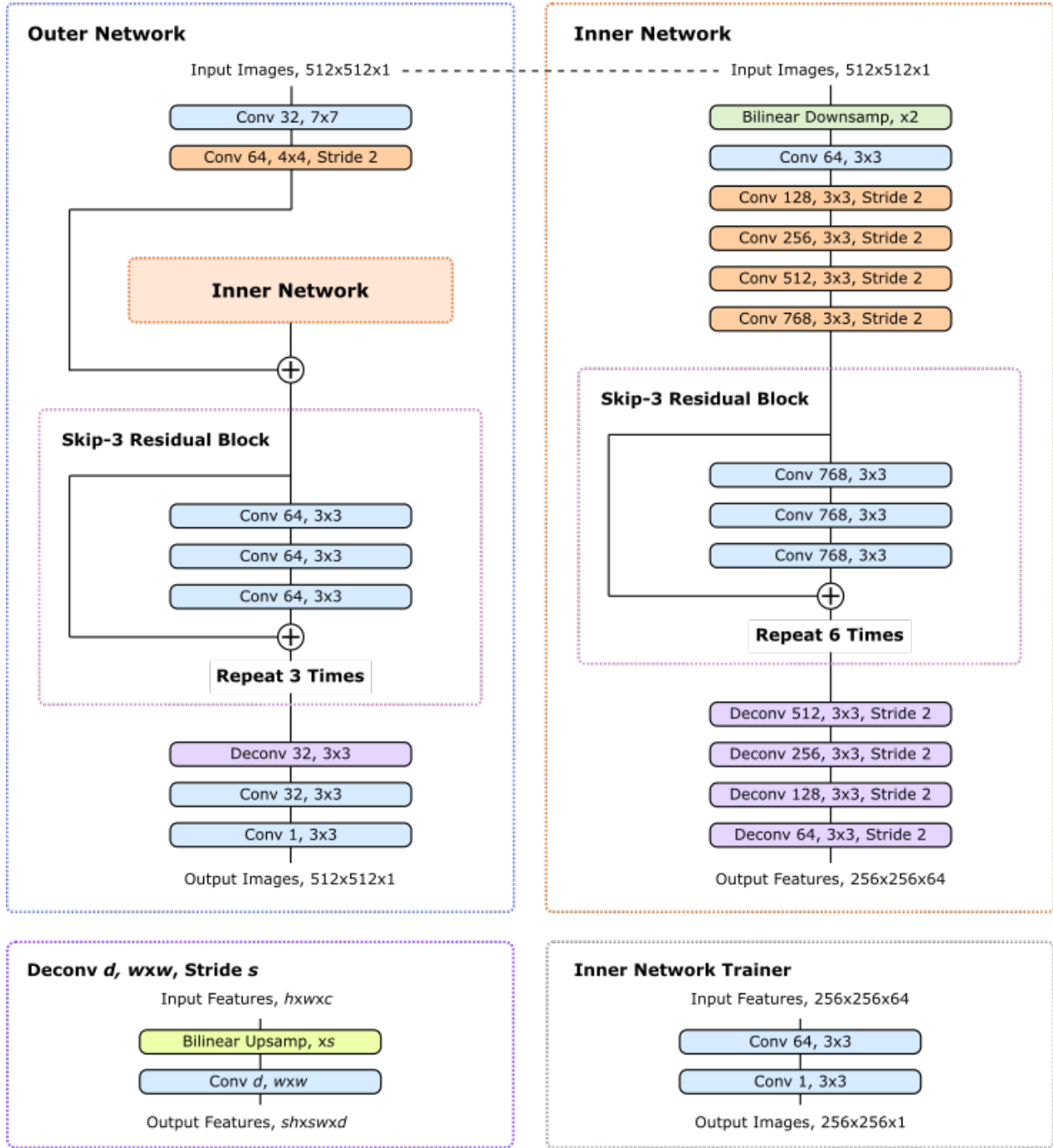


Figure 8: Two-stage generator that restores images from a fixed subset of their pixels. A dashed line indicates that the same image is input to the inner and outer network. Large scale features developed by the inner network are locally enhanced by the outer network and turned into images. An ancillary trainer network restores images from inner network features to provide dedicated feedback.

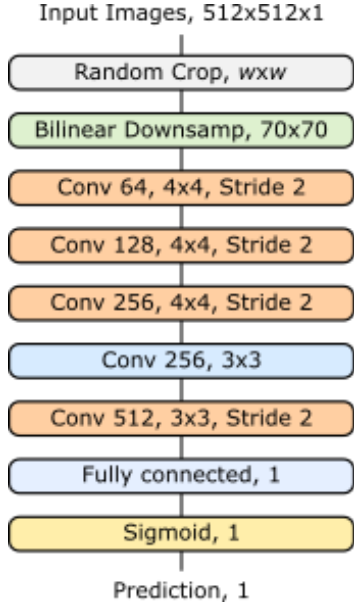


Figure 9: Discriminator for one-stage generation that predicts image labels from random $w \times w$ crops. Generators are adversarially trained by multiple discriminators with different w .

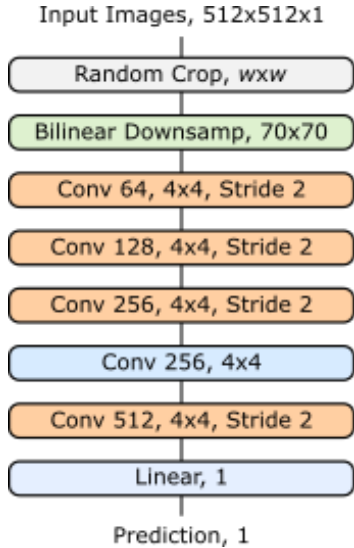


Figure 10: Discriminator for two-stage generation that predicts image labels from random $w \times w$ crops. Generators are adversarially trained by multiple discriminators with different w .

collapse to a single of few modes. Alternately, if the discriminators' learning rates are too low, they will struggle to learn discriminating features in response to changes in the generator. Neither of these imbalanced scenarios is desirable as the discriminators will not back-propagate meaningful gradients for the generator to learn from.

To reduce imbalanced learning, we dynamically adjust the learning rate of all $n_D = 3$ discriminators based on the running means

$$\mu_{\text{real}} \rightarrow \beta_{\text{real}} \mu_{\text{real}} + \frac{1 - \beta_{\text{real}}}{n_D} \sum_i^{n_D} D_i(x) \quad (2)$$

$$\mu_{\text{fake}} \rightarrow \beta_{\text{fake}} \mu_{\text{fake}} + \frac{1 - \beta_{\text{fake}}}{n_D} \sum_i^{n_D} D_i(G(x)) \quad (3)$$

where we chose $\beta_{\text{real}} = \beta_{\text{fake}} = 0.99$.

The discriminators' learning rate, η , was scaled from the imbalanced learning rate, η_0 , using a geometric product of logistic functions

$$\eta = 2\eta_0 \sqrt{\max[s_1 s_2, \varepsilon]} \quad (4)$$

$$s_1 = \frac{1}{1 + \exp[-a_{\text{real}}(1 - \mu_{\text{real}} - b_{\text{real}})]} \quad (5)$$

$$s_2 = \frac{1}{1 + \exp[-a_{\text{fake}}(\mu_{\text{fake}} - b_{\text{fake}})]} \quad (6)$$

where we chose $a_{\text{real}} = a_{\text{fake}} = 10$ and $b_{\text{real}} = b_{\text{fake}} = 0.5$. To avoid possible square roots of negative values and potential numerical instability of the square root function, we chose $\varepsilon = 10^{-5}$.

We note that learning rates can be dynamically adjusted for each discriminator individually; rather than for all discriminators using running mean predictions. However, individual learning rates would drive the discriminators towards similar behavior. If each discriminator's predictions are similar, their losses will be similar and learning at each scale will be similarly important. This would undermine a key strength of multi-scale discriminators: that differences between losses are higher at scales where discrimination is easier.

Dynamic relaxation: We modified the vanilla loss[29] for the i th of the $n_D = 3$ discriminators

$$L_{D_i}^{\text{vanilla}} = -\log[D_i(x)] - \log[1 - D_i(G(x))] \quad (7)$$

to

$$L_{D_i}^{\text{relax}} = -\log[D_i(x)(1 - D_i(x))^{\gamma_{\text{real}}}] - \log[(1 - D_i(G(x)))D_i(G(x))^{\gamma_{\text{fake}}}] \quad (8)$$

where γ_{real} and γ_{fake} were controlled using the default settings for our appendicized simple two-sided dynamic relaxation algorithm.

In practice, $\gamma_{\text{real}} = \gamma_{\text{fake}} = 1/9$ was constant throughout training, making the dynamic nature of the relaxation unimportant. Our dynamic relaxation is mainly a precaution against extreme discriminator greed at the start of training limiting convergence[30]³.

For the generator, we summed the vanilla loss for each discriminator without modification

$$L_G^{\text{relax}} = \sum_i^{n_D} \log[D_i(x)] + \log[1 - D_i(G(x))] \quad (9)$$

Label smoothing: Dynamic two-sided label smoothing based on discriminator performance was used to help prevent discriminator mode collapse. This is most important in the early stages of training when the difference between $x \sim \mathbb{P}_{\text{real}}$ and $G(x)$ is obvious. To track our $n_D = 3$ discriminators' performance, we use running means for real and fake predictions

$$\mu_{\text{real}} \rightarrow \beta_{\text{real}} \mu_{\text{real}} + \frac{1 - \beta_{\text{real}}}{n_D} \sum_i^{n_D} D_i(x) \quad (10)$$

$$\mu_{\text{fake}} \rightarrow \beta_{\text{fake}} \mu_{\text{fake}} + \frac{1 - \beta_{\text{fake}}}{n_D} \sum_i^{n_D} D_i(G(x)) \quad (11)$$

where we chose $\beta_{\text{real}} = \beta_{\text{fake}} = 0.99$. In this case, these running means are the same as those used to balance learning. However, they can be different.

The running means were used to smooth real and fake labels

$$1 \rightarrow [1 - \max(\mu_{\text{real}} - \mu_{\text{fake}}, 0)/h_{\text{real}}, 1] \quad (\text{real}) \quad (12)$$

$$0 \rightarrow [0, \max(\mu_{\text{real}} - \mu_{\text{fake}}, 0)/h_{\text{fake}}] \quad (\text{fake}) \quad (13)$$

where we chose $h_{\text{real}} = h_{\text{fake}} = 3$.

Our dynamic label smoothing is designed to prevent discriminator mode collapse in the early stages of training. It does this by increasing smoothing in the early stages of training when $x \sim \mathbb{P}_{\text{real}}$ and $G(x)$ are easily discriminated; making $\mu_{\text{real}} - \mu_{\text{fake}}$ large. In the final stages of training where $G(x) \rightarrow x$, $\mu_{\text{real}} - \mu_{\text{fake}} \rightarrow 0$. This means that dynamic label smoothing will not significantly limit the ability of the discriminator to focus on fine discriminating details in the final stages of training.

Natural statistics: Each of the $n_D = 3$ discriminators imagines features that distinguish between real and generated images. Differences between these features can be compared to calculate a natural loss, L_{nat} , that measures how real generated images look on multiple scales. Specifically, we calculated the mean absolute

differences between the features generated by the first $n_l = 4$ discriminator convolutions for each discriminator. Our natural statistic loss is given by the mean of these means after multiplication by a hyperparameter, λ_{nat} , that controls the contribution of natural statistics to the loss function,

$$L_{\text{nat}} = \frac{\lambda_{\text{nat}}}{n_D n_l} \sum_i^{n_D} \sum_j^{n_l} |D_{i,j}(G(x)) - D_{i,j}(x)|. \quad (14)$$

For our multi-scale discriminator, we chose $\lambda_{\text{nat}} = 120$.

Some authors use other priors to compute natural statistics. For instance, it is common to calculate mean squared; rather than absolute, errors as squared errors follow from a Gaussian prior with unity covariance for differences between features[12]. However, we note that that discriminators are ever-changing, non-linear systems that are likely to generate some large differences between features, even for very similar images. Since MSEs are disproportionately affected by small portions of features with large differences, which may not be meaningful, MSEs are typically less stable than absolute errors, especially in small GAN discriminators.

