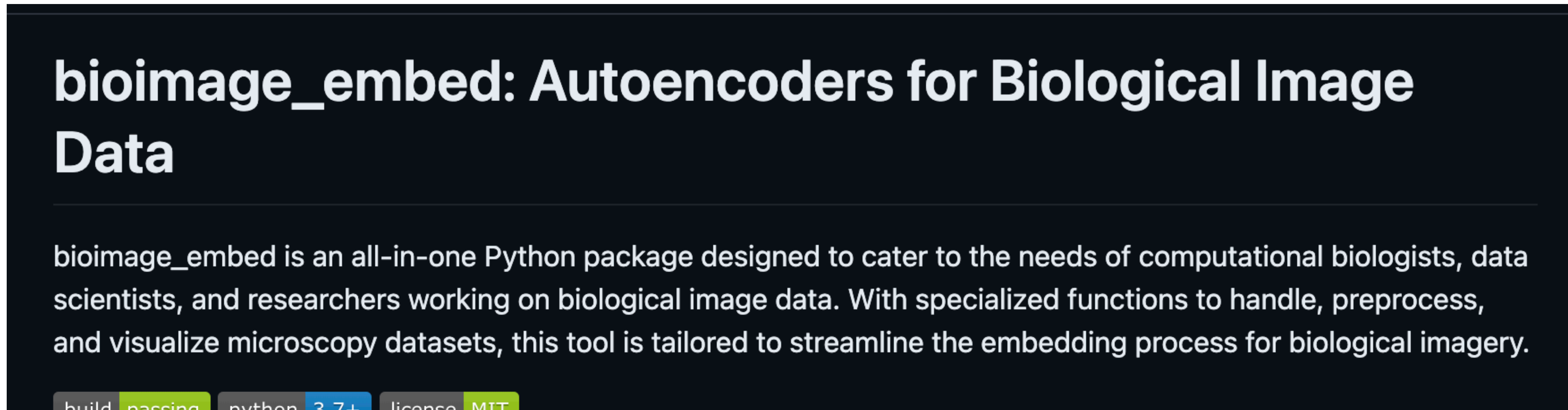
EMBL-EBI ML Ops Solution Accelerator Transformation Journey +

Representation learning for the BioImageArchive

Craig Russell



bioimage_embed: Autoencoders for Biological Image Data

bioimage_embed is an all-in-one Python package designed to cater to the needs of computational biologists, data scientists, and researchers working on biological image data. With specialized functions to handle, preprocess, and visualize microscopy datasets, this tool is tailored to streamline the embedding process for biological imagery.

Outline and Aims of BioImage Embed

Goal: Provide a model that can featurise bioimages

Outline

- BIA
- Autoencoders
    - AE,VAE,VQVAE,MAE
- Pretraining
    - Scale
- BioImage Embed
- More biological applications

Aims

- Large scale pretraining on EBI data
- Model fine tuning
- Flexible configuration with a BioImage Focus
- HPC and Cloud native
- Automated MLOps
    - Model selection, validation

Partnership with EMBL-EBI | Cast of Characters in Collaboration

Santiago Insua
Hybrid Cloud Manager

Craig Russell
Data Scientist

David Gomez
Hybrid DevOps Engineer

Matthew Hartley
Team Leader

C.D. Tiwari
Architect

EMBL-EBI **ML Ops Solution Accelerator Transformation Journey** +

Project Oversight

Raymond Hounon
Account Director

Hariprasad
Customer Engineer

Hatem Nawar
Customer Engineer

+

Adam Hammond
Regional Sales Lead - EMEA

Pallavi Satsangi
Delivery Head - Custom AI/ML

Delivery Team

Project Management

Saicharan Gurramkonda
Engagement Manager

Machine Learning

Samit Saxena
Technical Architect - Machine Learning

Jay Mangi
Snr. Machine Learning Engineer

Sakshi Garg
Machine Learning Engineer

Savi Bhide
Machine Learning Engineer

Saranya J
Machine Learning Engineer

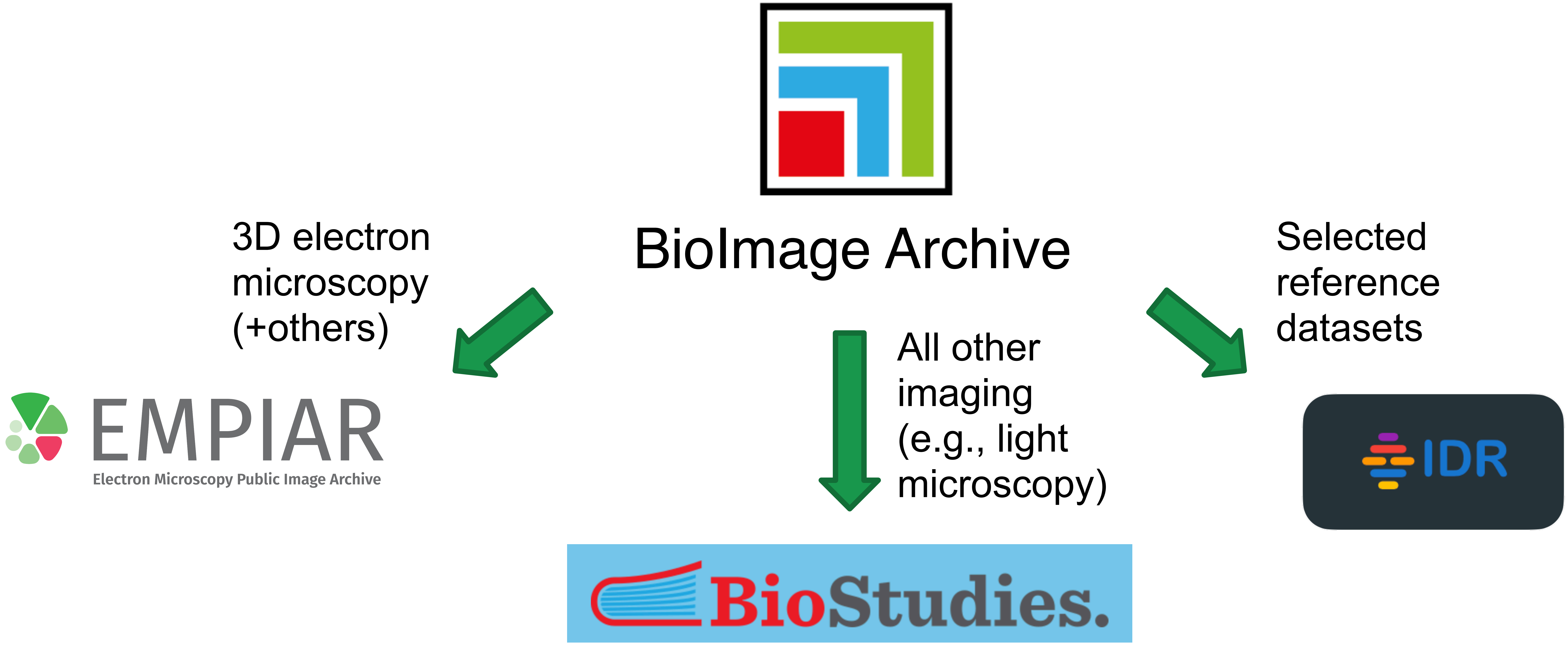
06
Weeks

1
Project Phase

10+
Dedicated professionals

BioImageArchive

Summer 2019: Launch



BioImage Archive

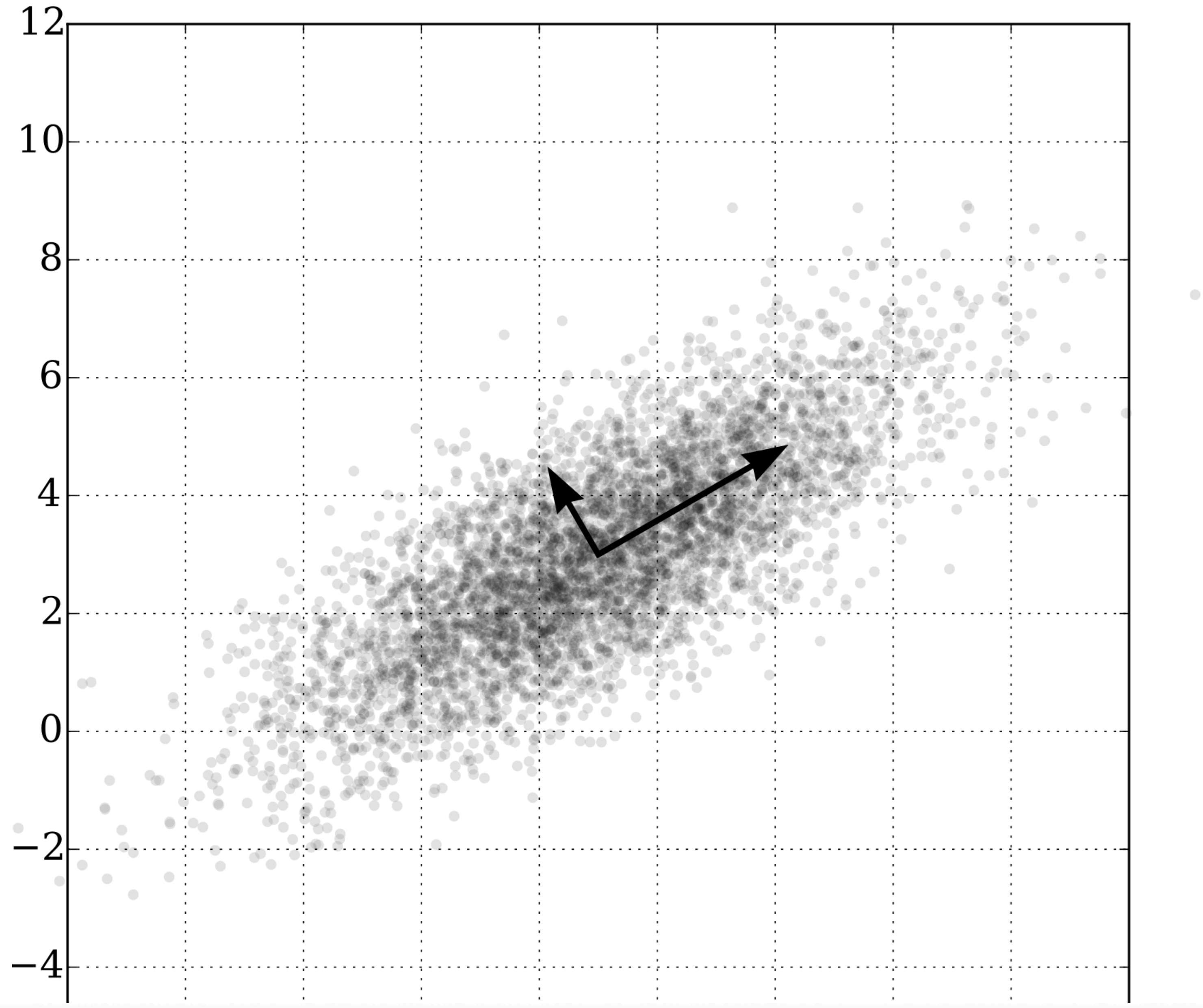
3D electron microscopy (+others)

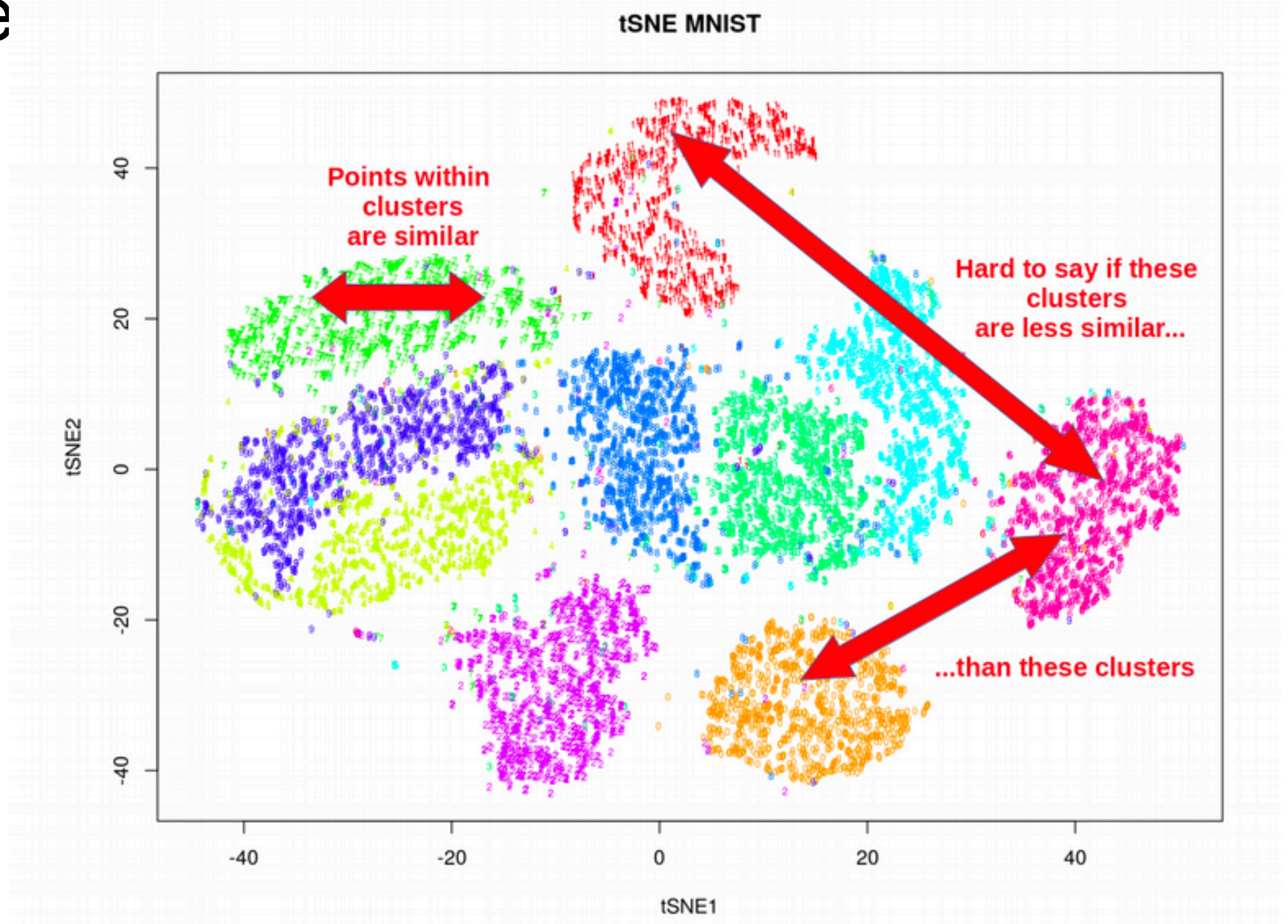
All other imaging (e.g., light microscopy)

Selected reference datasets

Dimensionality reduction

- We can represent the salient parts of data using di
  techniques
- **PCA** is a classical approach whereby eigenve
  axes of maximal variance
  - Rotation in higher dimensional space
- **UMAP** and **t-SNE** both use machine learning
  (manifold)