Our multi-scale discriminators downsample 70×70 inputs so we chose to use more stable absolute differences; however, we note that mean square errors may be more appropriate in larger discriminators. This is because discriminators with large feature spaces dampen instability by considering more features. If discriminator instability is not limiting, mean squared differences will make natural statistics more responsive to large differences between features.

Non-adversarial guidance: In the early stages of training, we add a non-adversarial mean squared error (MSE) loss, L_{MSE} , between the real and generated images to guide the generator. This speeds up early training as the generator's training no longer stalled by the discriminators' learning. Importantly, we Gaussian blur the real and generated images before calculating the mean squared error. This stops the MSE varying as a result of different noise levels in examples, providing more stable feedback to the generator. In our experiments, we used a symmetric 7×7 Gaussian blurring kernel with a standard deviation of 3.5 px.

We take the square root of high MSEs to prevent the network from being too disturbed by high errors using the rule

$$L_{\text{MSE}} = \begin{cases} \text{MSE}, & t\text{MSE} \leq 1 \\ \sqrt{\text{MSE}/t}, & t\text{MSE} > 1 \end{cases} \quad (15)$$

where we chose $t = 1000$.

The addition of L_{MSE} to the generator loss function was controlled by a hyperparameter, λ_{MSE} . We chose $\lambda_{\text{MSE}} = 1000$ at the start of training and linearly it decayed to $\lambda_{\text{MSE}} = 0$ in the first twelfth of training.

³We disagree with the authors' claim that relaxation is effective because it results in more modes being explored. Instead, we think the main reason is similar to that for batch normalization[31]: it helps maintain similar gradients throughout training.

L2 regularization: During training, L2 regularization[32] losses for the generator, $L_{L2,G}$, and i th of the $n_D = 3$ discriminators, L_{L2,D_i} , given by the quadrature sums of their trainable variables, were added to their loss functions. For the generator, the L2 loss was multiplied by $\lambda_{L2,G} = 10^{-5}$. For the i th discriminator, the L2 loss was multiplied by $\lambda_{L2,D_i} = 10^{-3}$. Adding an L2 loss helped to prevent the trainable variables growing unbounded, decreasing the ability of the networks to learn[21].

Spectral normalization: Each discriminator is spectrally normalized[28] using the power iteration method with 1 iteration per set of training operations. This regularizes the discriminators so that they made decisions using more input features. Spectral normalization helps to stabilize training, prevent mode collapse and develop more informative gradients for the generator to learn from.

Experience replay: To reduce destabilizing discriminator oscillations, we maintained an experience replay[33, 34] with 50 examples. Prioritizing the replay significant experiences improves reinforcement learning[35], so we only replay hard examples. We define hard examples to be those that the generator has the highest natural statistics losses for. Examples were swapped into the the experience replay on average $r = 0.2$ times per iteration and independently sampled without removal with probability $p_{\text{use}} = 0.2$.

In detail, examples were added to the experience replay if their natural statistics losses were higher than a threshold, L_t , where the subscript t is for threshold. This threshold was calculated from the first and second moments of the discriminators' natural statistics, $L_{\text{nat},1}$ and $L_{\text{nat},2}$, using decaying means

$$L_{\text{nat},1} = \beta_{\text{nat},1} L_{\text{nat},1} + (1 - \beta_{\text{nat},1}) L_{\text{nat}} \quad (16)$$

$$L_{\text{nat},2} = \beta_{\text{nat},2} L_{\text{nat},2} + (1 - \beta_{\text{nat},2}) L_{\text{nat}}^2 \quad (17)$$

$$\sigma_{\text{nat}}^2 = L_{\text{nat},2} - L_{\text{nat},1}^2 \quad (18)$$

where we chose $\beta_{\text{nat},1} = \beta_{\text{nat},2} = 0.99$.

To calculate L_t , we also used a decaying mean to monitor the rate that new examples were added to the replay, p_{in} ,

$$p_{\text{in}} \rightarrow \begin{cases} \beta_{\text{in}} p_{\text{in}}, & L_{\text{nat}} < L_t \\ \beta_{\text{in}} p_{\text{in}} + (1 - \beta_{\text{in}}), & L_{\text{nat}} \geq L_t \end{cases} \quad (19)$$

where we chose $\beta_{\text{in}} = 0.97$.

We combined the moments of the natural statistics and the rate of new examples being added to the replay to inform the update of L_t

$$L_t \rightarrow \begin{cases} \beta_t L_t \\ + (1 - \beta_t)(L_{\text{nat},1} + \Delta L^\uparrow), & p_{\text{in}} < p_{\text{use}} \\ \beta_t L_t \\ + (1 - \beta_t)(L_{\text{nat},1} + \Delta L^\downarrow), & p_{\text{in}} \geq p_{\text{use}} \end{cases} \quad (20)$$

where we chose $\beta_t = 0.99$ and incremented the buffer threshold by $\Delta L^\uparrow = -\Delta L^\downarrow = 5\sigma_{\text{nat}}$.

The calculation of L_t can be improved by using small geometric; rather than arithmetic, adjustments, restricting it to being positive by adding the update $L_t \rightarrow \max[L_t, 0]$ and accounting for asymmetric natural statistics losses, $\Delta L^\uparrow \neq -\Delta L^\downarrow$. However, further improvements to L_t accuracy are unlikely to have a significant effect on training.

Our method can be extended to the sampling probabilities of individual examples in the experience replay. This would allow the replay probabilities to be increased for examples with higher natural statistics losses. However, we did not feel this was necessary as we already select hard examples for the experience replay. In addition, high losses may be a momentary quirk of the discriminator.

Loss function: The total loss functions for the generator, L_G , and i th of the n_D discriminators, L_{D_i} , are

$$L_G = L_G^{\text{relax}} + \lambda_{\text{nat}} L_{\text{nat}} + \lambda_{\text{MSE}} L_{\text{MSE}} + \lambda_{L2,G} L_{L2,G} \quad (21)$$

$$L_{D_i} = L_{D_i}^{\text{relax}} + \lambda_{L2,D_i} L_{L2,D_i} \quad (22)$$

where we chose $\lambda_{\text{nat}} = 120$, $\lambda_{L2,G} = 10^{-5}$, $\lambda_{L2,D_i} = 10^{-3}$ and λ_{MSE} started at 1000 and was linearly decayed to 0 in the first twelfth of training.

Pre-trained models: We provide GANs trained with natural statistics for $40\times$ and $100\times$ compression of 512×512 TEM and STEM images.

4.2 GAN Training with MSE Guidance

Our training hyperparameters and learning protocols for training without natural statistics are similar to those for training with natural statistics. The differences are: (1) we kept $\lambda_{\text{MSE}} = 1000$ and $\lambda_{\text{nat}} = 0$ constant throughout training and (2) discriminators acted on downsampled random 512×512 , 225×225 and 70×70 crops; rather than downsampled random 512×512 , 256×256 and 128×128 crops.

Training without natural statistics is motivated by the observation that they limit convergence; especially at high compression ratios. This is because pixel sparsity and noise in the electron micrographs we are trying to compress make it difficult for the generator to learn consistent mappings. This is a problem as our discriminators do not have pooling layers, making them sensitive to spatial translation. Additionally, our discriminators are sensitive to noise characteristics that may vary between otherwise similar images. Altogether, this means that natural statistics losses can vary considerably for similar inputs, limiting the convergence of the generator.

Instead of using unstable natural statistics, we introduce a MSE loss to guide convergence. If a simple MSE was used, the generator would learn to output smooth images

that minimize the average MSE. However, smooth images are not realistic, conflicting with the adversarial loss. We solve this problem by Gaussian blurring images before calculating MSEs. This relegates the MSE loss to guiding the restoration of images with the correct overall structures while the adversarial loss controls fine details.

Pre-trained models: We provide GANs trained without natural statistics for 20 \times , 40 \times and 100 \times compression of 512 \times 512 TEM and STEM images.

4.3 MSE Training

The purpose of this paper is to showcase the ability of a GAN to restore images from a fixed set of randomly distributed pixels. However, natural statistics and binary-cross entropy losses measure perceptual; rather than Euclidean, distances between real and generated images. For comparison, we therefore trained non-adversarial generators for 40 \times TEM and STEM compression using a MSE and L2 error. In this section, we present our non-adversarial training hyperparameters and learning policy.

Perceptual distances are prone to producing obvious physical differences between original and generated images. For instance, our and others’[13] GAN generators often struggle to restore images with the correct brightness; especially for very bright or dark images. We confirm the observation in [13] that natural statistics improve coloration over vanilla losses; however, natural statistics are still a perceptual measure of distance. Ultimately, natural statistics are limited by the scale-invariance of our discriminators’ leaky ReLU activations. Spectral normalization also limits the ability of our discriminators to imagine large differences between features. This problem is furthered by our decision to use absolute differences between discriminator features; rather than a higher moment, to calculate natural statistics. The low moment further limits the discriminators’ already-limited ability to punish significant differences.

Iterations: TEM and STEM compression networks were trained for 100000 iterations.

Optimizer: We started with an initial learning rate $\eta_0 = 1$ and stepwise exponentially decayed it to

$$\eta = \eta_0 a^{\lfloor \text{iteration}/b \rfloor}, \quad (23)$$

where we chose $a = 0.7$ and $b = 12500$.

We started with a first moment of the momentum decay rate $\beta_1 = 0.9$ and linearly decayed it to $\beta_1 = 0.5$ over 100000 training iterations.

Loss function: We used an improved MSE between real and generated examples. To improve the MSE, we convolved the real and generated images with a 3 \times 3

TEM

	MAE		SSIM	
	Mean	Std Dev	Mean	Std Dev
10 \times , MSE Guidance	✓	✓	✓	✓
20 \times , MSE Guidance	✓	✓	✓	✗
40 \times , MSE Guidance	✓	✓	✓	✗
40 \times , Natural Guidance	✓	✓	✓	✗
40 \times , MSE Only	✓	✓	✓	✗
100 \times , Natural Guidance	✗	✓	✓	✗
100 \times , Natural Guidance	✗	✓	✓	✗

Table 1: Performance statistics

symmetric Gaussian kernel with standard deviation 1.5 px for TEM and a 5 \times 5 symmetric Gaussian kernel with standard deviation 2.5 px for TEM. This made the loss more stable by reducing its sensitivity to different noise levels in images.

Similar to adversarial training, we take the square root of high MSEs to prevent the network from being too disturbed by high errors using the rule

$$L_{\text{MSE}} = \begin{cases} \text{MSE}, & t\text{MSE} \leq 1 \\ \sqrt{\text{MSE}/t}, & t\text{MSE} > 1 \end{cases} \quad (24)$$

where we chose $t = 1000$.

Finally, we added an L2 regularization loss, L_{L2} , given by the quadrature sum of the trainable variables. Our total loss, L , is

$$L = L_{\text{MSE}} + \lambda_{\text{L2}} L_{\text{L2}}, \quad (25)$$

where we chose $\lambda_{\text{L2}} = 10^{-8}$ as the relative contribution of the L2 error to the total error.

Pre-trained models: We provide MSE-trained generators for 40 \times compression of 512 \times 512 TEM and STEM images.

Ancillary loss network: We resolve the image predicted by the inner network using a two-convolution ancillary network acting on the features flowing into the long residual connection. In our experiments, we found one convolution to be sufficient and only add the second convolution as a precaution.

5 Two-Stage Training

Architecture refinement for two-stage training. The experiments can be divided into a few major categories

1. Methods that increase the number of input-dependent input pixels. These include the default implementation of OpenCV’s[36] Gaussian blurring function and Navier-Stokes infilling[37], and $k=1$ nearest neighbour infilling using Scipy’s[38] exact Euclidean distance transform. The results are shown in fig. ??.

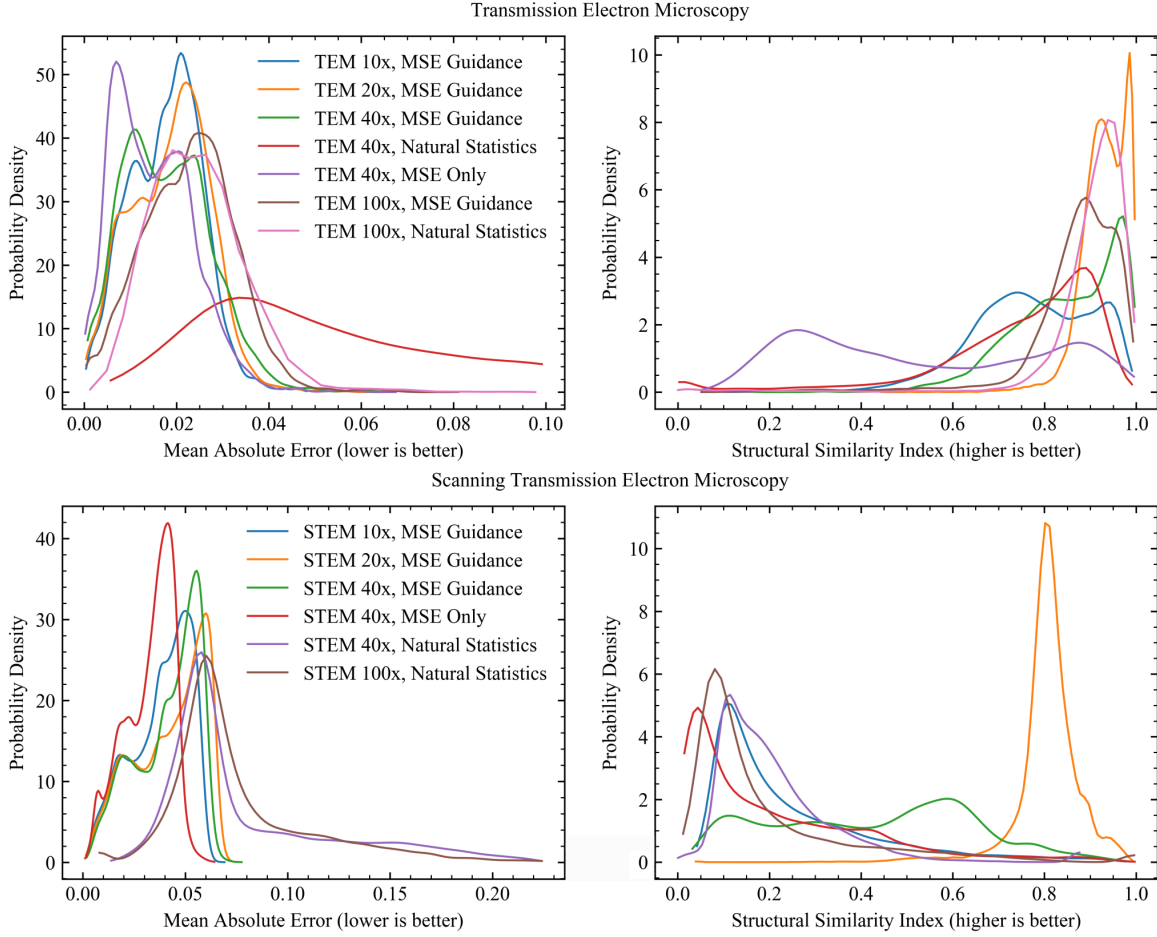


Figure 11: Mean absolute error (MAE) and structural similarity index (SSIM) performances of networks trained in a single stage on 20000 test set examples. Statistics are reported for ground truth images in $[0, 1]$. Decreasing TEM and STEM MAE PDF tails are truncated at 0.1 and 0.225, respectively. TODO: These figures show depthwise sepeparable convolutions with non-Wasserstein discriminators are sufficient for TEM; however, they fail to generate realistic fine detail for STEM. TODO: STEM performance using rank convolutions and Wasserstein discriminators.

#	Network Description	End Loss	Improvement
1	40× compression. 7×7 convolution before residual connection.	?	?
2	Input blurred by 11×11 Gaussian kernel.	?	?
3	Input Navier-Stokes infilled.	?	?
4	First 2 outer network skip-3 blocks replaced with 3 skip-2 blocks.	?	?
5	Concatenation and 1×1 convolution instead of long residual.	?	?
6	Input $k=1$ nearest neighbour infilled.	?	✓
7	No long residual. 768→1024 feature convolution. 5 D per G train op.	?	?
8	Squared difference discriminator. Bilinear downsampling by a factor of 4.	?	?
9	One-Stage training. No skip-3 residual blocks.	?	?
10	Ancillary loss network. Trainer network trained with combined loss.	?	?
11	Ancillary loss network with 2 convolutions. Only inner network loss.	?	?
12	150k iteration MSE fine-tuning for two-stage network training.	?	?
13	Single-stage training without ancillary loss tower	?	?
14	300k iteration MSE fine-tuning for two-stage network training.	?	?
15	Adversarial training with ancillary network. Blurred output.	?	?
16	100x compression. Adversarial training with ancillary tower. Blurred output.	?	?
17	Smaller discriminator receptive fields. Higher MSE contribution.	?	?

Table 2: Development of our two-stage adversarial network-in-network in chronological order. Improvements, marked with ✓, were kept whereas devolutions, marked with ✗, were not. End losses are means of the last 20000 training iterations.

- Major architecture and training variations. These include standard two-stage training, training together and one-stage training without a local enhancer in the outer network. All of these approaches may appear to produce similar performances in fig. ???. However, given that decaying learning rates were used, they may suggest that two-stage training is best. However, that conclusion cannot meaningfully be drawn from that data. However, we note that the results suggest that it has similar or better performance and, because it is the standard approach, it is the best as its use will allow for easier comparison with other experiments.
- We experimented with variations of the two-stage network architecture. These included replacing the first two skip-3 residual blocks in the local enhancer with three skip-2 residual blocks, increasing the number of global enhancer channels from 768 to 1024, decreasing the smaller network input image size to 128×128; rather than the 256×256 used elsewhere, concatenating the embedded input with the inner network output; rather than using a long residual connection, and the standard pix2pixHD architecture with a long residual connection across the inner network. All our variations lowered performance relative to the pix2pixHD baseline.
- We introduced an ancillary network to resolve outputs from the inner network features. This increased the rate of convergence and final performance, as shown in fig. ??.
- Learning curves for 20×, 40× and 100× compression of coloured ImageNet images. Errors are lower for lower compression. See fig. ??.
- Reducing the adversarial loss reduces the Huber loss, as shown in fig. ???. This is expected as real images are noisy. Since there is not enough information to reproduce the noise exactly, generating noise is therefore expected to increase Euclidean distances between the real and generated images. Only the starts of two of the adversarial learning curves. One is to show that the error is higher for 100× compression. That makes sense as there is less information. The other is for a duplicate 40× compression experiment to show the variation between experiment.
- As we used batch size 1 in all our experiments, we were concerned about overfitting. This fear was heightened by performance appearing to slightly decrease when the number of global enhancer convolutions was increased from 768 to 1024. Out of curiosity, we therefore trialed replacing every convolution with a 1×1 convolution with the same number of channels, followed by the convolution outputting a single channel. This meant that the neural network had to decompress images by using stacks of convolutions to do an image-to-image translation at every layer of the original network. This increased the error, as shown in fig. ???. However, the error increase is lower than might be expected. Perhaps convolutions with a small number of channels may be suitable for mobile or other computation-bound

applications. TODO: Graph networks with fractions of the convolutions in the original network e.g. a factor of 4 or 16 fewer. Perhaps our network cannot make use of all the convolutional channels at this small batch size?

8. Learning curves for $40\times$ and $100\times$ compression are shown in fig. 21.

ELU paper: [39].

6 Coloured Image Compression

Use ImageNet[40] 2012 data for tasks 1 and 2. MSE training may be sufficient. Also perform adversarial training to compare. Do a MSE and GAN example at $100\times$ compression.

7 Performance

We compare the performance of our networks by calculating the 20000 absolute errors and structural similarity indexes[41] of restorations for each network we are presenting here. Other methods that are often used to measure GAN performance include the Inception score or Fréchet Inception Distance. However, the inception score has limitations[42]. In this context, Inception is overfit to less noisy images.

STEM noise generation is explained in [43].

Need a comparison against existing methods. Traditional methods are very good for small infills; however, they start to fail as the region to be infilled grows larger. Textures become unrealistic and they fail to capture fine detail. Nevertheless, they are decent for low frequency images e.g. image of blobs. In this paper, we will investigate fast marching and Navier-Stokes infilling. This will include the presentation of examples of where these methods do well and where they do badly. We will then present MAE and SSIM performance statistics for the methods. We will also provide the statistic.

8 Discussion

Natural statistics losses are often added to GAN generator loss functions to introduce a dependence on image content matching. However, most discriminators do not have pooling layers, making them sensitive to image translation. Differences between features for real and generated images may also vary significantly for different noise levels. Coupled with discriminator variation as they train, this makes natural statistics unstable at low momentum. This is problematic as high momentum limits GAN convergence.

In this paper, we reduce natural statistic instability by averaging natural statistics at multiple discriminator depths

across multiple discriminators. However, natural statistics are still insufficient at high compression rates as our lack of discriminator pooling layers made them sensitive to image translation. This is problematic as; heuristically, high compressions and noisy pixels make it unrealistic for a generator to consistently restore images. Ultimately, inconsistent natural gradients confused our generators, resulting in them generating cartoon-like images that have the correct overall structure and brightness; however, they lack correct fine detail like noise characteristics.

To overcome the limitations of natural statistics, we propose a modified MSE. This is calculated between Gaussian blurred images to reduce sensitivity to varying noise levels. However, it only reduces the dependence on translational variance; rather than removing it. This non-adversarial guidance performs a similar role to natural statistics; such as reducing mode collapse and improving generated image coloration, while overcoming some of its limitations in this application. We note that this is only one possible alternative to simple natural statistics and encourage others to experiment. Other variants include

- Using a structural similarity index or other method to measure difference between discriminator features to reduce dependence on different noise levels. However, this is more expensive computationally.
- Only using deep natural statistics.
- Using pooling layers in the discriminators to reduce sensitivity to image translation.

In part, some of the issues we encountered with natural statistics are a result of spectrally normalizing our discriminators. This allows discriminators to use as many features as possible when making decisions, reducing their ability to perceive features clearly differentiating $x \sim \mathbb{P}_{\text{real}}$ from $G(x)$. Overall, this is a boon as it stabilizes vanilla training; however, it prevents natural statistics from being an objective measure of perceptual difference. They are no longer strongly coupled to differentiating features. Instead, our natural statistics measure differences between features developed using as much of the input as possible.

Our architecture was developed for end-to-end training so that it would be easier for others to train. However, we expect it to be possible to improve performance by multi-stage training a network-in-network, similar to [13]. Other improvements may include (1) introducing more residual connections to reduce signal loss or (2) training with an ancillary global enhancer loss to back-propagate gradients to the global enhancer directly.