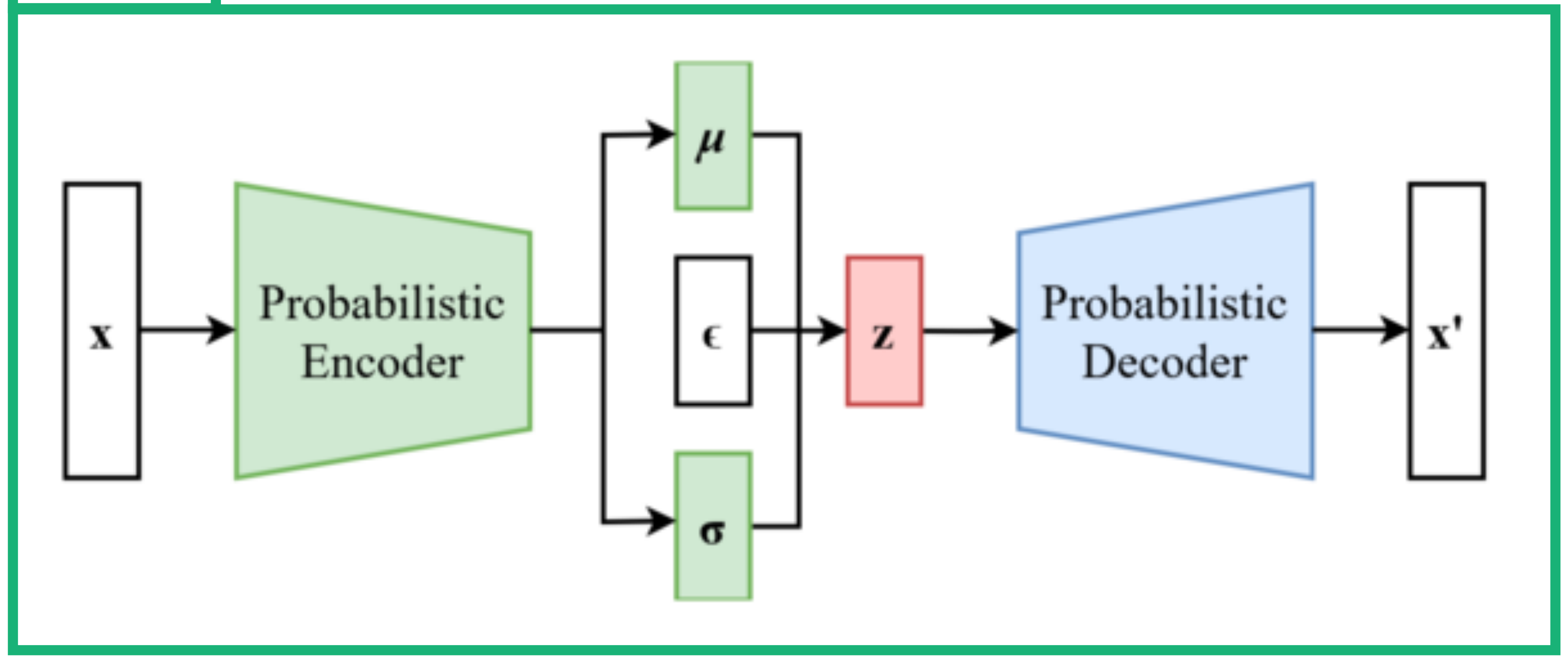




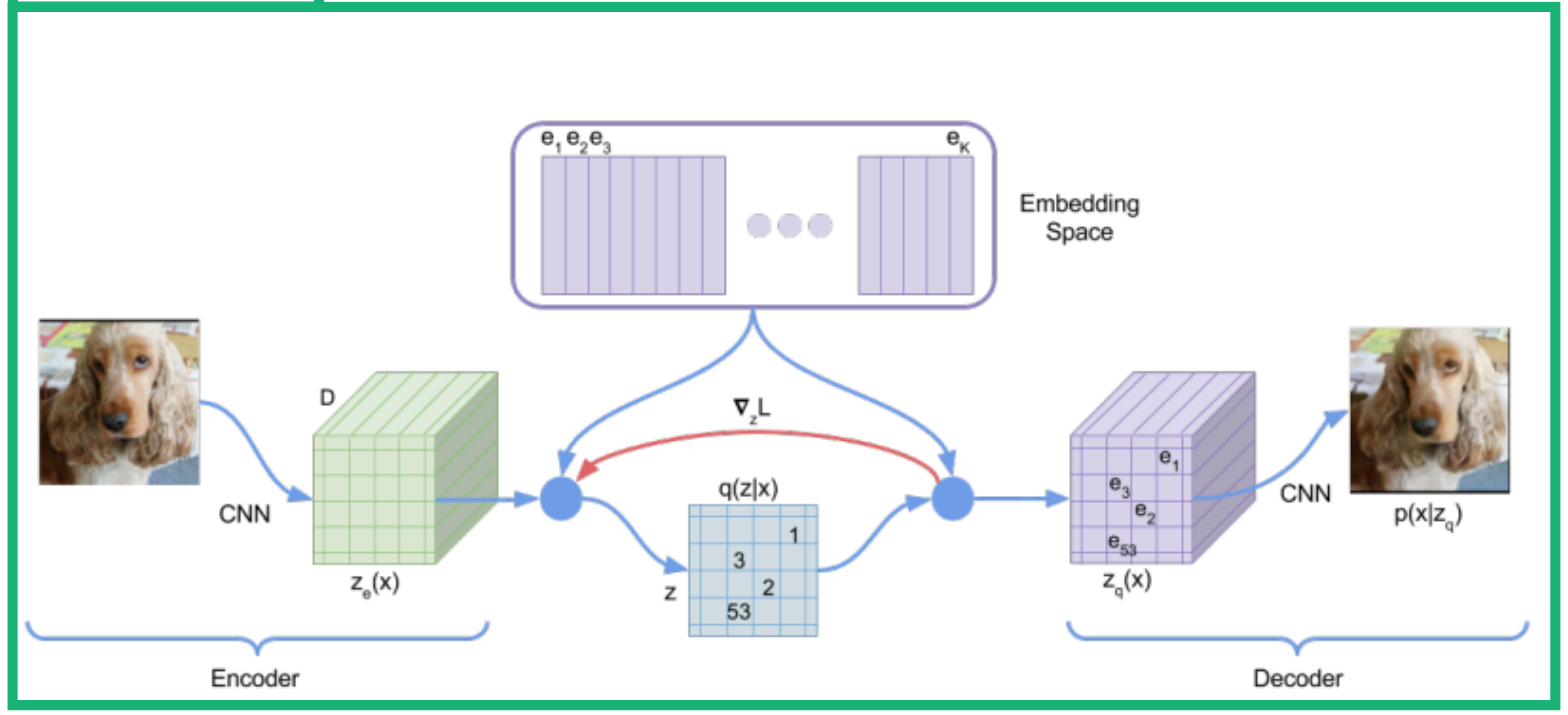
Flavours of AutoEncoders

- **AutoEncoders'** latent space is usually
  - **Sparse**
  - **Unstructured**
  - **Nonsense between points (posterior collapse)**
- **Variational AutoEncoders (VAE)**
  - add a gaussian prior to latent (embedding) space
    - No longer **sparse** or **unstructured**
- **Vector quantised VAEs (vqVAE)** ✓
  - discretise the latent space
    - No longer **sparse** or **unstructured**
    - Helps with **posterior collapse**

VAE



vqVAE



https://arxiv.org/abs/1312.6114

Masked Autoencoders are Scalable Learners of Cellular Morphology

Oren Kraus* Kian Kenyon-Dean* Saber Saberian Maryam Fallah Peter McLean

Jess Leung Vasudev Sharma Ayla Khan Jia Balakrishnan Safiye Celik

Maciej Sypetkowski Chi Vicky Cheng Kristen Morse Maureen Makes

Ben Mabey Berton Earnshaw

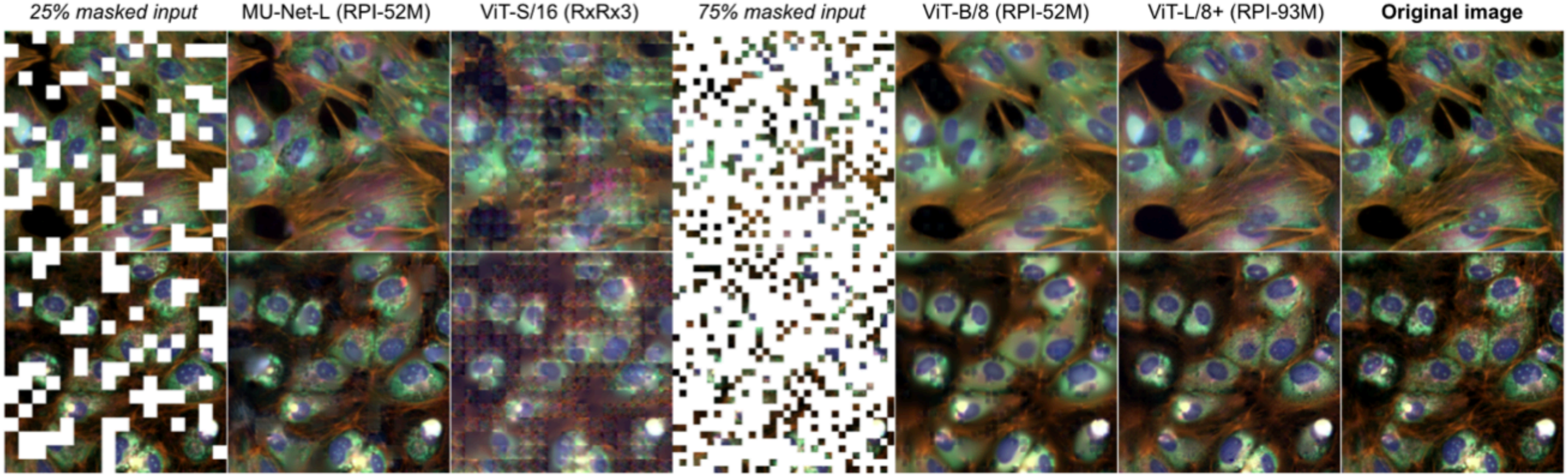
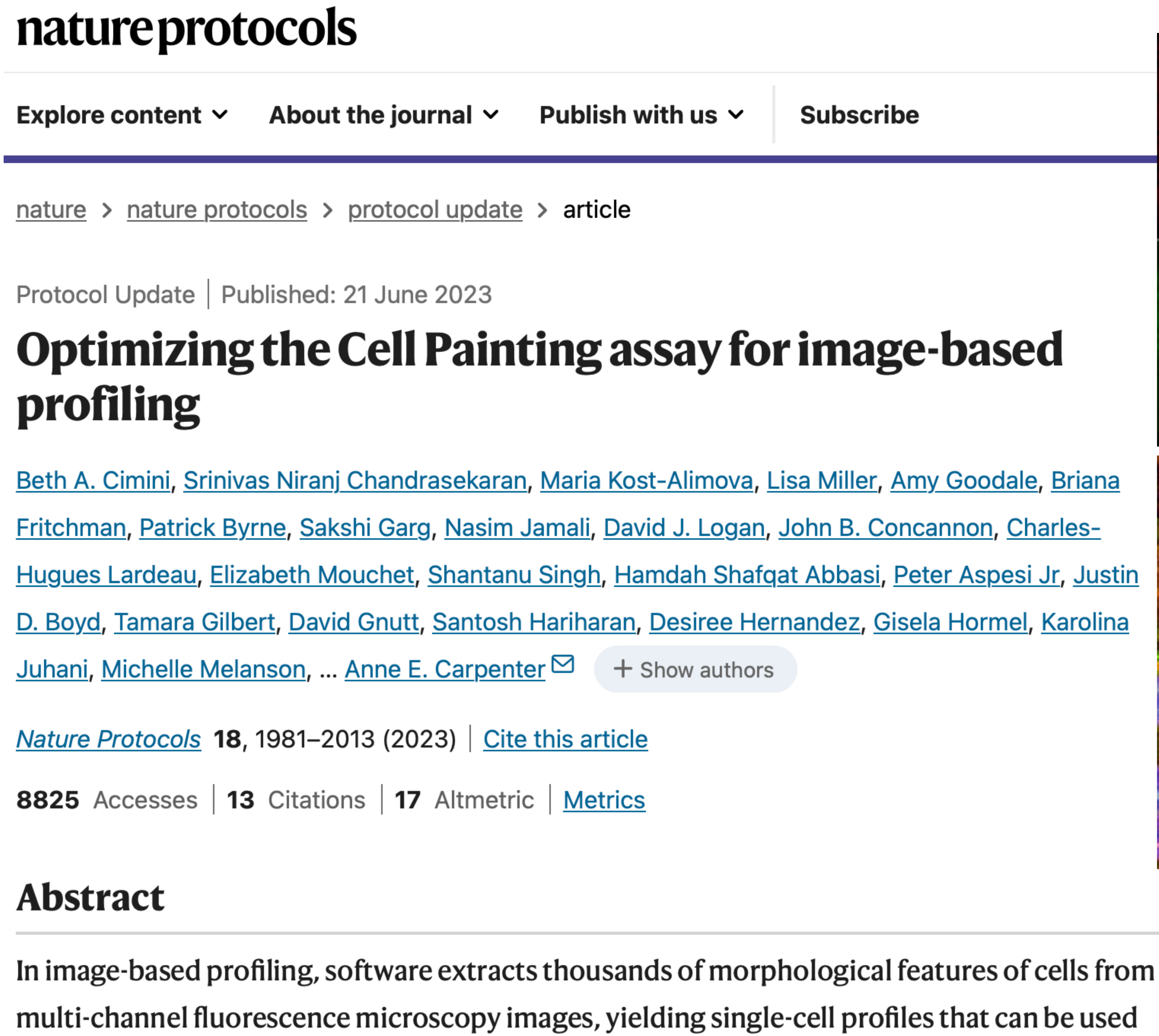


Figure 1: Visualizing reconstructions from masked random *validation* images for different MAEs.