This paper focuses on super-compression, giving examples of $20\times$, $40\times$ and $100\times$ compression of TEM and STEM images. Our process restores $20\times$ compressed images almost perfectly, $40\times$ compressed images okay if

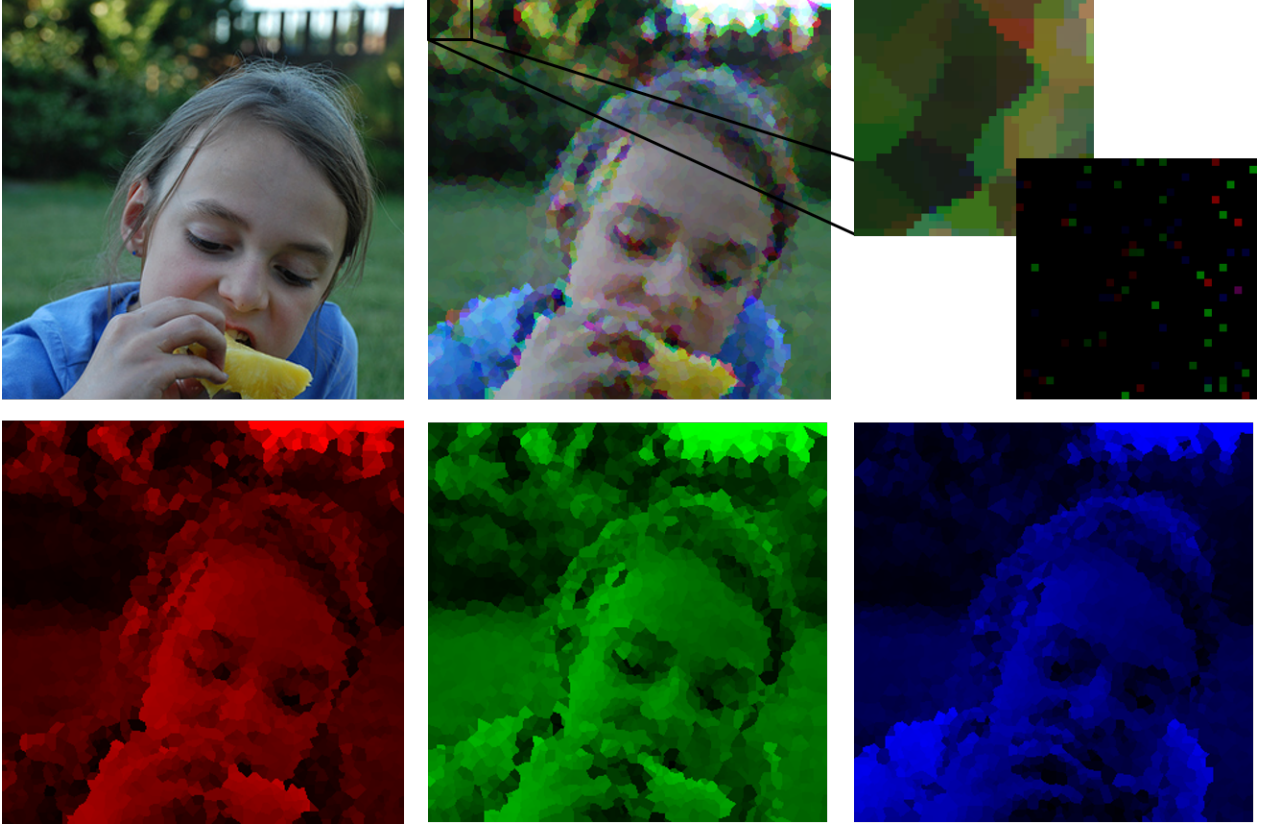


Figure 12: Red, green and blue (RGB) channels in $320 \times 320 \times 3$ colour image are independently nearest neighbour infilled. The zoom-in shows infilling around $32 \times 32 \times 3$ positioned pixels for $40\times$ compression. The original image is top-left.

they don't have many fine details and $100\times$ compressed images okay if they don't have many small details. These compression ratios many times above typical compression ratios of 10 or less. Consequently, we expect variants of our network trained for $20\times$ or lower compression to be very successful if applied to standard images.

Give more detailed explanation of encryption. Explain that tradition neural networks that produce a string of logits can be decoded more easily by training small neural networks on them. This is because logits describe high-level features. However, this is not the case for individual pixels. Instead, a significant portion of the information is implicitly encoded in positions of the marked pixels. It is not realistic to restore this information without the mask. Consider a 512×512 mask with 1 in every 32 pixels marked. There are $\frac{512^2}{32} C = 2.5 \times 10^{15829}$ possible combinations. The number that needs to be considered can be reduced as pixels will only need to be approximately locations to improve the restorations, however, there is still an extremely high number of combinations to consider. Adding to this, the information gained from changing the positions of a few pixels is likely to be insignificant compared to the noise between training runs.

The encryption is so strong because there is a huge number of meaningful arrangements for a given subset that would generate meaningful restorations. The mask is needed to constrain the reality.

For encryption, it is probably a good idea to use truly random numbers to generate the mask. That avoids hacking through trial of popular pseudo-random number sequences.

Gaussian blurring pixels positioned on the fixed mask before the first convolutions allows the first kernels to learn to convolute differentiating features, rather than how to infill a largely black image. Gaussian blurring is a linear transformation so there is no information loss.

9 Conclusions

This paper presents a high-resolution GAN for image super-compression and encryption. We demonstrate that it can realistically restore images compressed by $20\times$ to $100\times$. To achieve these high compression ratios, we use multiple discriminators acting on different scales with a novel non-adversarial loss that replaces natural statistics. We further improve training by (1) introducing new policies to balance generator and discriminator learning in

the context of multi-scale discrimination, (2) developing improved learning rate, momentum and batch normalization decay policies to accelerate training and (3) spectral normalizing our discriminators. Our networks have been designed to be easy to use by making them end-to-end trainable. Additionally, we have made our code and twelve models pre-trained for electron microscopy available at [github-repo-loc](#).

References

- [1] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural networks*, vol. 61, pp. 85–117, 2015.
- [2] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, “cudnn: Efficient primitives for deep learning,” *arXiv preprint arXiv:1410.0759*, 2014.
- [3] M. T. McCann, K. H. Jin, and M. Unser, “A review of convolutional neural networks for inverse problems in imaging,” *arXiv preprint arXiv:1710.04011*, 2017.
- [4] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, “A survey of deep neural network architectures and their applications,” *Neurocomputing*, vol. 234, pp. 11–26, 2017.
- [5] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, “Revisiting distributed synchronous sgd,” *arXiv preprint arXiv:1604.00981*, 2016.
- [6] 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, pp. 265–283, 2016.
- [7] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
- [8] J. M. Ede, “Autoencoders, kernels, and multilayer perceptrons for electron micrograph restoration and compression,” *arXiv preprint arXiv:1808.09916*, 2018.
- [9] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, “Fast and accurate image super-resolution with deep laplacian pyramid networks,” *IEEE transactions on pattern analysis and machine intelligence*, 2018.
- [10] J. Kim, J. Kwon Lee, and K. Mu Lee, “Deeply-recursive convolutional network for image super-resolution,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1637–1645, 2016.
- [11] 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*, 2017.
- [12] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, “Autoencoding beyond pixels using a learned similarity metric,” *arXiv preprint arXiv:1512.09300*, 2015.
- [13] 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,” *arXiv preprint arXiv:1711.11585*, 2017.
- [14] S. v. d. Walt, S. C. Colbert, and G. Varoquaux, “The numpy array: a structure for efficient numerical computation,” *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011.
- [15] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” *arXiv preprint*, 2017.
- [16] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.
- [17] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.
- [18] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” *arXiv preprint arXiv:1802.02611*, 2018.
- [19] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” *arXiv preprint*, 2016.
- [20] S. Xiang and H. Li, “On the effects of batch and weight normalization in generative adversarial networks,” *arXiv preprint arXiv:1704.03971*, 2017.
- [21] 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*, pp. 901–909, 2016.
- [22] E. Hoffer, R. Banner, I. Golan, and D. Soudry, “Norm matters: efficient and accurate normalization schemes in deep networks,” *arXiv preprint arXiv:1803.01814*, 2018.

- [23] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, p. 3, 2013.
- [24] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.
- [25] A. Aitken, C. Ledig, L. Theis, J. Caballero, Z. Wang, and W. Shi, "Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize," *arXiv preprint arXiv:1707.02937*, 2017.
- [26] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," 2010.
- [27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [28] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *arXiv preprint arXiv:1802.05957*, 2018.
- [29] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.
- [30] H. You, Z. Jiao, H. Xu, J. Li, Y. Wang, and X. Gao, "Restricting greed in training of generative adversarial network," *arXiv preprint arXiv:1711.10152*, 2017.
- [31] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?(no, it is not about internal covariate shift)," *arXiv preprint arXiv:1805.11604*, 2018.
- [32] J. Kukačka, V. Golkov, and D. Cremers, "Regularization for deep learning: A taxonomy," *arXiv preprint arXiv:1710.10686*, 2017.
- [33] D. Pfau and O. Vinyals, "Connecting generative adversarial networks and actor-critic methods," *arXiv preprint arXiv:1610.01945*, 2016.
- [34] e. a. Shrivastava, A., "Learning from simulated and unsupervised images through adversarial training," *arXiv preprint arXiv: 161207828*, 2016.
- [35] e. a. Schaul, Tom, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.
- [36] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.
- [37] M. Bertalmio, A. L. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. I–I, IEEE, 2001.
- [38] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online; accessed ;today;].
- [39] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.
- [40] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.
- [41] 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*, vol. 13, no. 4, pp. 600–612, 2004.
- [42] S. Barratt and R. Sharma, "A note on the inception score," *arXiv preprint arXiv:1801.01973*, 2018.
- [43] F. Timischl, M. Date, and S. Nemoto, "A statistical model of signal-noise in scanning electron microscopy," *Scanning*, vol. 34, no. 3, pp. 137–144, 2012.
- [44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [45] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, p. 5, 2018.

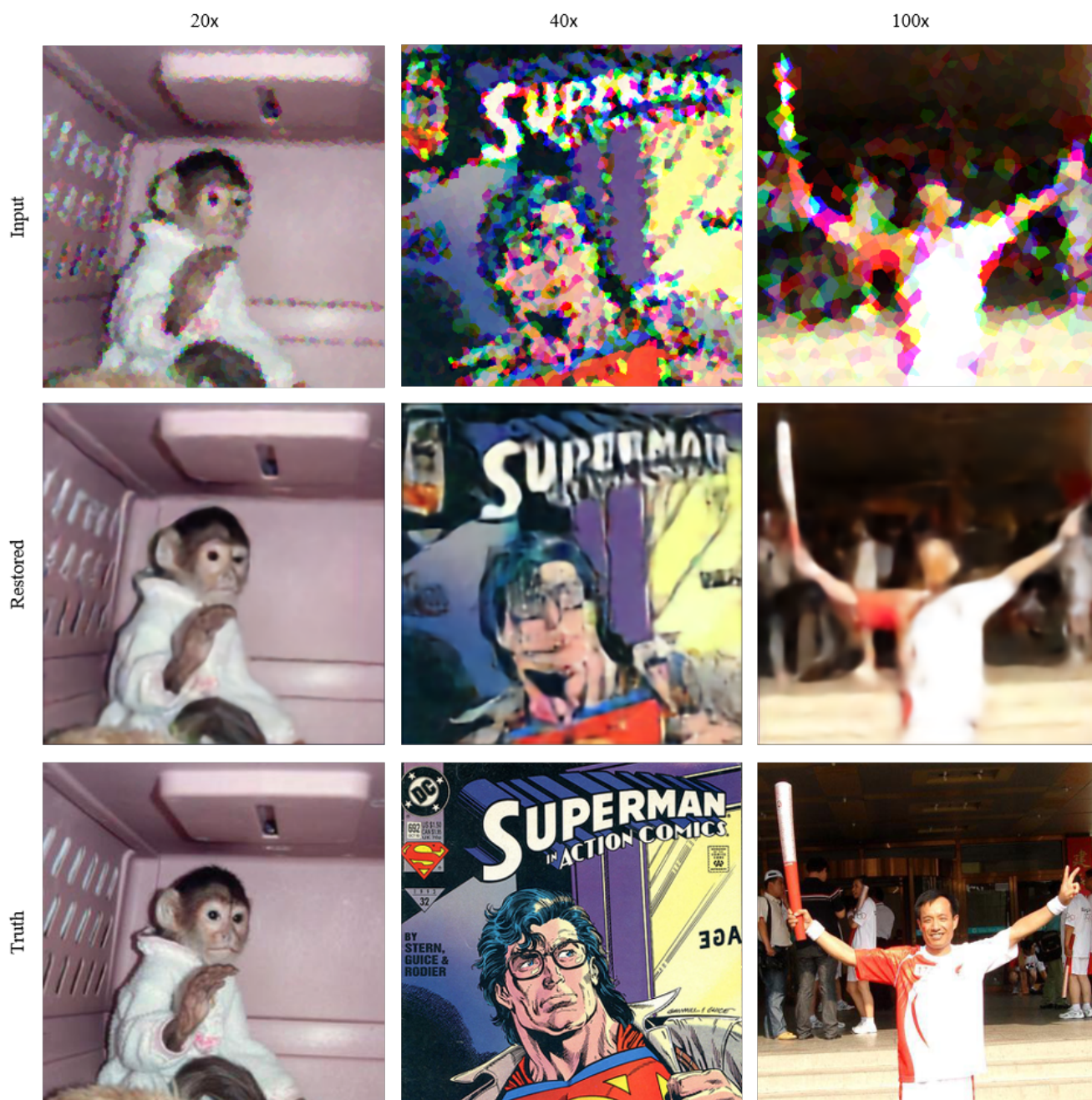


Figure 13: Coloured images generated by 20 \times , 40 \times and 100 \times compression networks non-adversarially trained on ImageNet. Artefacts become apparent by 40 \times compression e.g. a person is hallucinated in the top-left of the superman restoration.

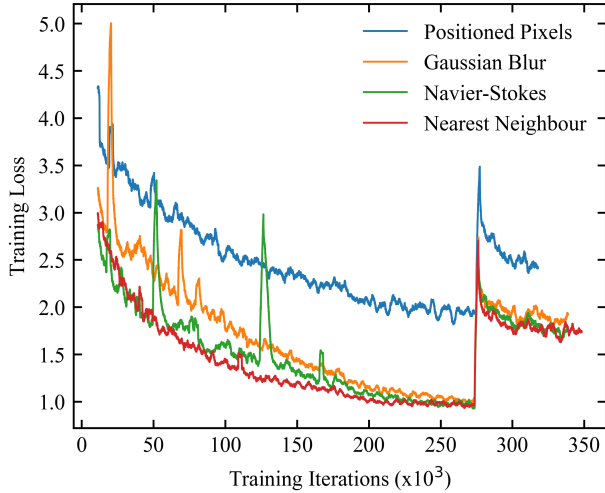


Figure 14: Errors are lowered by infilling blank regions after positioning known pixels. Final Performance is highest for Navier-Stokes infilling, closely followed by $k=1$ nearest neighbour infilling. Gaussian blurring with an 11×11 kernel has lower performance.

10 Additional Experiments

Learning curves for other experiments will be presented and discussed in this section. All learning curves are 2500 iteration boxcar averaged.

Constant filling of unknown input pixels does not fully utilize the first layers of the generator. It also requires the generator to learn the difference between variable and non-variable inputs. To increase the number of variable input pixels, we applied an 11×11 , standard deviation TODO Gaussian kernel, Navier-Stokes infilled and 1 nearest neighbour infilled positioned pixels. As shown by fig. 14, all infilling methods improve performance. Final performance is highest for Navier-Stokes infilling. However, nearest neighbour infilling has similar final performance, more stable learning and is much faster. As a result, we nearest neighbour infill positioned pixels.

Performance for architecture variations is presented in fig. 16. Residual connections make betwixt convolutions perform perturbative transformations[44]. To investigate more perturbative stages, we replaced the last 2 skip-3 residual blocks in the local enhancer with 3 skip-2 residual blocks. However, the effect on performance is not significant. Performance is also similar if 768 channel global enhancer convolutions with 1024 channel. Halving the inner network input size and removing one of the inner network’s strided encoding convolutions increases error and destabilizes training. In contrast, residually connecting; rather than concatenating, the inner network output, similar to [45], decreases error and stabilizes training. As a result,

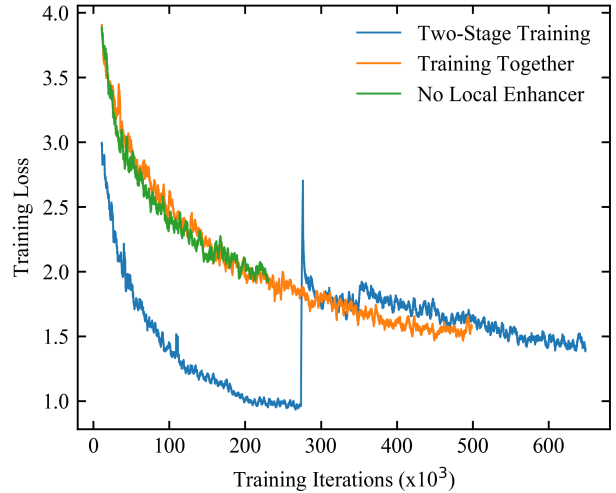


Figure 15: Training the inner and outer network together, two-stage training where the the inner and outer network are trained separately then fine-tuned together and training together with no local enhancer.

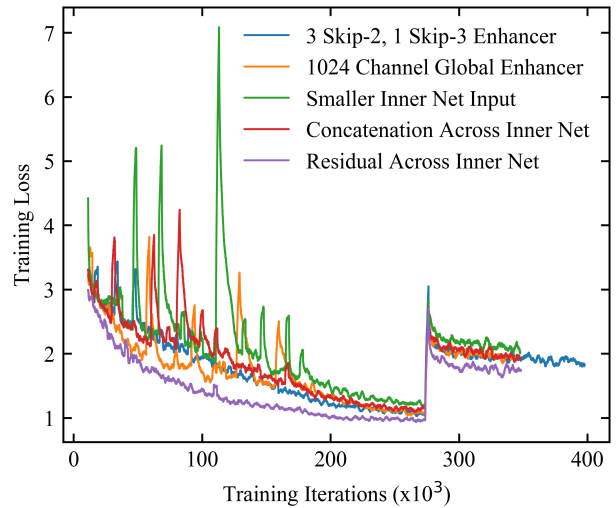


Figure 16: Errors for architecture variations. Performance is highest for two-stage training with a residual connection across the inner network. Similar decreases in performance are seen if the first two skip-3 residual blocks in the local enhancer are replaced with 3 skip-2 blocks, global enhancer channels are increased from 768 to 1024 or the long residual is replaced with a concatenation. The lowest performance is for images input to the inner network being downsampled by a factor of 4 along each dimension; rather than 2.

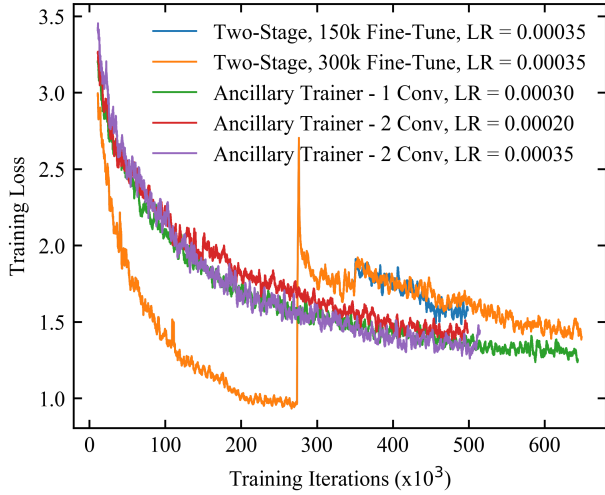


Figure 17: Auxiliary network guided inner network training outperforms two-stage training with fine-tuning. Auxiliary network guided performance is higher for a learning rate (LR) of 0.0003 than for a learning rate of 0.0002.

we replaced concatenation with addition.

To demonstrate that our network can be applied to, we applied it to For comparison, learning curves for different compression ratios are presented in [?]. Error are higher for higher compression.

To investigate the effect of single channel image bottlenecks in our architecture, we replace each convolution with a 1×1 convolution with the same number of channels, followed by a convolution with the original kernel size and 1 output channel. As shown by fig. 20, this substantially decreases performance.

To justify the number of convolutions, we show learning curves for 1/1, 1/2, 1/4, 1/8, 1/16 and 1/32 times the number of channels outputted by every convolution. Increasing the number of channels decreases the error. However, the decrease in error decreases as the number of convolutions is increased. becoming small and starts to plateau as the number of convolutions used in our architecture is approached. Nevertheless, the difference has

Examples of training with and without an error spike are shown in fig. 24

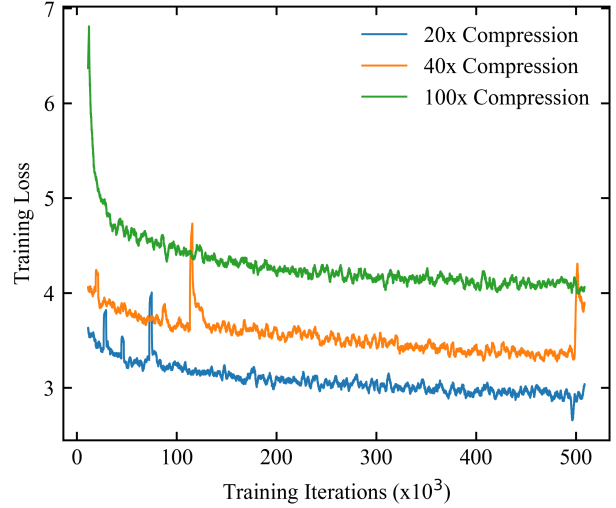


Figure 18: Learning curves for 20 \times , 40 \times and 100 \times compression of coloured ImageNet images. Errors are higher for higher compression.

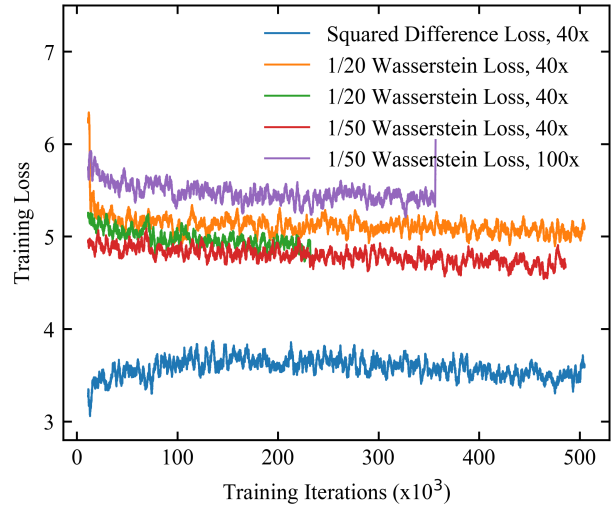


Figure 19: Learning curves for adversarial fine-tuning of 40 compression networks. Errors are lower for squared difference losses than for gradient-penalized energy-based Wasserstein losses. The Wasserstein losses were multiplied by 1/20 or 1/50 before addition to the generator loss.

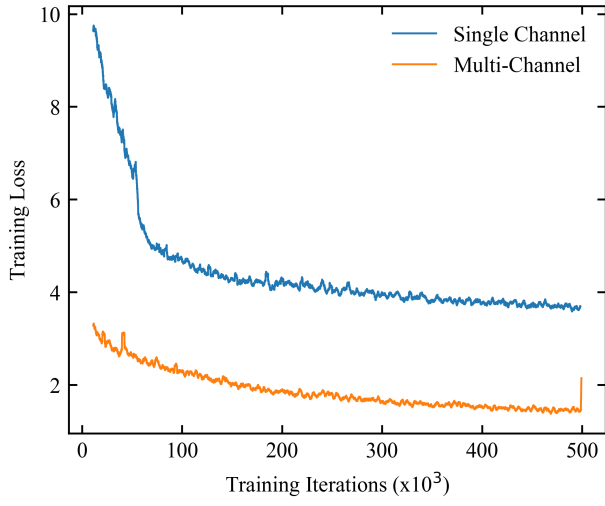


Figure 20: Effect of bottlenecking the architecture by replacing every convolution with 1×1 convolution then a 3×3 convolution to 1 output channel.

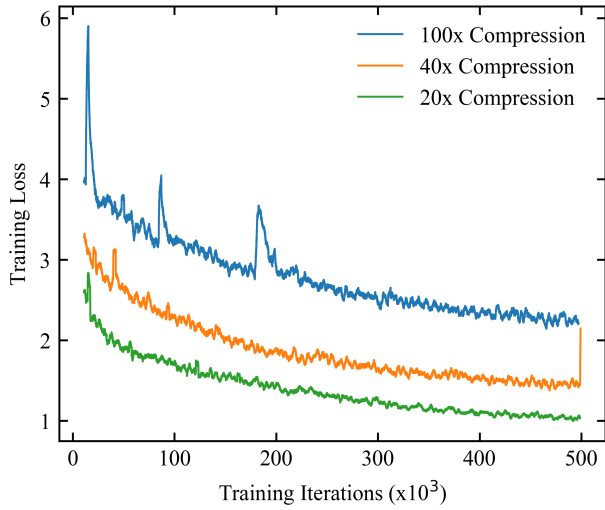


Figure 21: Learning curves for $20\times$, $40\times$ and $100\times$ compression. Errors are lower for lower compression.