nature protocols

Explore content ⌄ About the journal ⌄ Publish with us ⌄ Subscribe

nature › nature protocols › protocol update › article

Protocol Update | Published: 21 June 2023

Optimizing the Cell Painting assay for image-based profiling

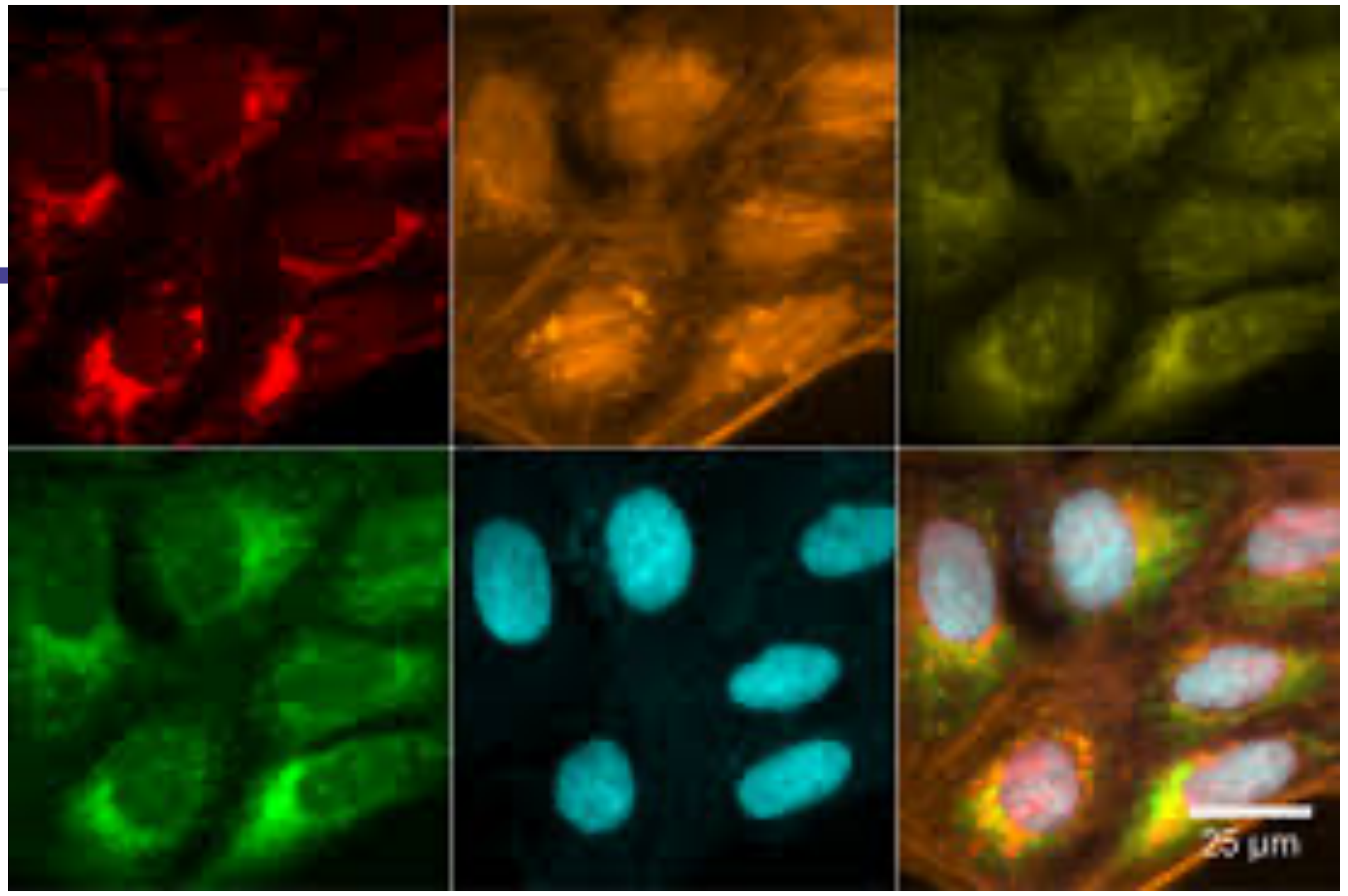
Beth A. Cimini, Srinivas Niranj Chandrasekaran, Maria Kost-Alimova, Lisa Miller, Amy Goodale, Briana Fritchman, Patrick Byrne, Sakshi Garg, Nasim Jamali, David J. Logan, John B. Concannon, Charles-Hugues Lardeau, Elizabeth Mouchet, Shantanu Singh, Hamdah Shafqat Abbasi, Peter Aspesi Jr, Justin D. Boyd, Tamara Gilbert, David Gnutt, Santosh Hariharan, Desiree Hernandez, Gisela Hormel, Karolina Juhani, Michelle Melanson, … Anne E. Carpenter ✉ + Show authors

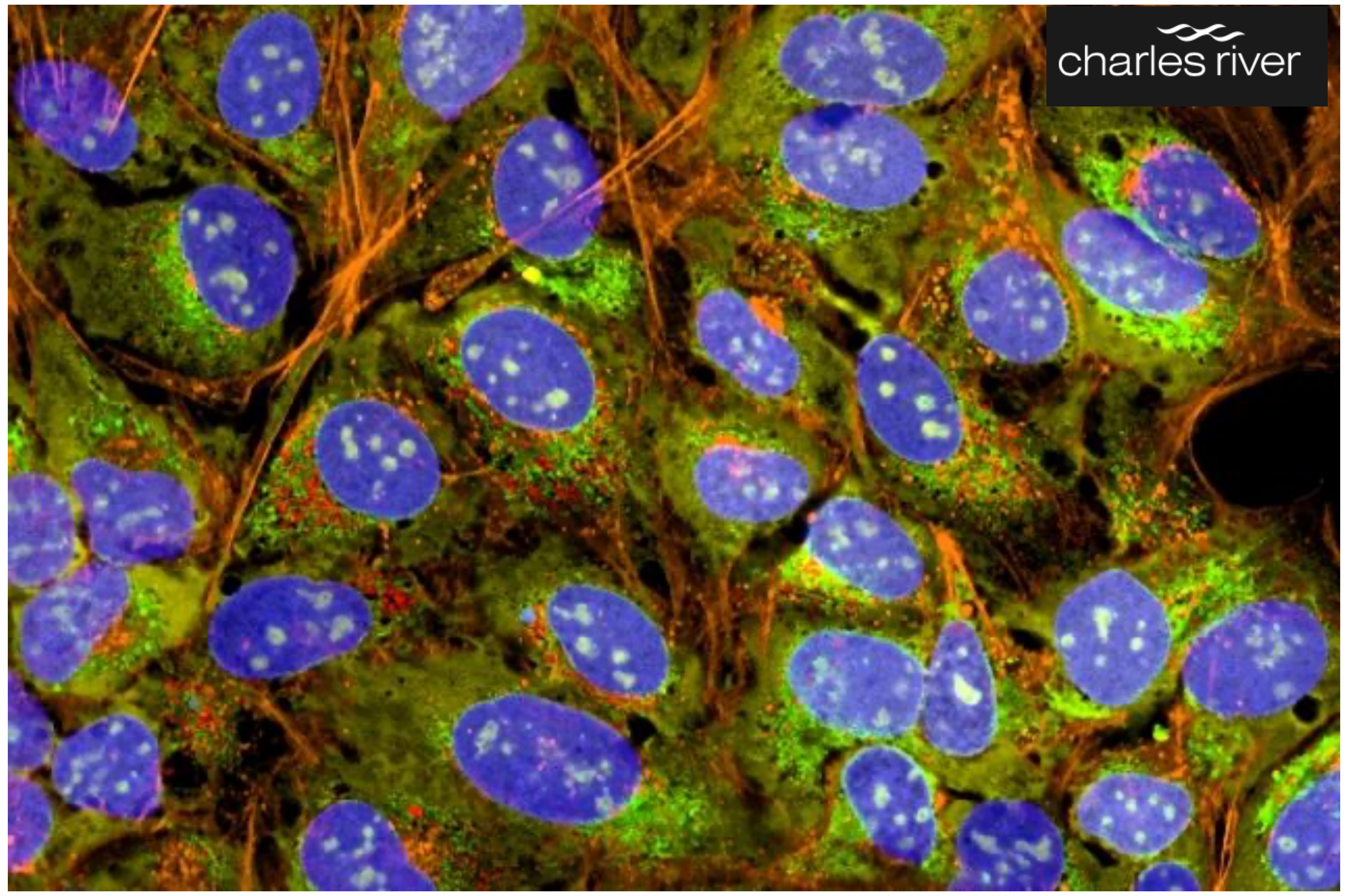
Nature Protocols **18**, 1981–2013 (2023) | Cite this article

8825 Accesses | **13** Citations | **17** Altmetric | Metrics

Abstract

In image-based profiling, software extracts thousands of morphological features of cells from multi-channel fluorescence microscopy images, yielding single-cell profiles that can be used





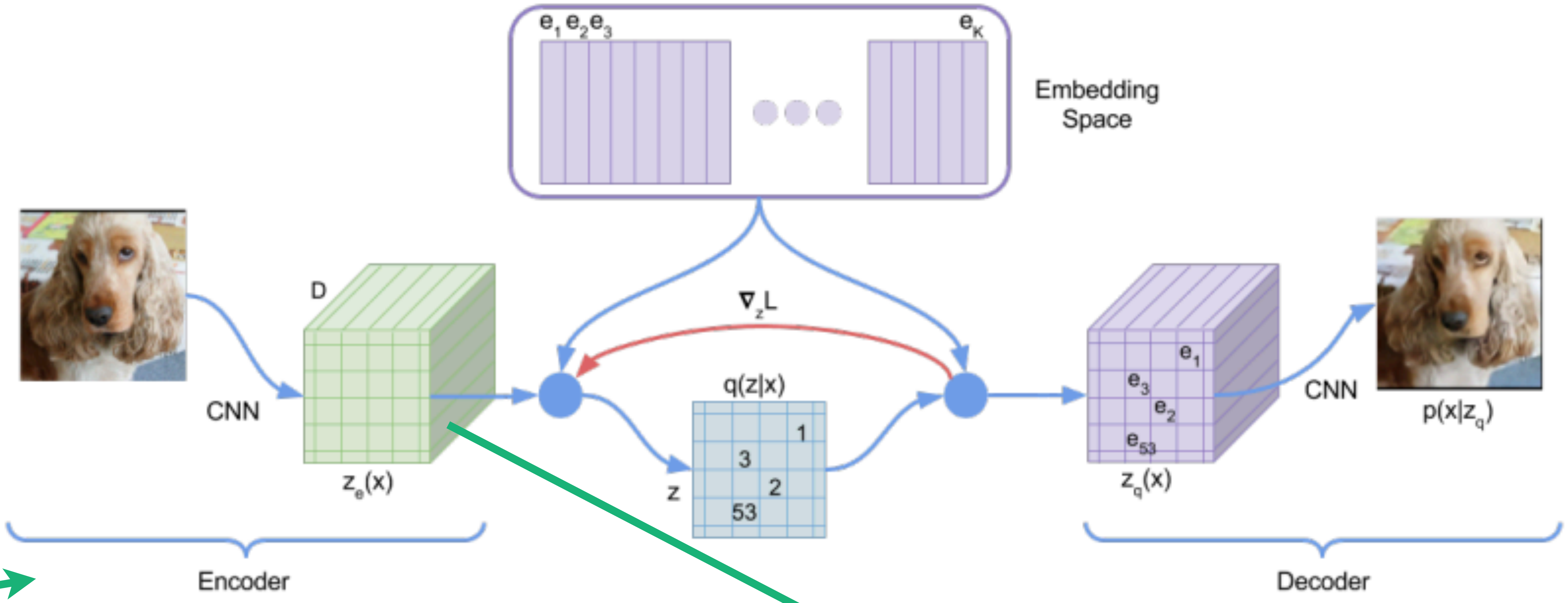
JUMP-Cell Painting Consortium
Joint Undertaking in Morphological Profiling

Masked Autoencoders for Microscopy are Scalable Learners of Cellular Biology

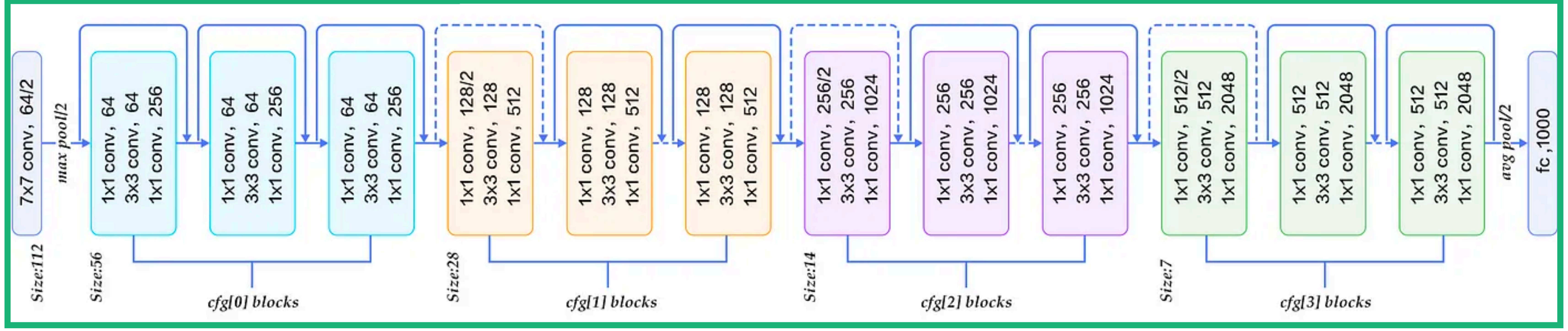
Oren Kraus[1] Kian Kenyon-Dean[1] Saber Saberian[1] Maryam Fallah[1] Peter McLean[1]
Jess Leung[1] Vasudev Sharma[1] Ayla Khan[1] Jia Balakrishnan[1] Safiye Celik[1]
Dominique Beaini[2] Maciej Sypetkowski[2] Chi Vicky Cheng[1] Kristen Morse[1]
Maureen Makes[1] Ben Mabey[1] Berton Earnshaw[1,2]
[1]Recursion [2]Valence Labs

Vector quantised variational AutoEncoder (vqVAE)

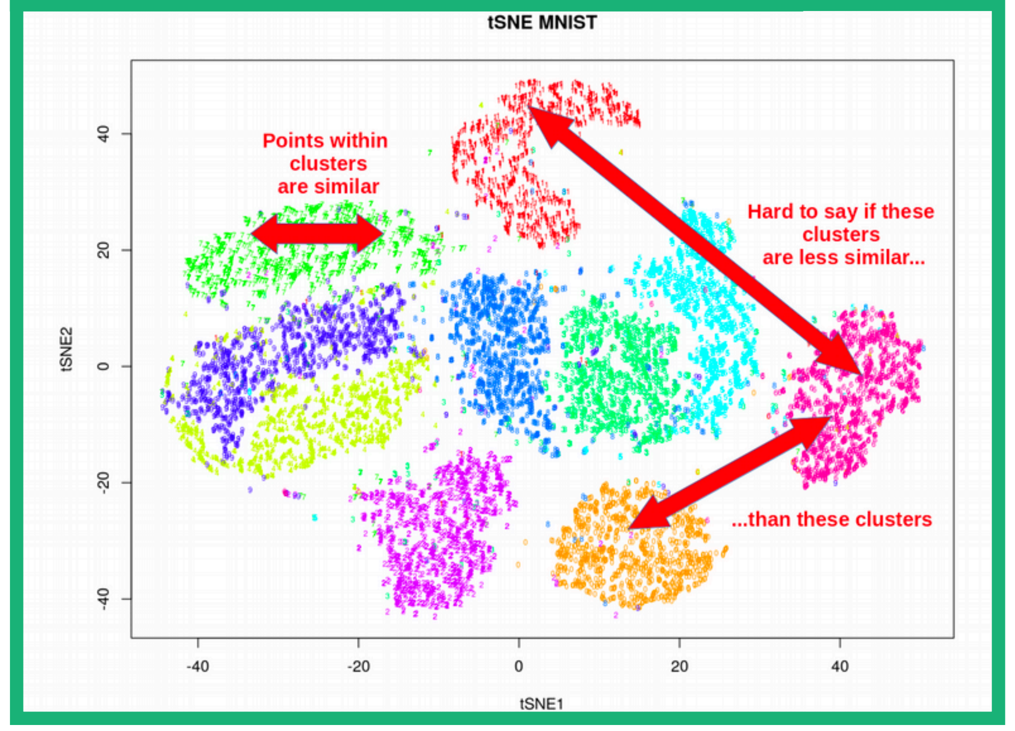
- ResNet-50 encoder-decoder pair
    - Deep convolutional neural network with “residual” skip connections
- ~ 2 million parameters