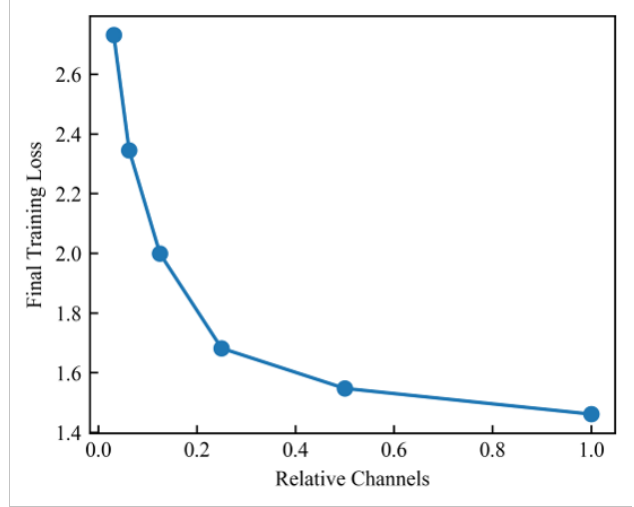
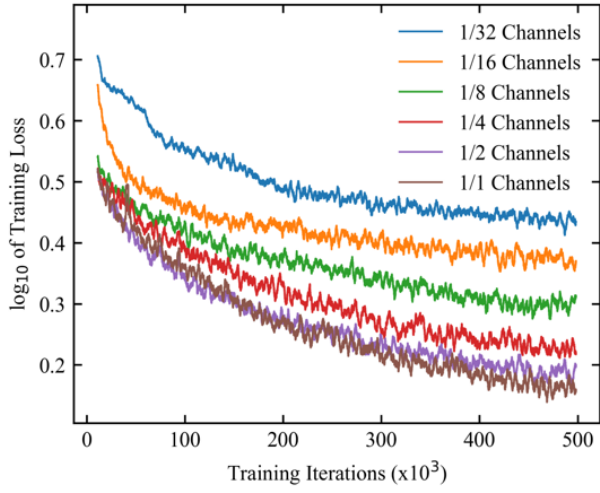


Figure 22: Learning curves for networks with 1/1, 1/2, 1/4, 1/8, 1/16 and 1/32 times the number of channels outputted by every convolution. A line graph for the mean training losses for the final 20000 iterations shows that errors decrease as the number of channels is increased. The decrease in error per channels increase decreases as the number of channels increases and begins to plateau near the number of channels in our network (relative channels = 1).

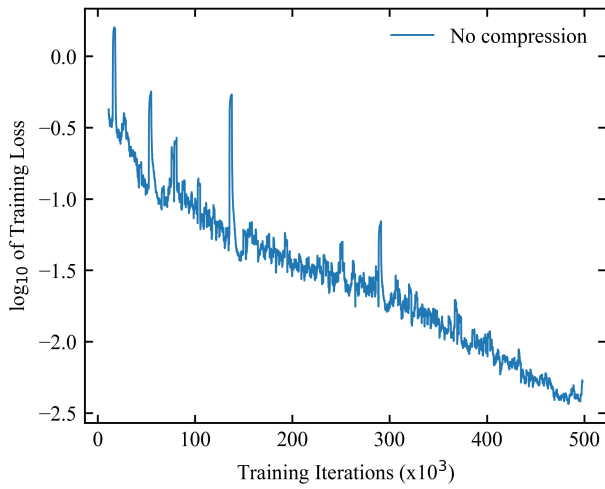


Figure 23: Learning curve for no compression showing the error decreases to near-zero.

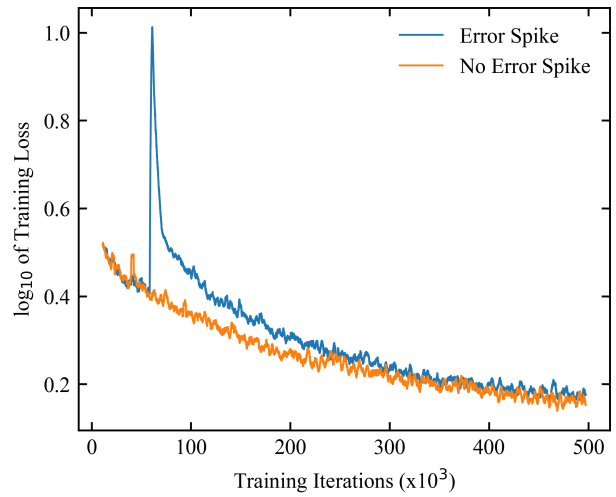


Figure 24: Typical learning curves 40 \times compression with and without an error spike. Convergence is reduced by the error spike.

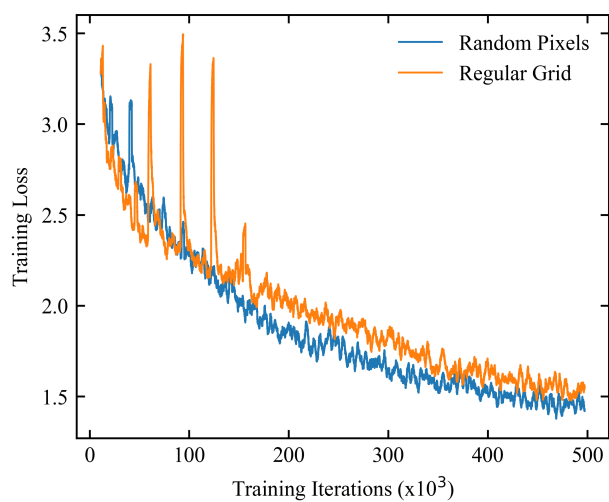


Figure 25: Learning curves for $40\times$ compression using fixed random and regular-grid pixel subsets. The final error is lower for a fixed random pixel subset.

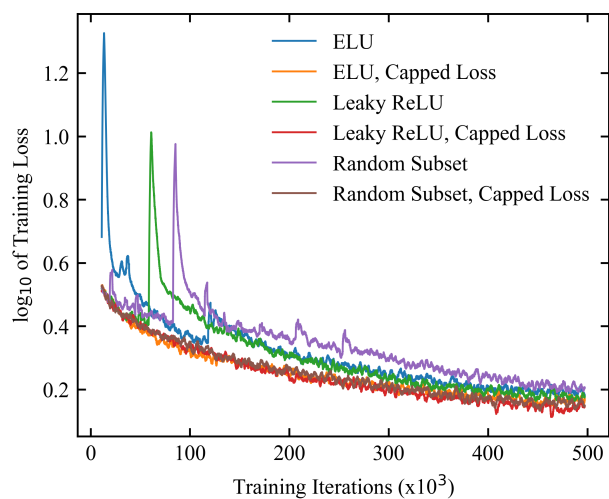


Figure 26: Learning curves for $40\times$ compression... clipping improves performance!

11 Appendix

Algorithm 1 Two-sided dynamic relaxation. Default settings used in our experiments are $\beta_{\text{real}} = 0.99$, $\beta_{\text{fake}} = 0.99$, $\Delta\gamma_{\text{real}}^{\downarrow} = 0.00020$, $\Delta\gamma_{\text{real}}^{\uparrow} = 0.00025$, $\Delta\gamma_{\text{fake}}^{\downarrow} = 0.00020$, $\Delta\gamma_{\text{fake}}^{\uparrow} = 0.00025$, $\gamma_{\text{real},\min} = 0$, $\gamma_{\text{fake},\min} = 0$, $u_{\text{real}} = 0.9$, $u_{\text{fake}} = 0.1$. In practice, this relaxation had little effect on training and is only reported for completeness.

Require: $\beta_{\text{real}}, \beta_{\text{fake}} \in [0, 1]$ Decay rates of prediction moving means

Require: $u_{\text{real}}, u_{\text{fake}} \in [0, 1]$, $u_{\text{real}} > u_{\text{fake}}$ Targets to relax the discriminator to meet

Require: n_D Number of discriminators

Require: $\Delta\gamma_{\text{real}}^{\downarrow}, \Delta\gamma_{\text{real}}^{\uparrow}, \Delta\gamma_{\text{fake}}^{\downarrow}, \Delta\gamma_{\text{fake}}^{\uparrow} > 0$ Relaxation adjustment steps

Require: $\gamma_{\text{real},\min}, \gamma_{\text{fake},\min} \geq 0$ Limits on relaxation

Require: $D_i \in [0, 1]$, $i \in \mathbb{N}_{n_D}$ Predictive discriminators

Require: G Generator

$\mu_{\text{real}} \leftarrow 0.5$ Initialize running mean for real examples

$\mu_{\text{fake}} \leftarrow 0.5$ Initialize running mean for fake examples

$\gamma_{\text{real},\max} \leftarrow 1/u_{\text{real}} - 1$ Maximum relaxation needed for extreme greed for real examples

$\gamma_{\text{fake},\max} \leftarrow u_{\text{fake}}/(1 - u_{\text{fake}})$ Maximum relaxation needed for extreme greed for fake examples

while Generator not converged **do**

 Sample $x \sim \mathbb{P}_{\text{real}}$ a batch of real data

 Generate $G(x)$ a batch of fake data

 Optimize generator and discriminator for 1 training step

$\mu_{\text{real}} \leftarrow \beta_{\text{real}}\mu_{\text{real}} + \frac{1 - \beta_{\text{real}}}{n_D} \sum_i^{n_D} D_i(x)$

$\mu_{\text{fake}} \leftarrow \beta_{\text{fake}}\mu_{\text{fake}} + \frac{1 - \beta_{\text{fake}}}{n_D} \sum_i^{n_D} D_i(G(x))$

if $\mu_{\text{real}} > u_{\text{real}}$ **then**

$\gamma_{\text{real}} \leftarrow \min(\gamma_{\text{real}} + \Delta\gamma_{\text{real}}^{\uparrow}, \gamma_{\text{real},\max})$

else if $\mu_{\text{real}} < u_{\text{real}}$ **then**

$\gamma_{\text{real}} \leftarrow \max(\gamma_{\text{real}} - \Delta\gamma_{\text{real}}^{\downarrow}, \gamma_{\text{real},\min})$

end if

if $\mu_{\text{fake}} < u_{\text{fake}}$ **then**

$\gamma_{\text{fake}} \leftarrow \min(\gamma_{\text{fake}} + \Delta\gamma_{\text{fake}}^{\uparrow}, \gamma_{\text{fake},\max})$

else if $\mu_{\text{fake}} > u_{\text{fake}}$ **then**

$\gamma_{\text{fake}} \leftarrow \max(\gamma_{\text{fake}} - \Delta\gamma_{\text{fake}}^{\downarrow}, \gamma_{\text{fake},\min})$