CNN: ResNet-50



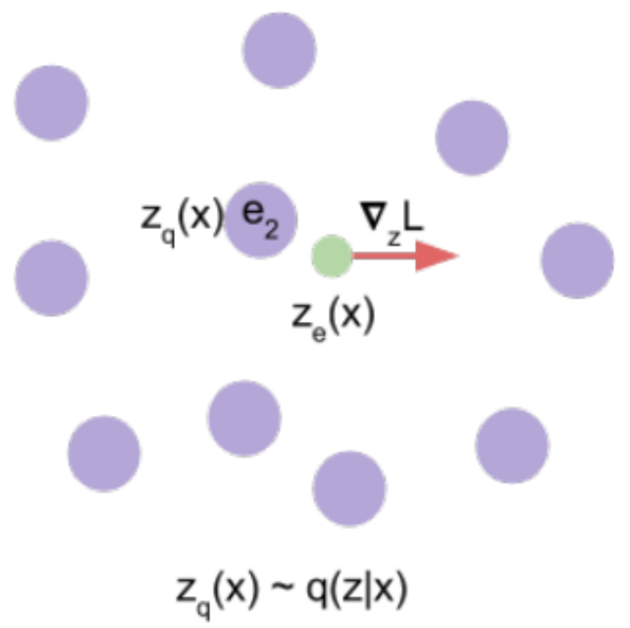
Latent space



https://arxiv.org/abs/1312.6114

VQVAE model loss

VQ-VAE loss



1. $\mathscr{L}_{\text{recon}}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{n} \left(x_i - x_i'\right)^2$

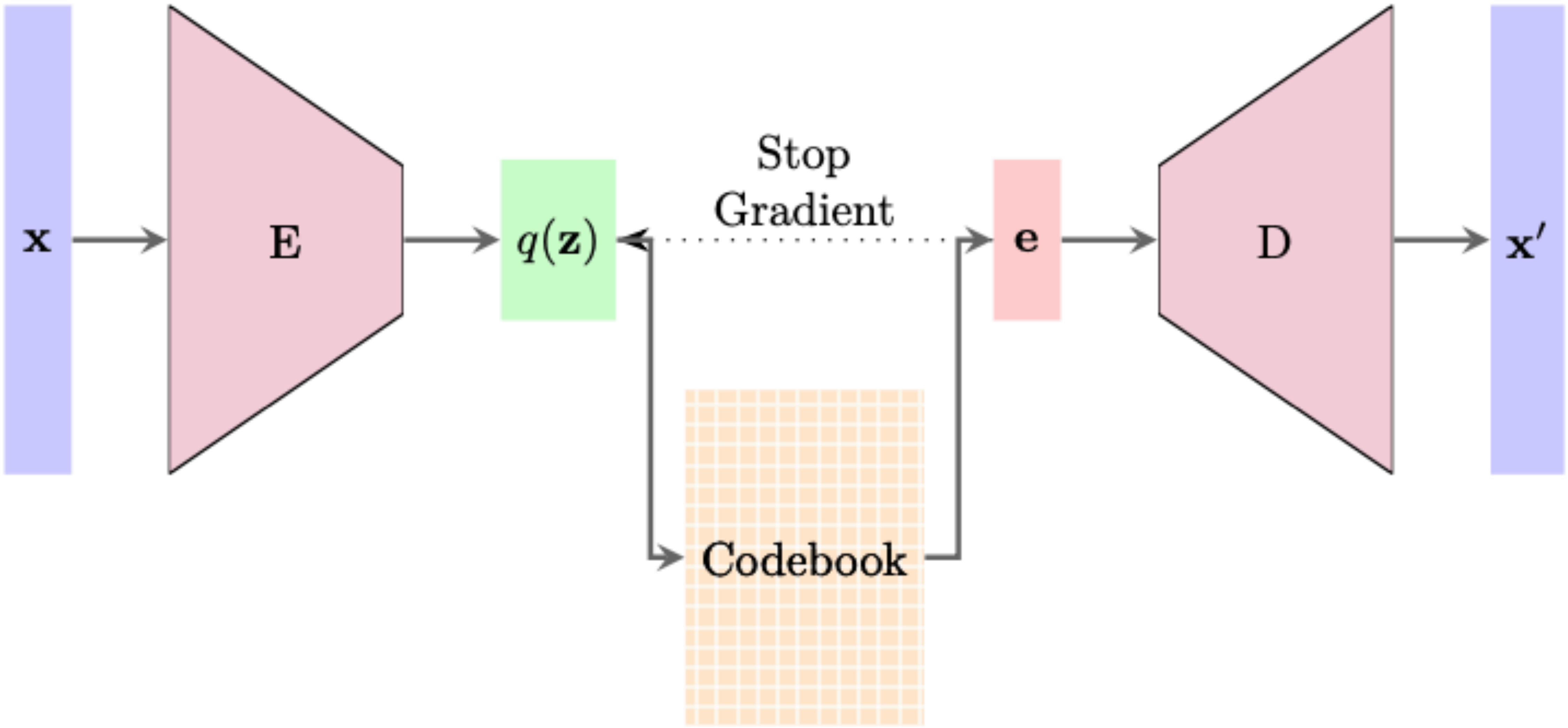
- Reconstruction loss

2. $\mathscr{L}_{\text{vq}} = ||sg[\mathbf{z}_e] - \mathbf{e}||_2^2$

- As the codebook does not have a gradient, the model cannot learn to use the codebook embeddings without a special loss term.

3. $\mathscr{L}_{\text{commit}} = ||\mathbf{z}_e - sg[\mathbf{e}]||_2^2$

- The commitment loss enforces the encoder to tightly associate its outputs with the codebook embeddings

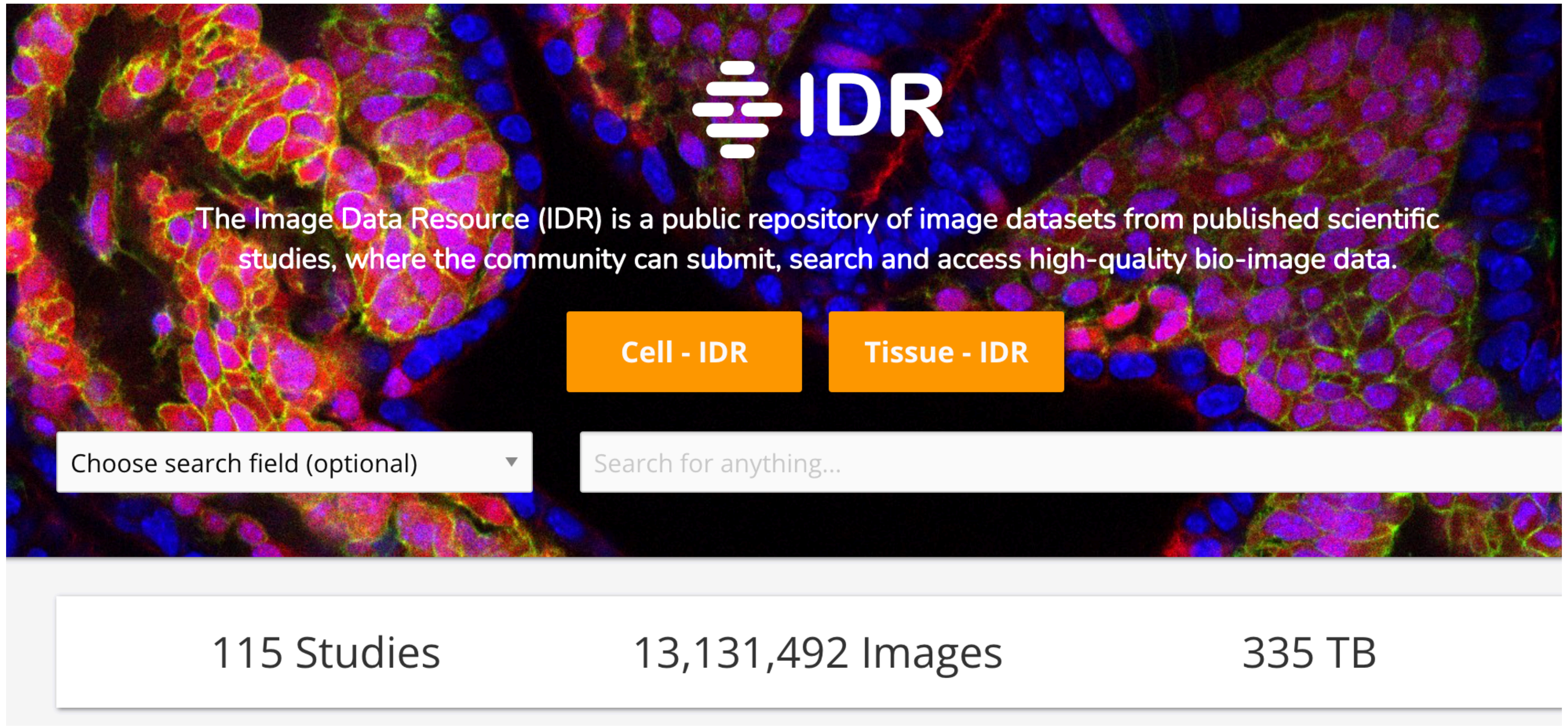


BioImage Archive

Image Data Resource (IDR)

https://idr.openmicroscopy.org/

- Reference database of published microscopy data
- Contains ~ 13 million biological images across 115 studies
- Spans multiple organisms, tissues, cell types, disease states, treatment conditions etc.

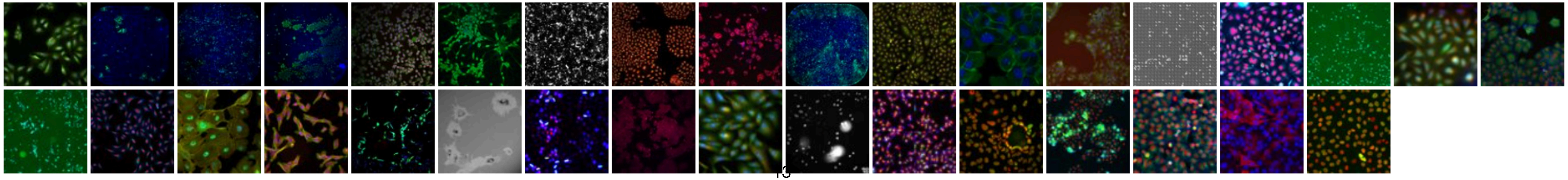


IDR

The Image Data Resource (IDR) is a public repository of image datasets from published scientific studies, where the community can submit, search and access high-quality bio-image data.

115 Studies | 13,131,492 Images | 335 TB

High-content screening (human)



Initial Pretraining scope

Selected datasets

- Subset of human cell high-content screening IDR studies with interesting and simple image classification problems

Experiments	**Image Dataset Size**	**Associated metadata**
IDR0006	[16.6 TB]	Genes (localisation)
IDR0036	[1.20 TB]	Cell states
IDR0093	[1.62 TB]	Genes (morphometric response)
IDR0094	[1.41 TB]	COVID Drug response

Technology stack

`bioimage_embed`

- **`bioimage_embed`** generates the models
- Uses `timm` for optimiser+scheduler pair
- `pydantic` for configuration and configuration validation
- `hydra` for the CLI

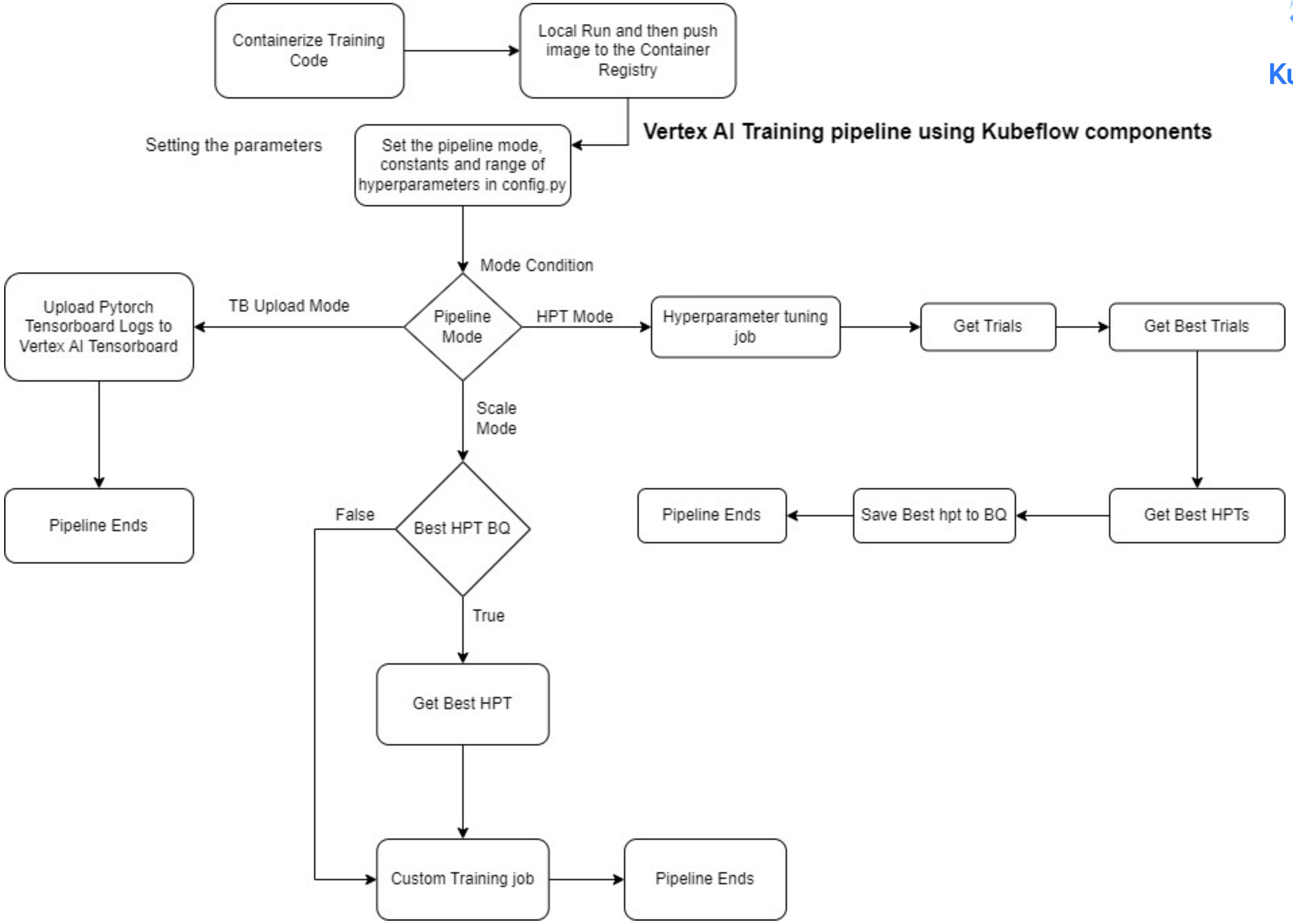
timm

- Ray for hyperparamter tuning
- Wraps this in `pytorch lightning`, handling
  - Data parallelisms, splitting
  - Checkpointing, logging
- `Albumentations` for augmentations

RAY

	Backbones	Model
prod	ResNet18, ResNet50, ResNet101, ResNet152	VAE, VQVAE, AE
beta	VIT-H, VIT-L, 🤗	MAE

Pipeline Workflow



Containerize Training Code

Local Run and then push image to the Container Registry

Vertex AI Training pipeline using Kubeflow components

Setting the parameters

Set the pipeline mode, constants and range of hyperparameters in config.py

Mode Condition

Pipeline Mode

TB Upload Mode

Upload Pytorch Tensorboard Logs to Vertex AI Tensorboard

Pipeline Ends

HPT Mode

Hyperparameter tuning job

Get Trials

Get Best Trials

Get Best HPTs

Save Best hpt to BQ

Pipeline Ends

Scale Mode

Best HPT BQ

False

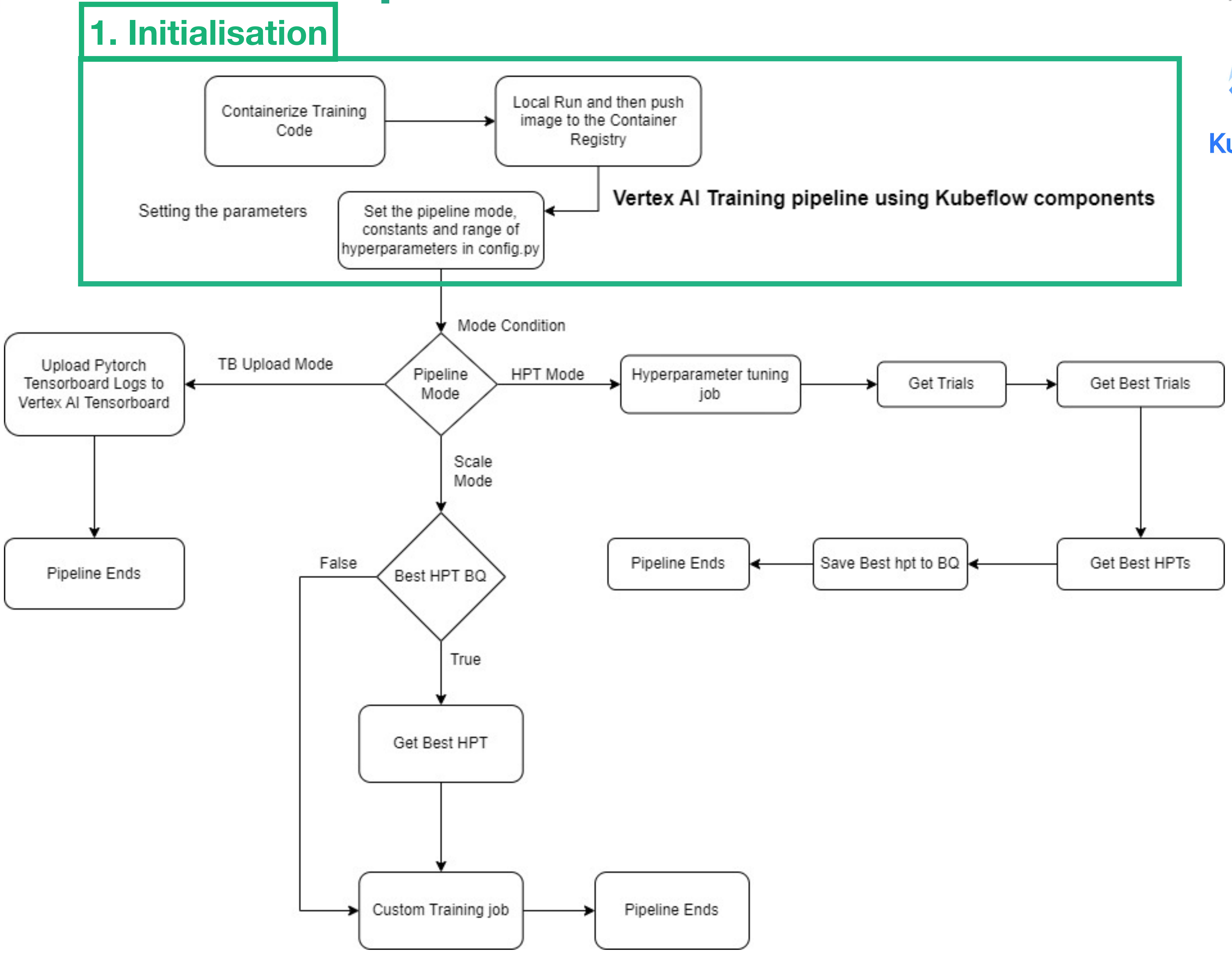
True

Get Best HPT

Custom Training job

Pipeline Ends

Pipeline Workflow



1. Initialisation
Containerize Training Code
Local Run and then push image to the Container Registry
Setting the parameters
Set the pipeline mode, constants and range of hyperparameters in config.py
Vertex AI Training pipeline using Kubeflow components
Mode Condition
Pipeline Mode
TB Upload Mode
Upload Pytorch Tensorboard Logs to Vertex AI Tensorboard
Pipeline Ends
HPT Mode
Hyperparameter tuning job
Get Trials
Get Best Trials
Get Best HPTs
Save Best hpt to BQ
Pipeline Ends
Scale Mode
Best HPT BQ
False
True
Get Best HPT
Custom Training job
Pipeline Ends