end if

end while

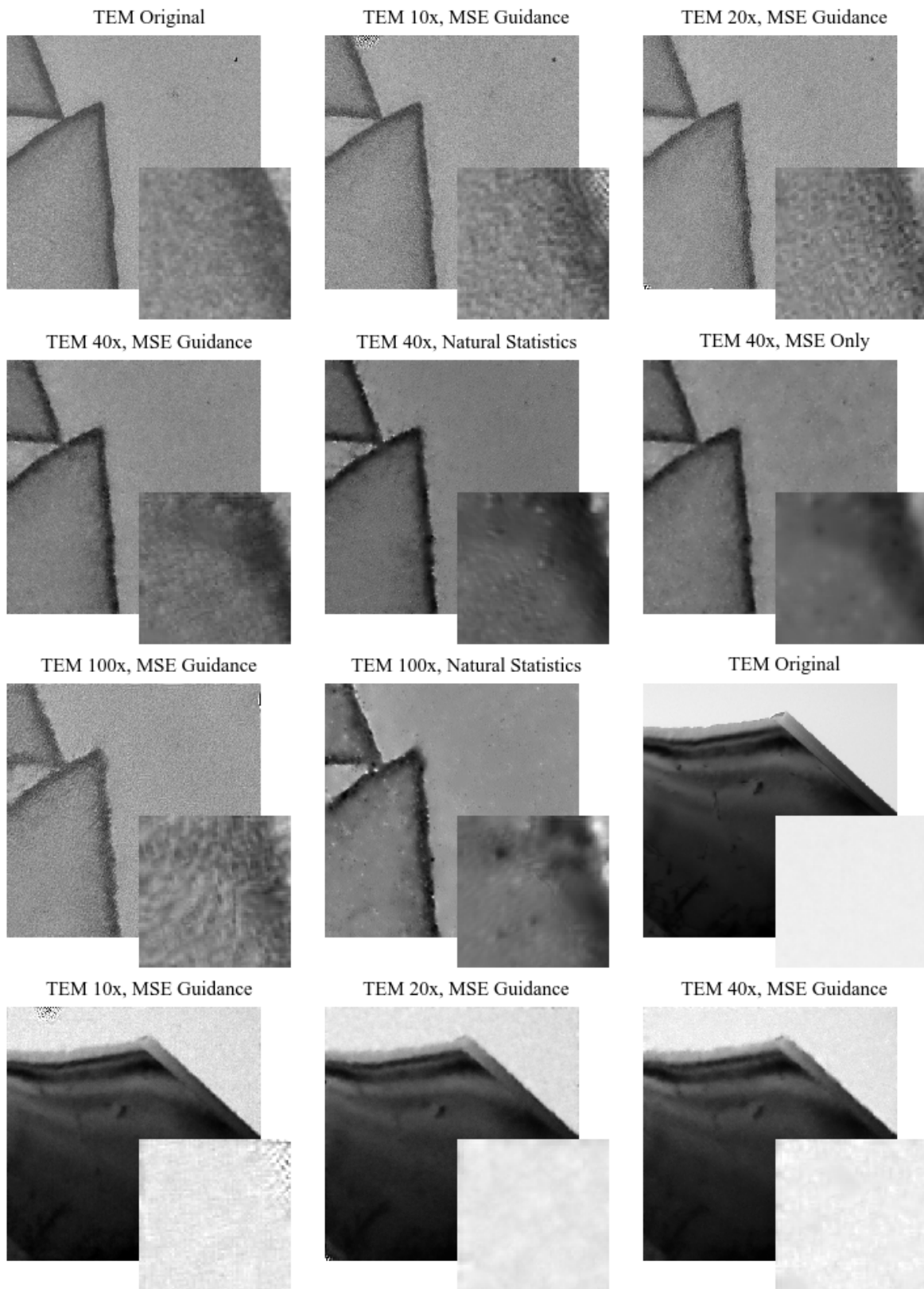


Figure 27: Juxtaposition of 512×512 images that have been compressed and restored using our process. Enlarged 64×64 regions from the top-left of each image are inset to ease comparison.

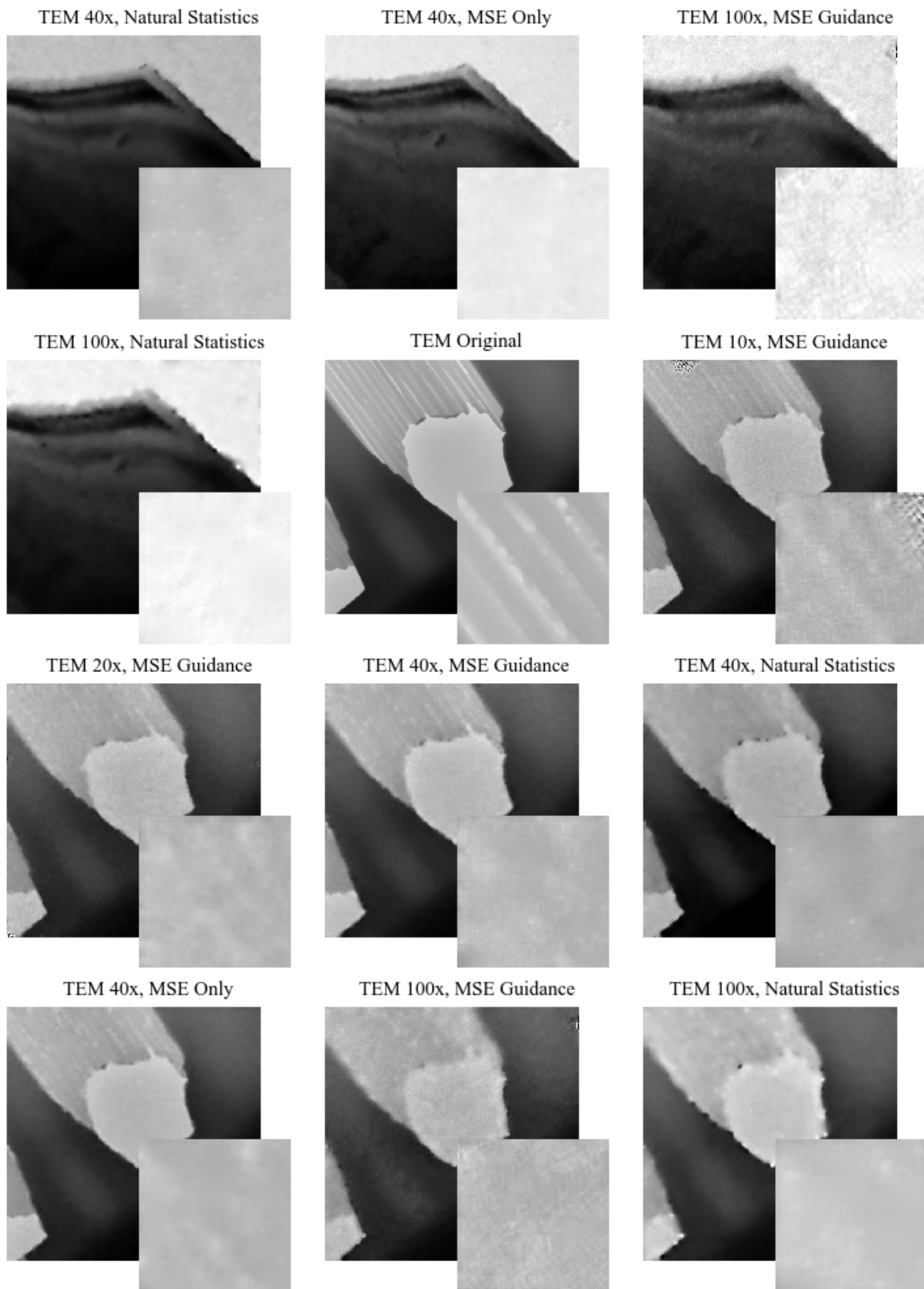


Figure 28: Continued juxtaposition of 512×512 images that have been compressed and restored using our process. Enlarged 64×64 regions from the top-left of each image are inset to ease comparison.

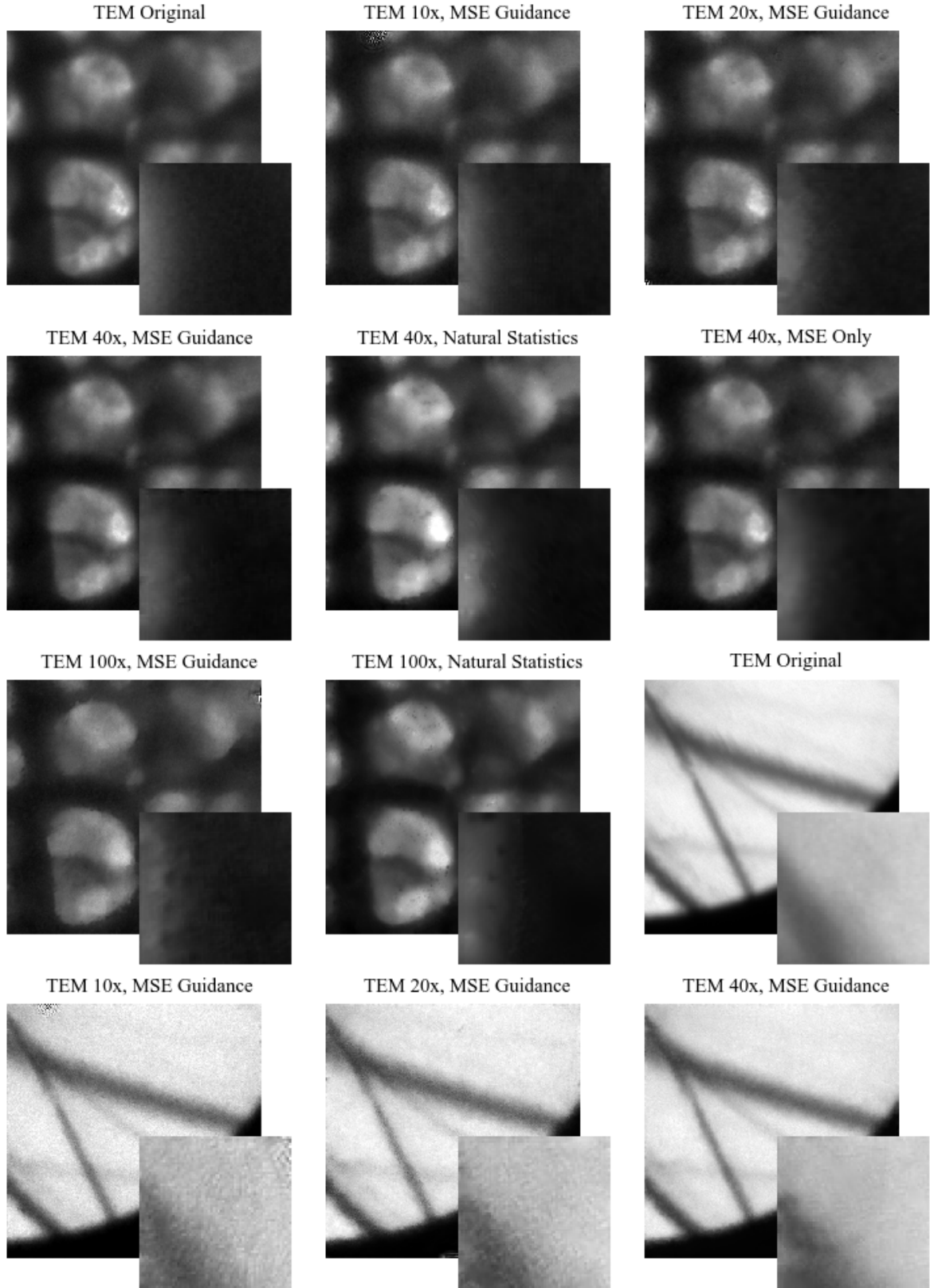


Figure 29: Continued juxtaposition of 512×512 images that have been compressed and restored using our process. Enlarged 64×64 regions from the top-left of each image are inset to ease comparison.