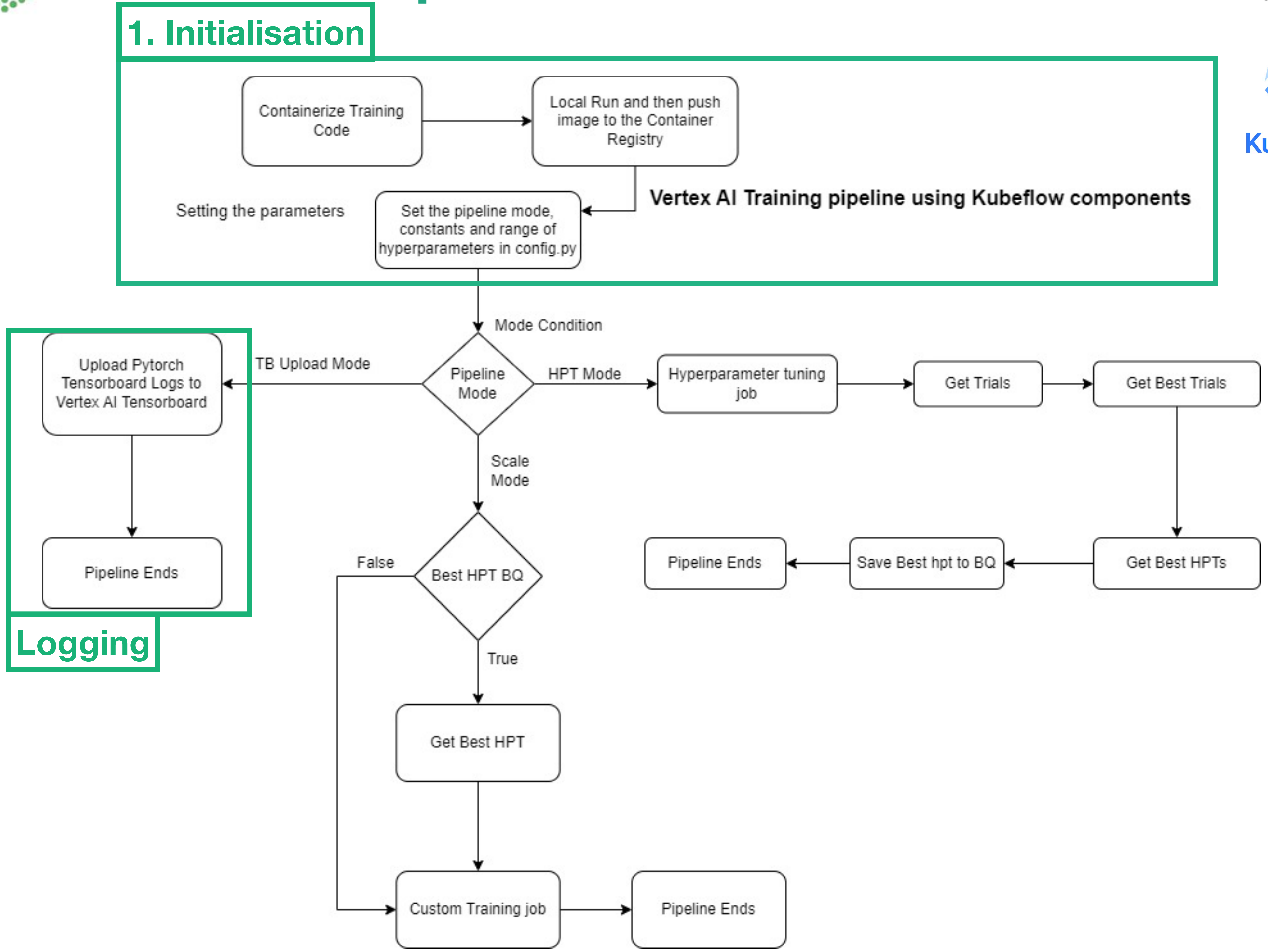
EMBL-EBI

Pipeline Workflow

quantiphi
AI - Data - Cloud

Kubeflow

vertex.ai



1. Initialisation

Containerize Training Code

Local Run and then push image to the Container Registry

Vertex AI Training pipeline using Kubeflow components

Setting the parameters

Set the pipeline mode, constants and range of hyperparameters in config.py

Mode Condition

Pipeline Mode

TB Upload Mode

Upload Pytorch Tensorboard Logs to Vertex AI Tensorboard

Pipeline Ends

Logging

HPT Mode

Hyperparameter tuning job

Get Trials

Get Best Trials

Get Best HPTs

Save Best hpt to BQ

Pipeline Ends

Scale Mode

Best HPT BQ

False

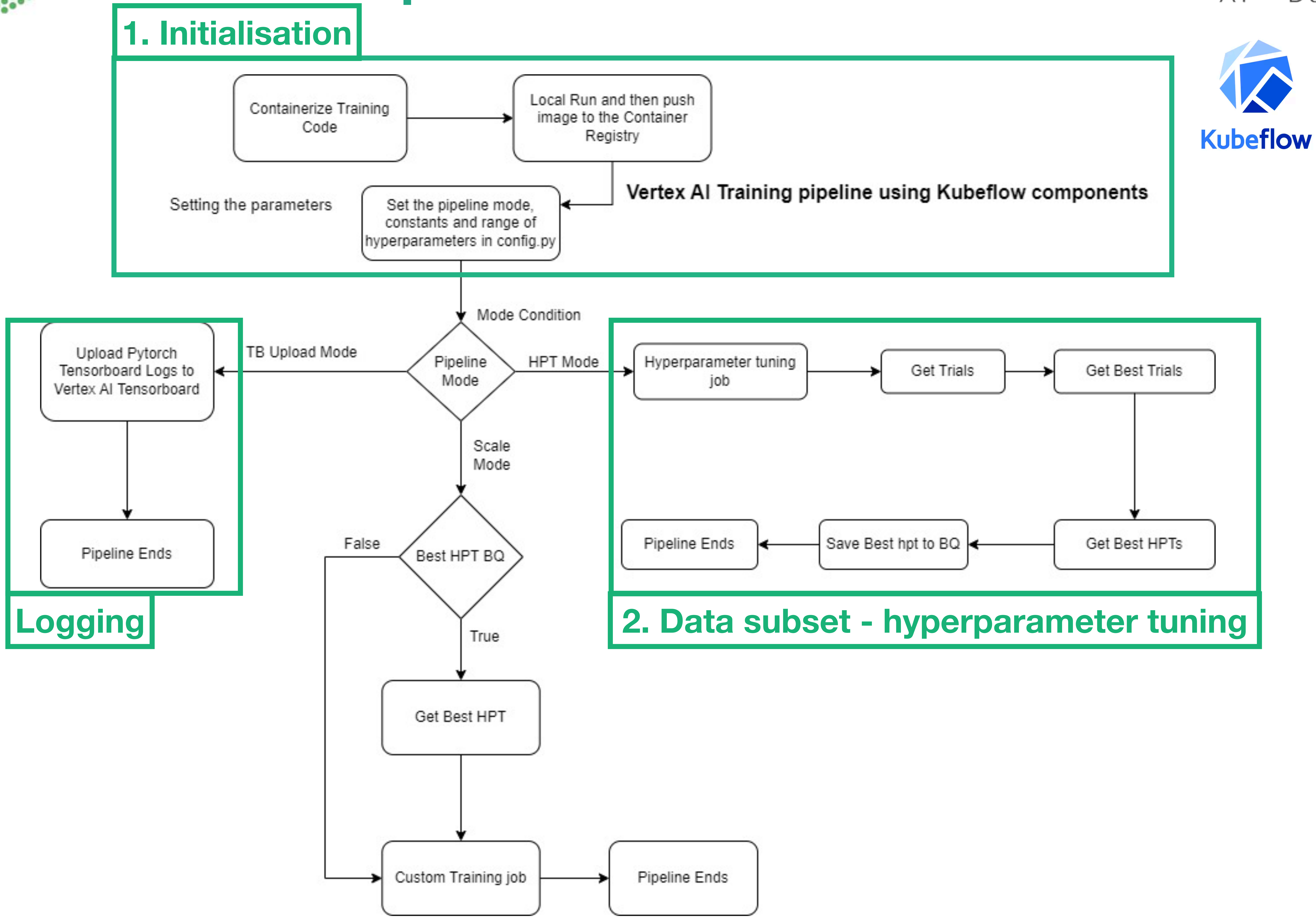
True

Get Best HPT

Custom Training job

Pipeline Ends

Pipeline Workflow



1. Initialisation

Containerize Training Code

Local Run and then push image to the Container Registry

Vertex AI Training pipeline using Kubeflow components

Setting the parameters

Set the pipeline mode, constants and range of hyperparameters in config.py

Mode Condition

Pipeline Mode

TB Upload Mode

Upload Pytorch Tensorboard Logs to Vertex AI Tensorboard

Pipeline Ends

Logging

HPT Mode

Hyperparameter tuning job

Get Trials

Get Best Trials

Get Best HPTs

Save Best hpt to BQ

Pipeline Ends

2. Data subset - hyperparameter tuning

Scale Mode

Best HPT BQ

False

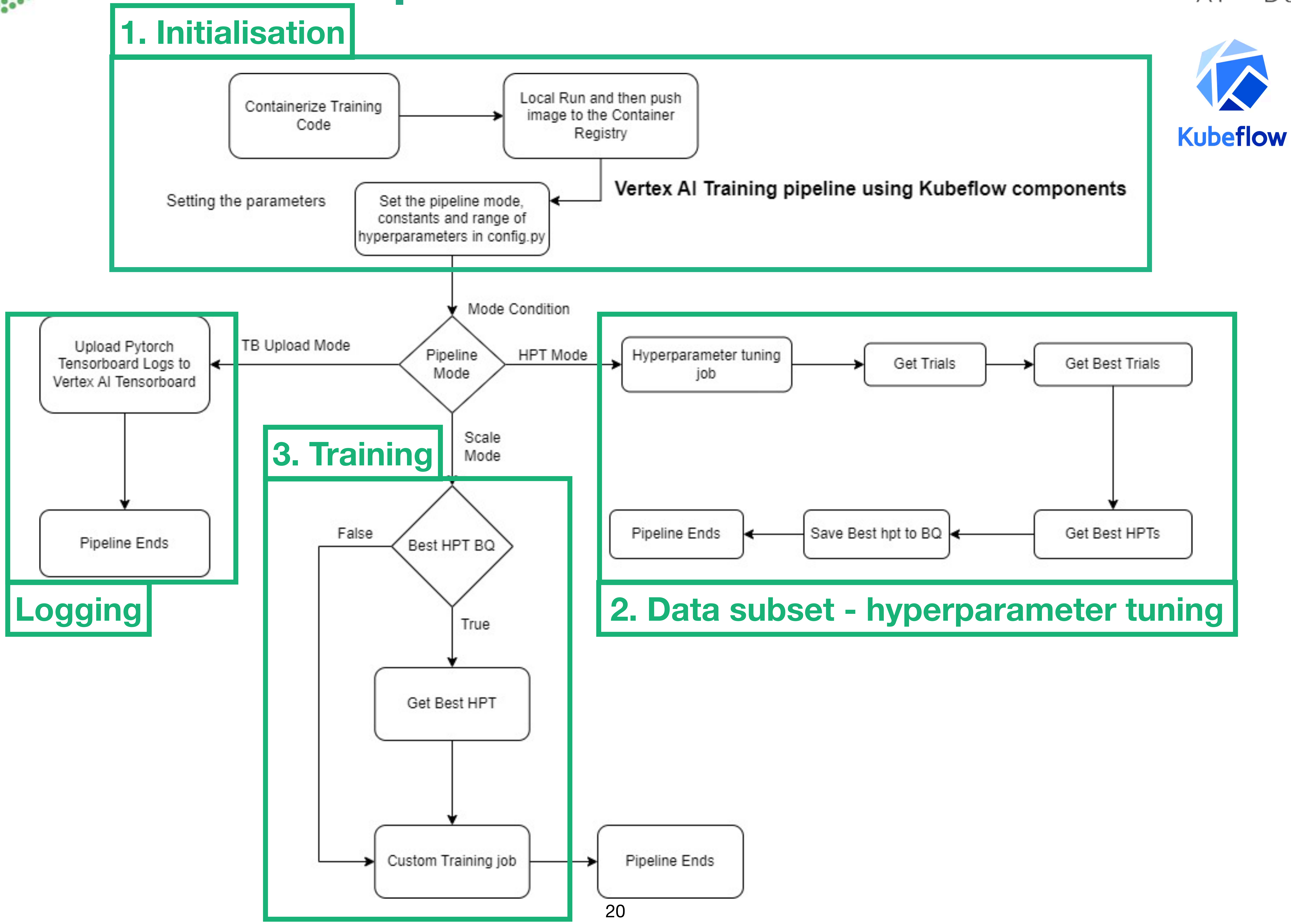
True

Get Best HPT

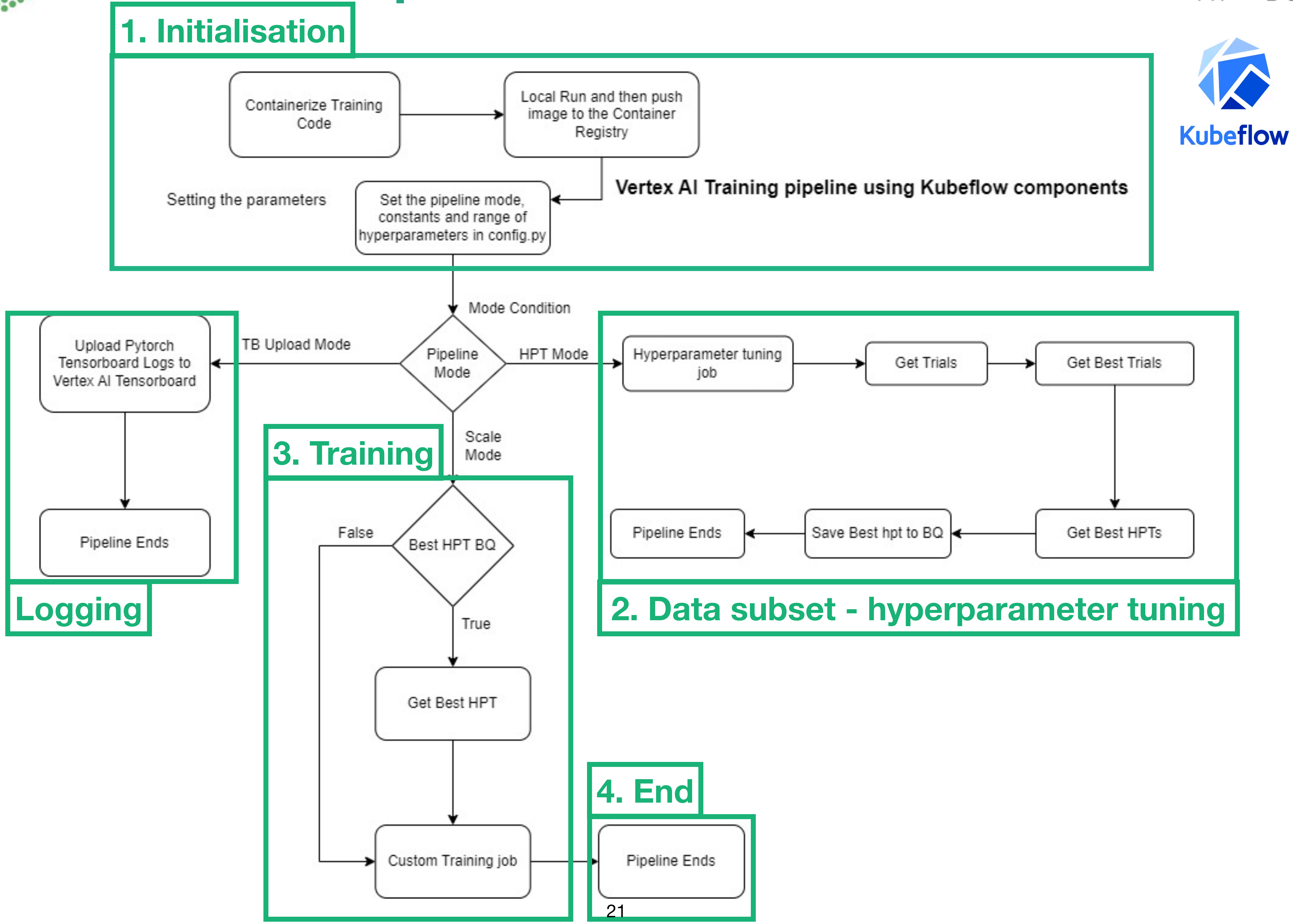
Custom Training job

Pipeline Ends

Pipeline Workflow


1. Initialisation
Containerize Training Code
Local Run and then push image to the Container Registry
Setting the parameters
Set the pipeline mode, constants and range of hyperparameters in config.py
Vertex AI Training pipeline using Kubeflow components
Mode Condition
Pipeline Mode
TB Upload Mode
HPT Mode
Scale Mode
Upload Pytorch Tensorboard Logs to Vertex AI Tensorboard
Pipeline Ends
Logging
3. Training
False
Best HPT BQ
True
Get Best HPT
Custom Training job
Pipeline Ends
Hyperparameter tuning job
Get Trials
Get Best Trials
Get Best HPTs
Save Best hpt to BQ
Pipeline Ends
2. Data subset - hyperparameter tuning

Pipeline Workflow



1. Initialisation

Containerize Training Code → Local Run and then push image to the Container Registry → Set the pipeline mode, constants and range of hyperparameters in config.py

Setting the parameters

Vertex AI Training pipeline using Kubeflow components

Mode Condition

Pipeline Mode

TB Upload Mode

HPT Mode

Scale Mode

Logging

Upload Pytorch Tensorboard Logs to Vertex AI Tensorboard → Pipeline Ends

2. Data subset - hyperparameter tuning

Hyperparameter tuning job → Get Trials → Get Best Trials → Get Best HPTs → Save Best hpt to BQ → Pipeline Ends

3. Training

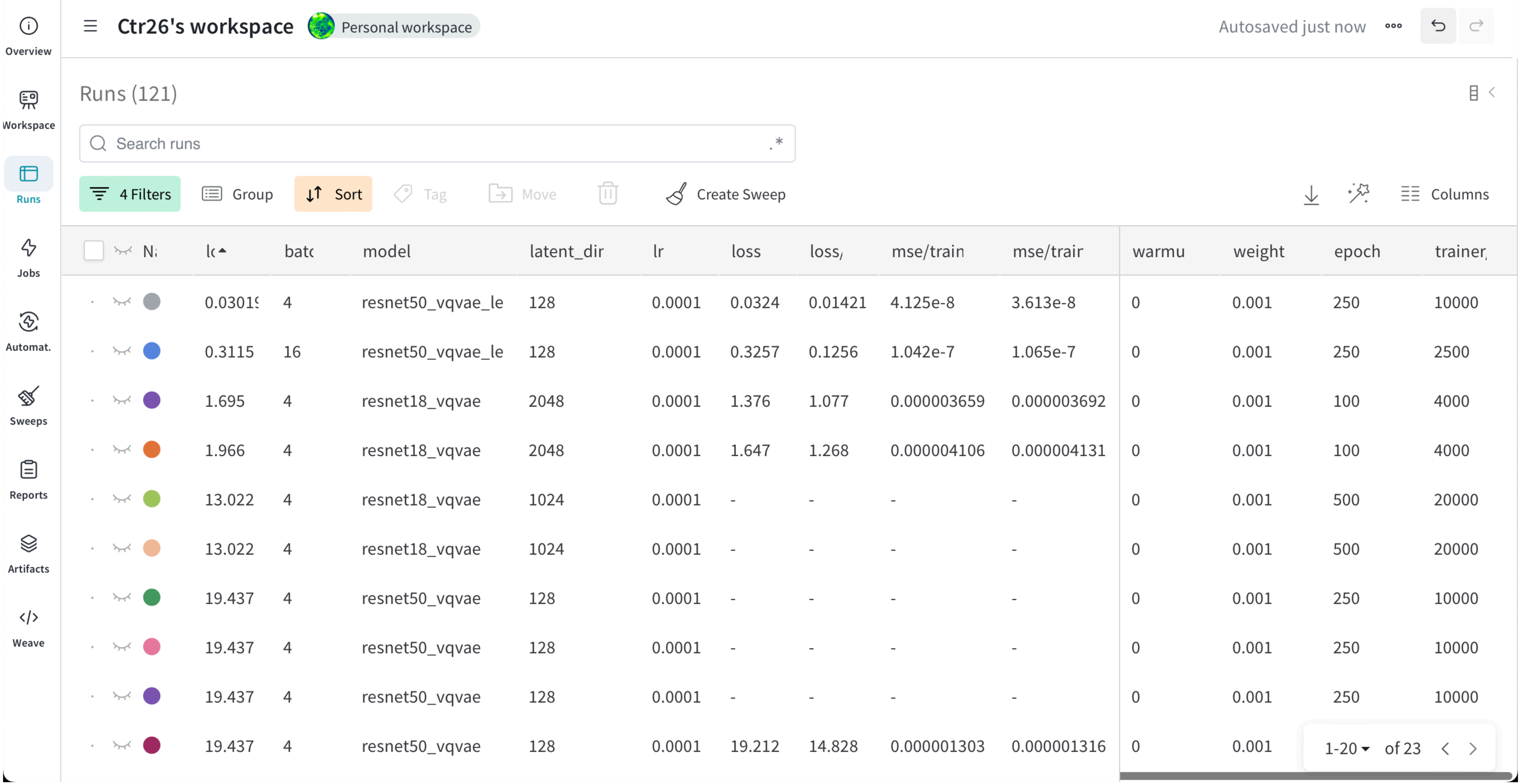
Best HPT BQ

True → Get Best HPT → Custom Training job

False → Custom Training job

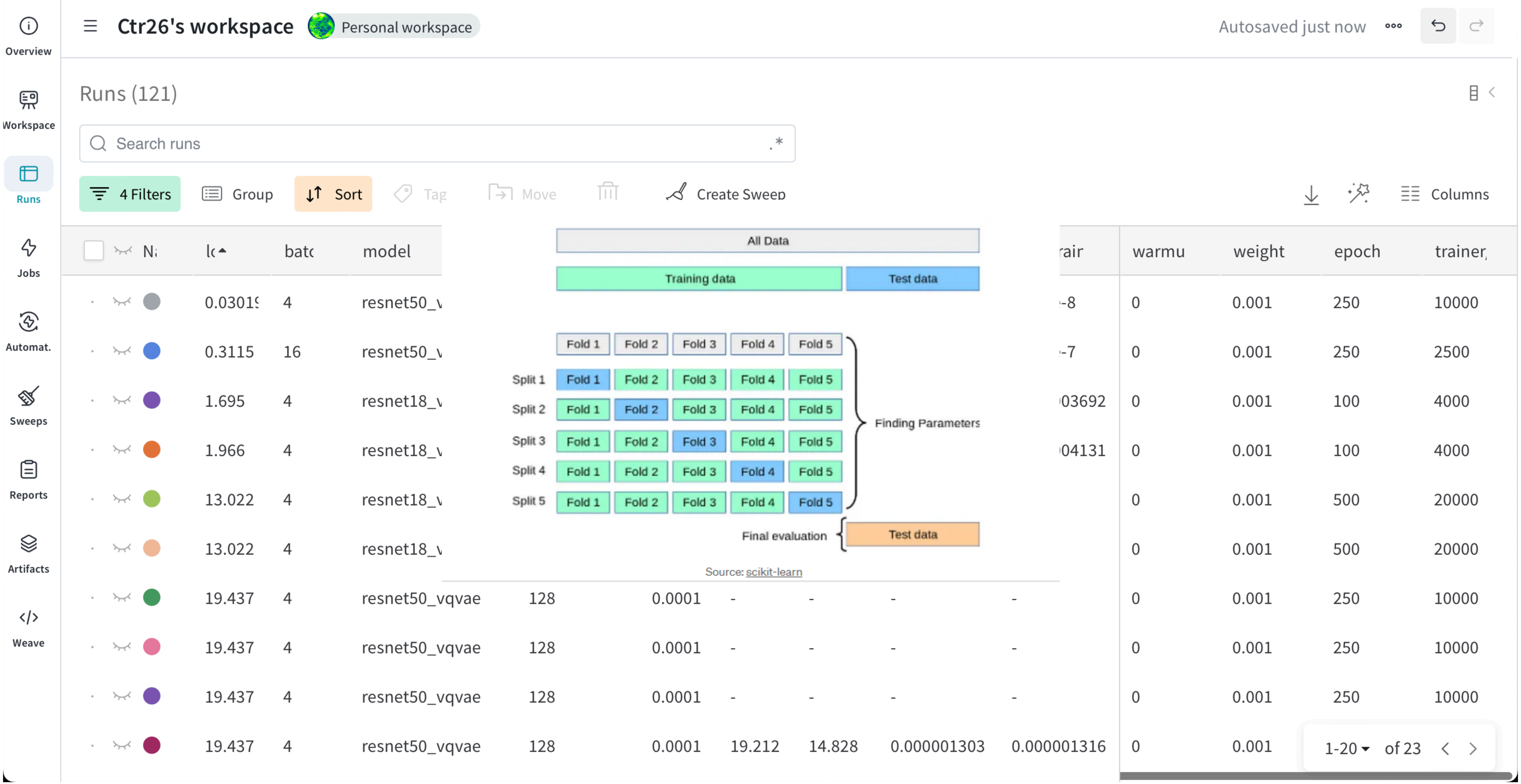
4. End

Pipeline Ends



Runs (121)

	Na	lo▲	batc	model	latent_dir	lr	loss	loss/	mse/train	mse/trair	warmu	weight	epoch	trainer
□	●	0.0301	4	resnet50_vqvae_le	128	0.0001	0.0324	0.01421	4.125e-8	3.613e-8	0	0.001	250	10000
	●	0.3115	16	resnet50_vqvae_le	128	0.0001	0.3257	0.1256	1.042e-7	1.065e-7	0	0.001	250	2500
	●	1.695	4	resnet18_vqvae	2048	0.0001	1.376	1.077	0.000003659	0.000003692	0	0.001	100	4000
	●	1.966	4	resnet18_vqvae	2048	0.0001	1.647	1.268	0.000004106	0.000004131	0	0.001	100	4000
	●	13.022	4	resnet18_vqvae	1024	0.0001	-	-	-	-	0	0.001	500	20000
	●	13.022	4	resnet18_vqvae	1024	0.0001	-	-	-	-	0	0.001	500	20000
	●	19.437	4	resnet50_vqvae	128	0.0001	-	-	-	-	0	0.001	250	10000
	●	19.437	4	resnet50_vqvae	128	0.0001	-	-	-	-	0	0.001	250	10000
	●	19.437	4	resnet50_vqvae	128	0.0001	-	-	-	-	0	0.001	250	10000
	●	19.437	4	resnet50_vqvae	128	0.0001	19.212	14.828	0.000001303	0.000001316	0	0.001		



Source: scikit-learn

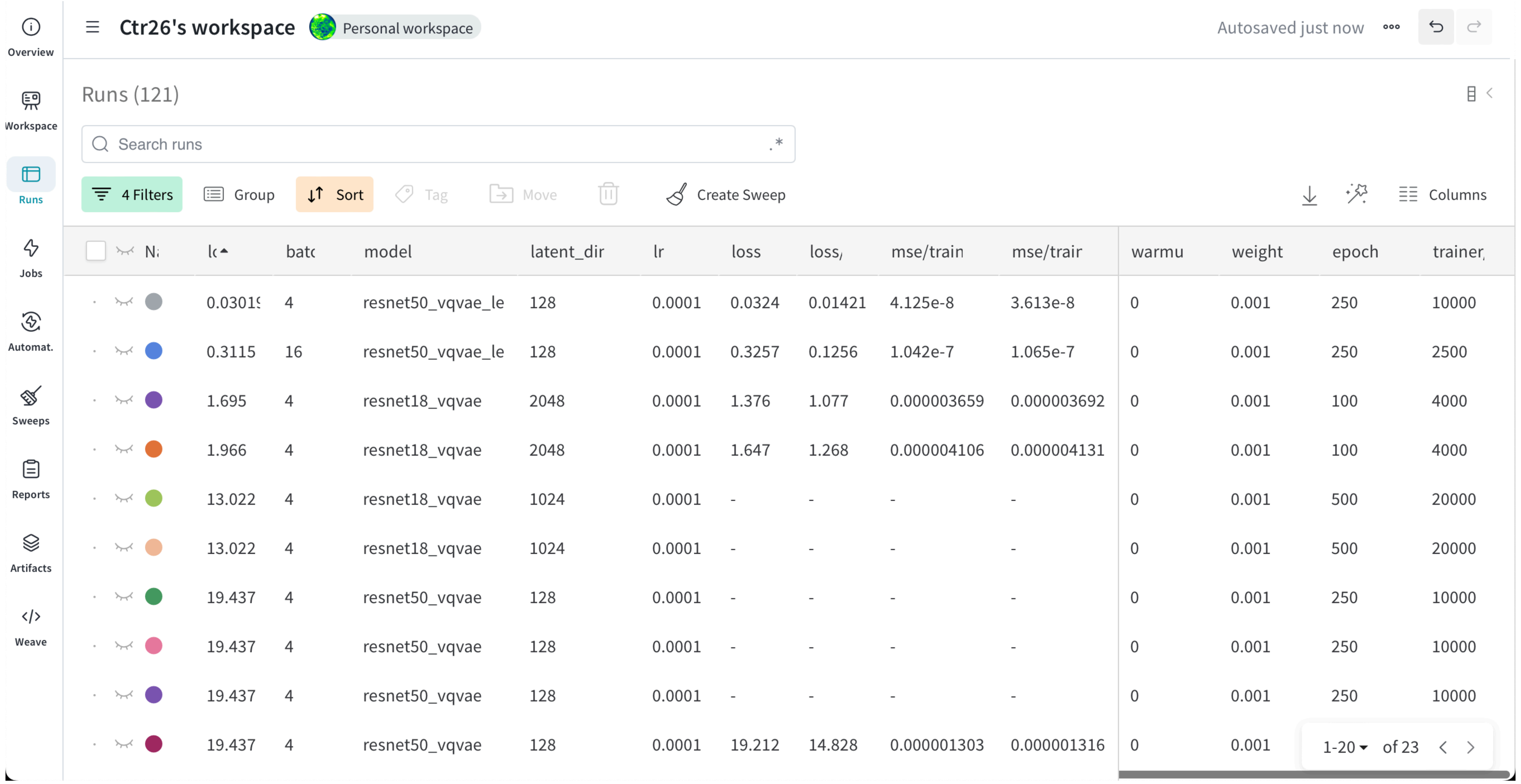
	Na	lc▲	batc	model						air	warmu	weight	epoch	trainer
●		0.0301	4	resnet50_v						-8	0	0.001	250	10000
●		0.3115	16	resnet50_v						-7	0	0.001	250	2500
●		1.695	4	resnet18_v						03692	0	0.001	100	4000
●		1.966	4	resnet18_v						04131	0	0.001	100	4000
●		13.022	4	resnet18_v							0	0.001	500	20000
●		13.022	4	resnet18_v							0	0.001	500	20000
●		19.437	4	resnet50_vqvae	128	0.0001	-	-	-	-	0	0.001	250	10000
●		19.437	4	resnet50_vqvae	128	0.0001	-	-	-	-	0	0.001	250	10000
●		19.437	4	resnet50_vqvae	128	0.0001	-	-	-	-	0	0.001	250	10000
●		19.437	4	resnet50_vqvae	128	0.0001	19.212	14.828	0.000001303	0.000001316	0	0.001		

1-20 ▾ of 23

Model training

`bioimage_embed`

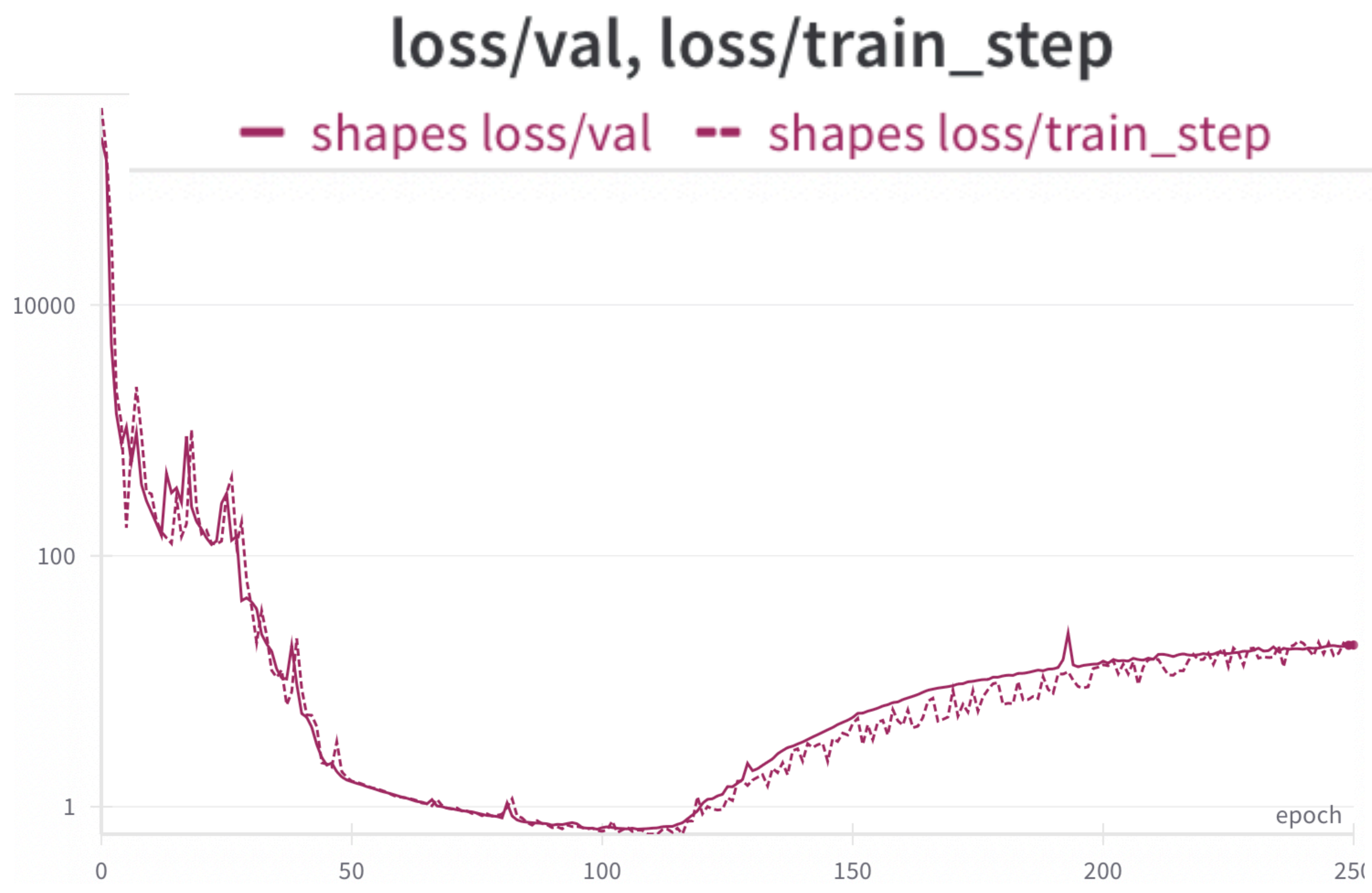
	Parameter	Value
Model	epochs	100
	batch_size	64
	latent_dim	16
	num_embeddings	16
	num_hiddens	16
	num_residual_hiddens	32
	num_residual_layers	150
	commitment_cost	0.25
	decay	0.99
Optimizer	opt	LAMB
	lr	0.001
	weight_decay	0.0001
	momentum	0.9
LR scheduler	sched	cosine
	min_lr	1e-4
	warmup_epochs	5
	warmup_lr	1e-6
	cooldown_epochs	10
	t_max	50
	cycle_momentum	False



Runs (121)

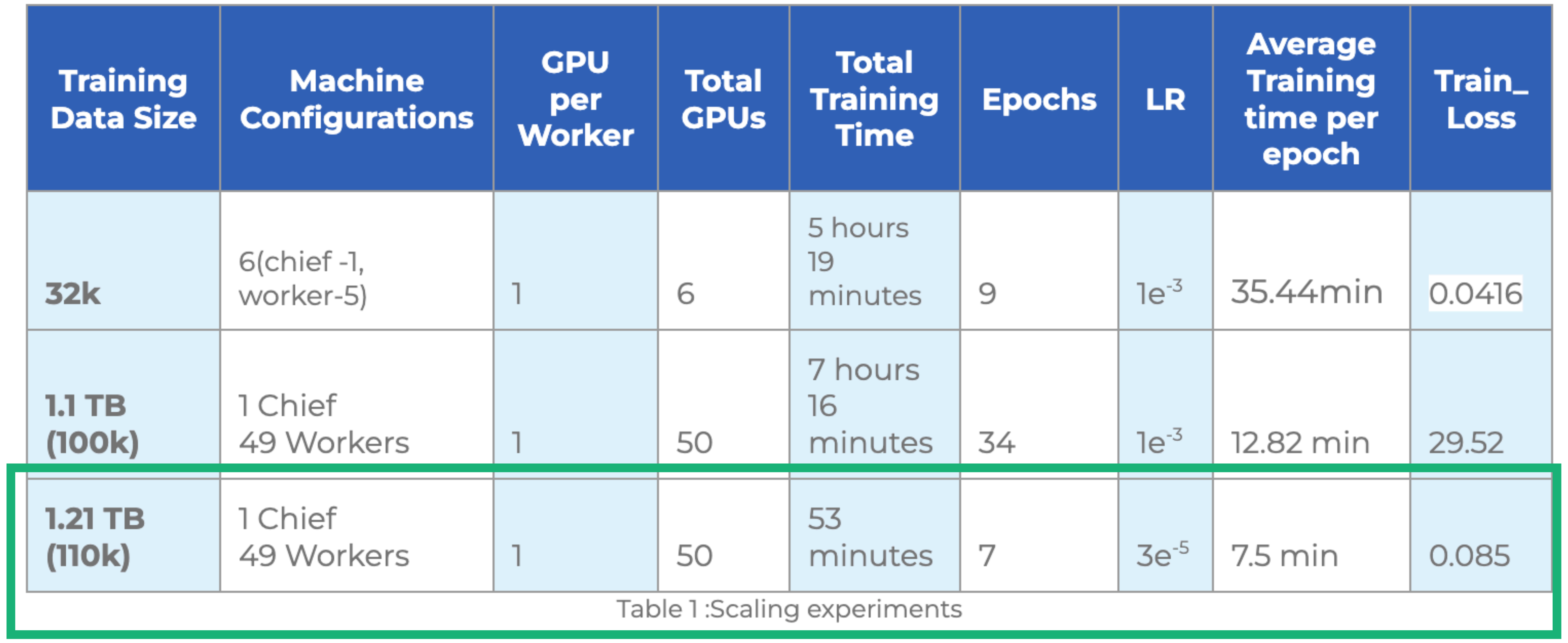
N:	lc▲	batc	model	latent_dir	lr	loss	loss/	mse/train	mse/trair	warmu	weight	epoch	trainer,
●	0.0301ξ	4	resnet50_vqvae_le	128	0.0001	0.0324	0.01421	4.125e-8	3.613e-8	0	0.001	250	10000
●	0.3115	16	resnet50_vqvae_le	128	0.0001	0.3257	0.1256	1.042e-7	1.065e-7	0	0.001	250	2500
●	1.695	4	resnet18_vqvae	2048	0.0001	1.376	1.077	0.000003659	0.000003692	0	0.001	100	4000
●	1.966	4	resnet18_vqvae	2048	0.0001	1.647	1.268	0.000004106	0.000004131	0	0.001	100	4000
●	13.022	4	resnet18_vqvae	1024	0.0001	-	-	-	-	0	0.001	500	20000
●	13.022	4	resnet18_vqvae	1024	0.0001	-	-	-	-	0	0.001	500	20000
●	19.437	4	resnet50_vqvae	128	0.0001	-	-	-	-	0	0.001	250	10000
●	19.437	4	resnet50_vqvae	128	0.0001	-	-	-	-	0	0.001	250	10000
●	19.437	4	resnet50_vqvae	128	0.0001	-	-	-	-	0	0.001	250	10000
●	19.437	4	resnet50_vqvae	128	0.0001	19.212	14.828	0.000001303	0.000001316	0	0.001		