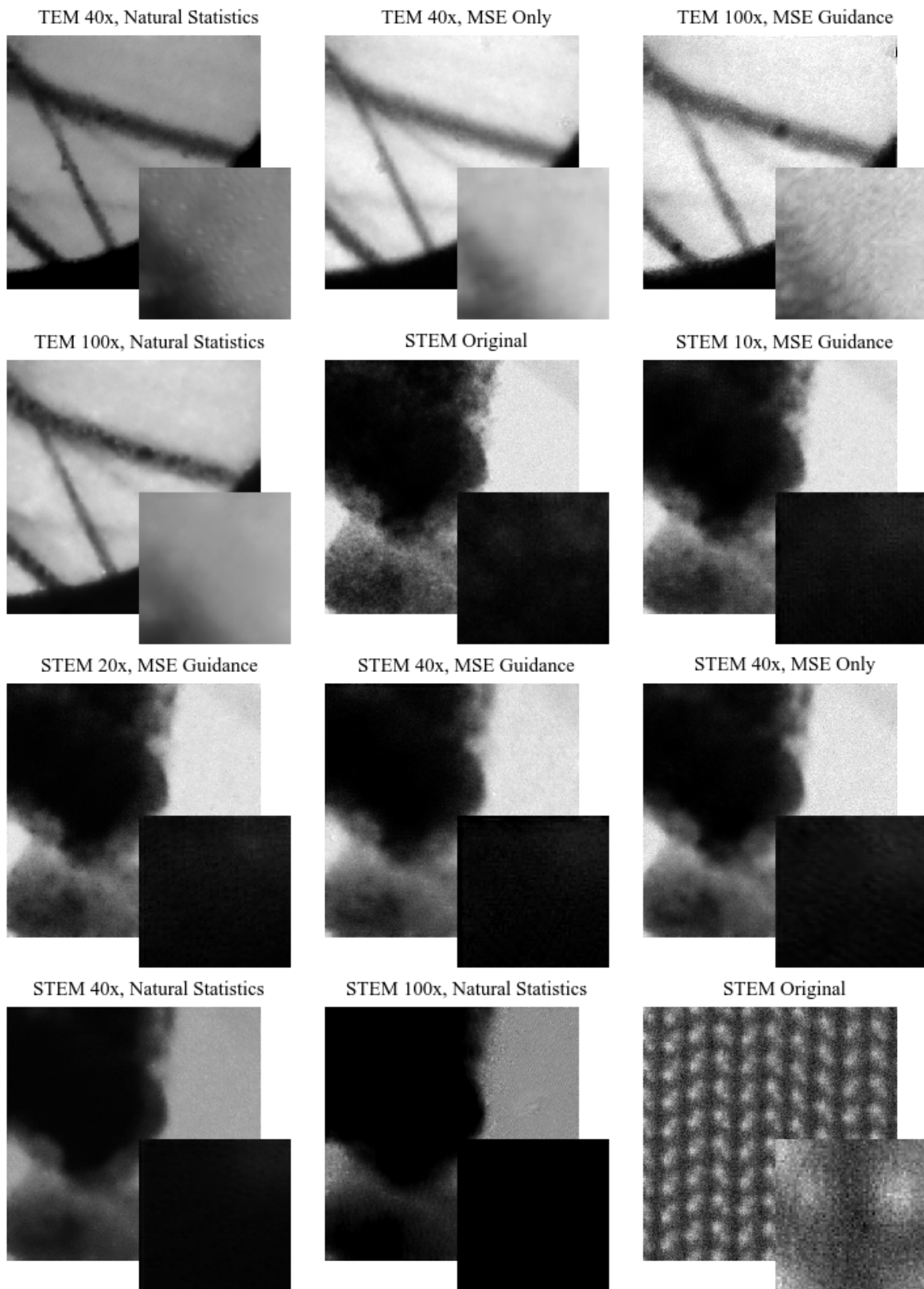


Figure 30: Continued juxtaposition of 512×512 images that have been compressed and restored using our process. Enlarged 64×64 regions from the top-left of each image are inset to ease comparison.

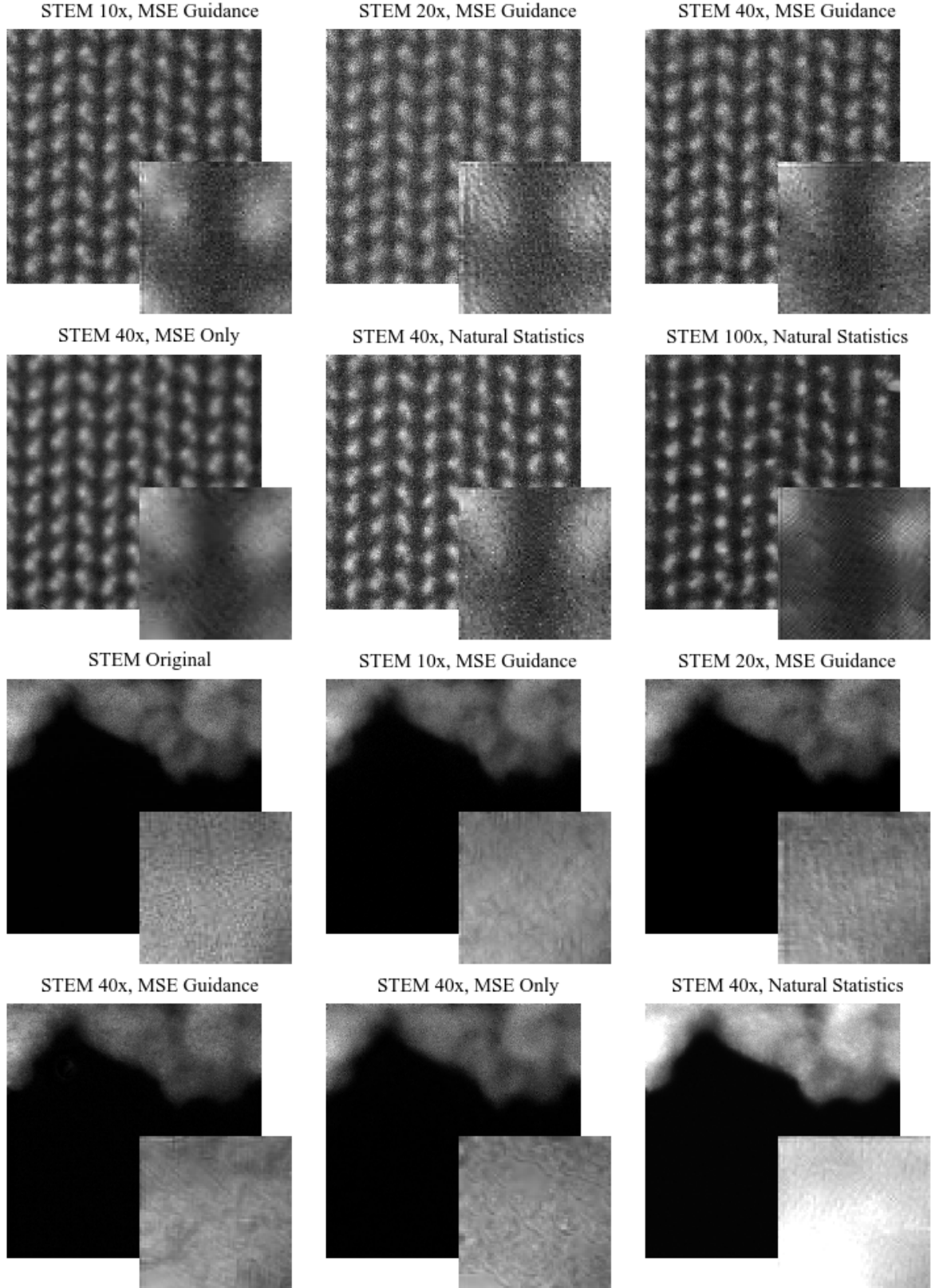


Figure 31: Continued juxtaposition of 512×512 images that have been compressed and restored using our process. Enlarged 64×64 regions from the top-left of each image are inset to ease comparison.

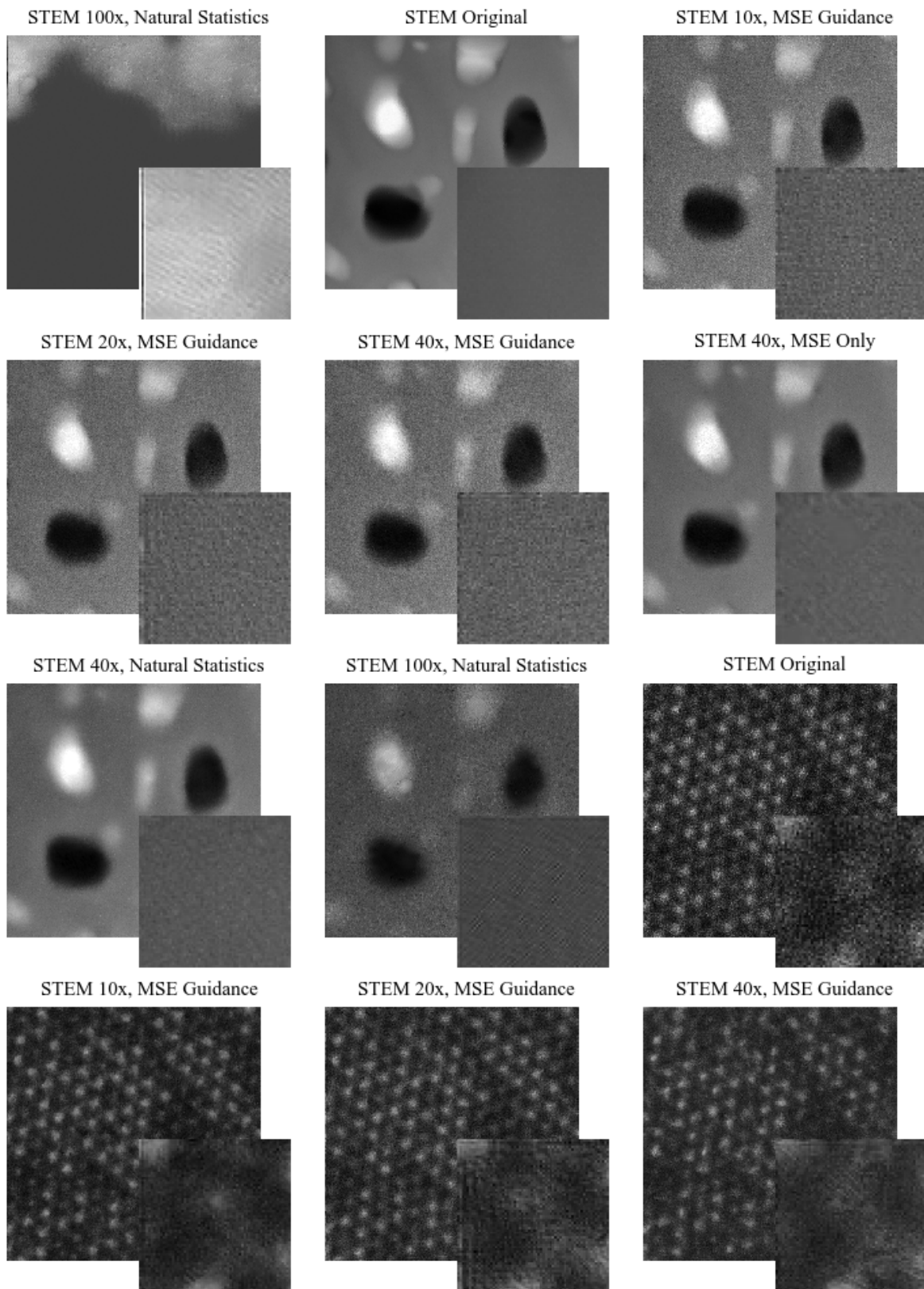


Figure 32: Continued juxtaposition of 512×512 images that have been compressed and restored using our process. Enlarged 64×64 regions from the top-left of each image are inset to ease comparison.

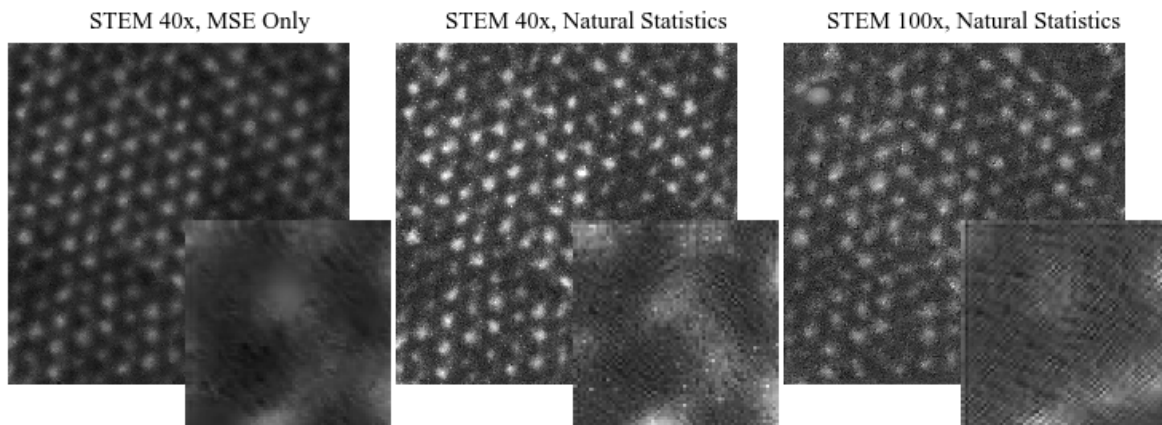


Figure 33: Continued juxtaposition of 512×512 images that have been compressed and restored using our process. Enlarged 64×64 regions from the top-left of each image are inset to ease comparison.