1-20 ▾ of 23 ‹ ›



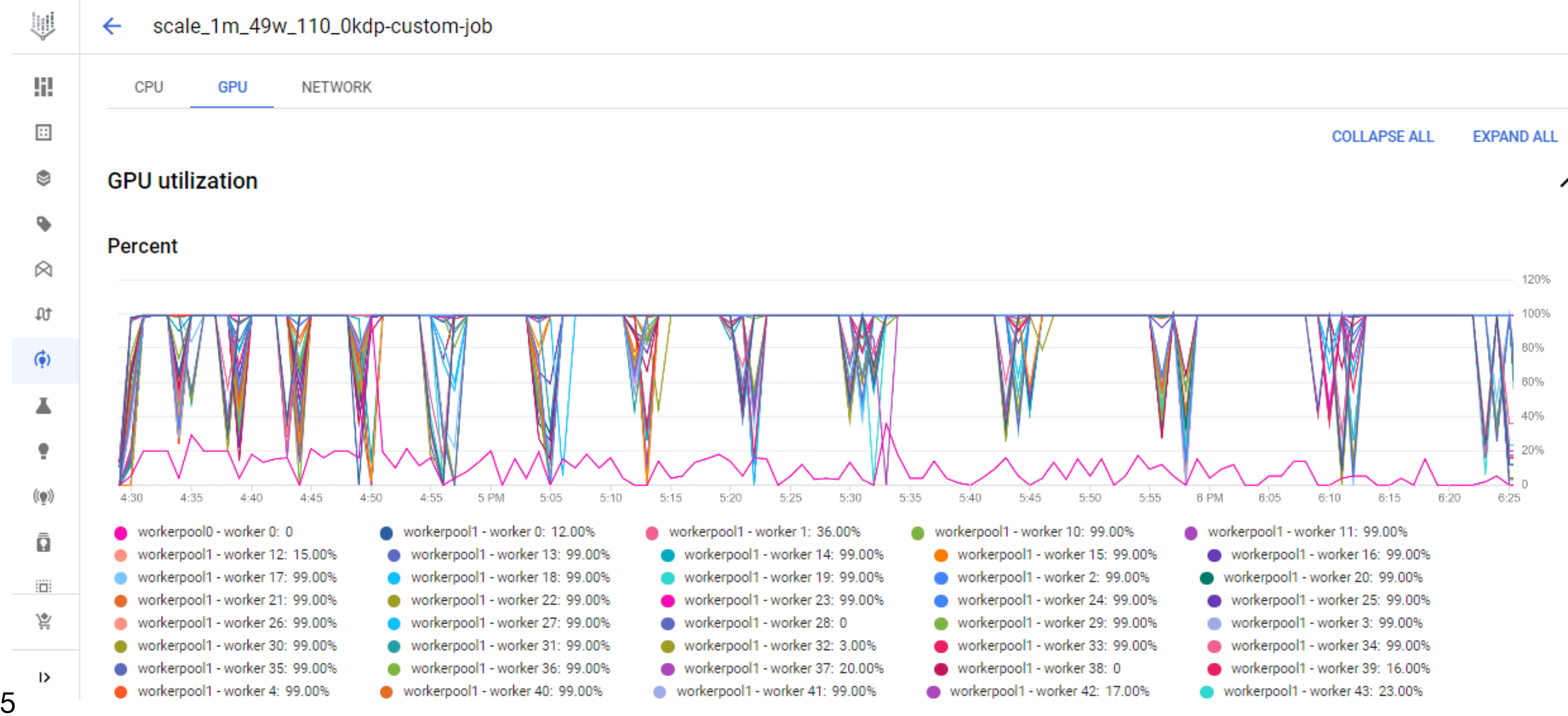
Experiments for scaling

- CPU machine type used :n1-standard-16
- The accelerator used is **GPU**: NVIDIA_TESLA_T4
    - **50 concurrent GPUs**
    - Training time reduced due to **early stopping**
        - Training time is heavily reliant on hyperparameters
        - Decreasing learning rate yielded 7-fold improvement in training time
- Scale training of the model up to **110 000 images (1.2 TB)**, representing a large subset of **study IDR00093**.



Training Data Size	Machine Configurations	GPU per Worker	Total GPUs	Total Training Time	Epochs	LR	Average Training time per epoch	Train_ Loss
32k	6(chief -1, worker-5)	1	6	5 hours 19 minutes	9	$1e^{-3}$	35.44min	0.0416
1.1 TB (100k)	1 Chief 49 Workers	1	50	7 hours 16 minutes	34	$1e^{-3}$	12.82 min	29.52
1.21 TB (110k)	1 Chief 49 Workers	1	50	53 minutes	7	$3e^{-5}$	7.5 min	0.085

Table 1 :Scaling experiments



scale_1m_49w_110_0kdp-custom-job

GPU utilization

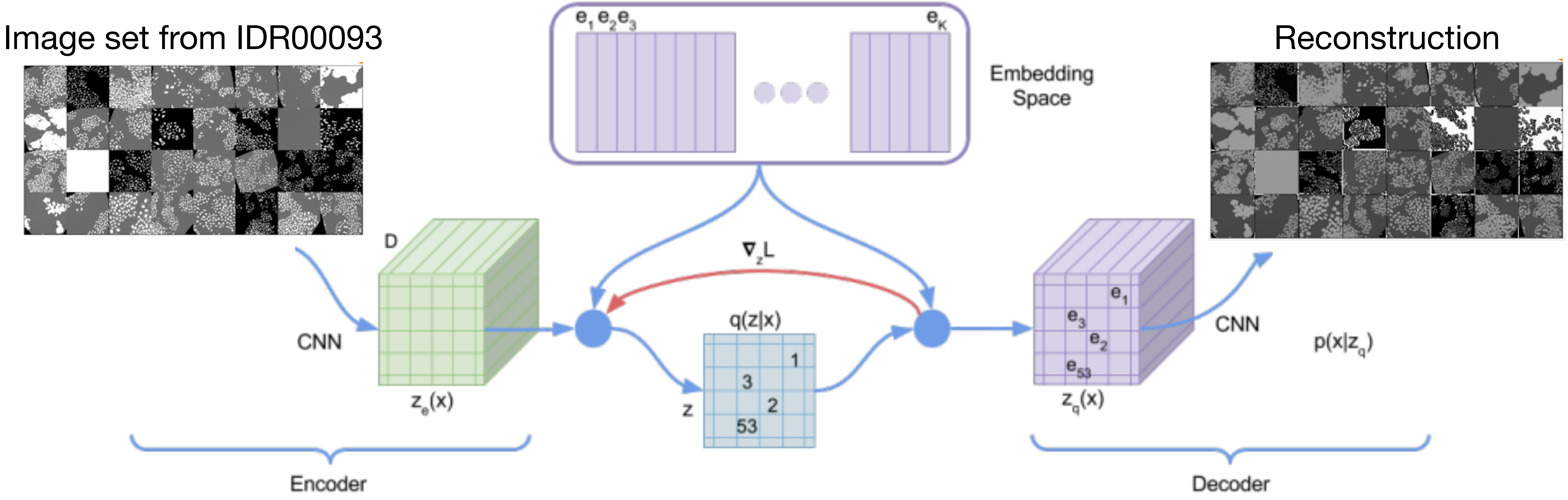
● workerpool0 - worker 0: 0
● workerpool1 - worker 0: 12.00%
● workerpool1 - worker 1: 36.00%
● workerpool1 - worker 10: 99.00%
● workerpool1 - worker 11: 99.00%
● workerpool1 - worker 12: 15.00%
● workerpool1 - worker 13: 99.00%
● workerpool1 - worker 14: 99.00%
● workerpool1 - worker 15: 99.00%
● workerpool1 - worker 16: 99.00%
● workerpool1 - worker 17: 99.00%
● workerpool1 - worker 18: 99.00%
● workerpool1 - worker 19: 99.00%
● workerpool1 - worker 2: 99.00%
● workerpool1 - worker 20: 99.00%
● workerpool1 - worker 21: 99.00%
● workerpool1 - worker 22: 99.00%
● workerpool1 - worker 23: 99.00%
● workerpool1 - worker 24: 99.00%
● workerpool1 - worker 25: 99.00%
● workerpool1 - worker 26: 99.00%
● workerpool1 - worker 27: 99.00%
● workerpool1 - worker 28: 0
● workerpool1 - worker 29: 99.00%
● workerpool1 - worker 3: 99.00%
● workerpool1 - worker 30: 99.00%
● workerpool1 - worker 31: 99.00%
● workerpool1 - worker 32: 3.00%
● workerpool1 - worker 33: 99.00%
● workerpool1 - worker 34: 99.00%
● workerpool1 - worker 35: 99.00%
● workerpool1 - worker 36: 99.00%
● workerpool1 - worker 37: 20.00%
● workerpool1 - worker 38: 0
● workerpool1 - worker 39: 16.00%
● workerpool1 - worker 4: 99.00%
● workerpool1 - worker 40: 99.00%
● workerpool1 - worker 41: 99.00%
● workerpool1 - worker 42: 17.00%
● workerpool1 - worker 43: 23.00%

Visual assessment of model image reconstruction

Input and output of trained model

Image set from IDR00093

Reconstruction



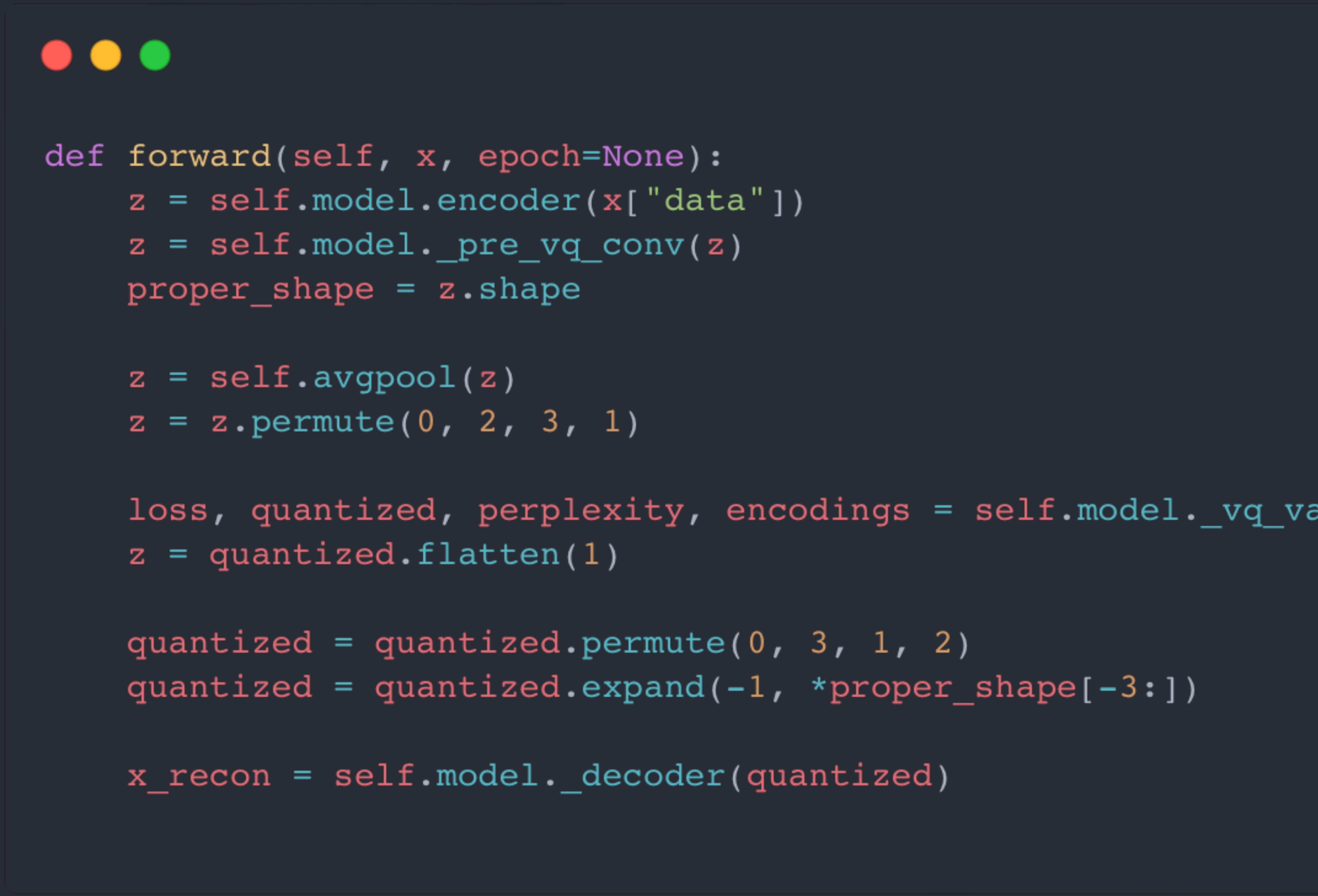
Bioimage specific considerations and tricks

1. Tunable latent space size
    - Users want to choose how many features the model generates
2. ND Colour Channels
    - Most models by default are 3 colour channels
    - BioImages do not abide by this
3. Batch effects
    - Images from repeats should produce similar features
4. Mixed datasets
    - Variable size length of channels and dyes, time, depth

1. Tunable latent space size

VQVAE as an example

- ***Issue:***
  - Encoders vary in their feature size
  - Decoders expect a specific tensor size in
  - ResNetXX sizes:
    - 18,34,50: 512
    - 101,152: 2048
- **Trick:**
  - Adaptive averaging from *encoder* output
  - Tensor repeat (`torch.expand`) back to what the *decoder* expects

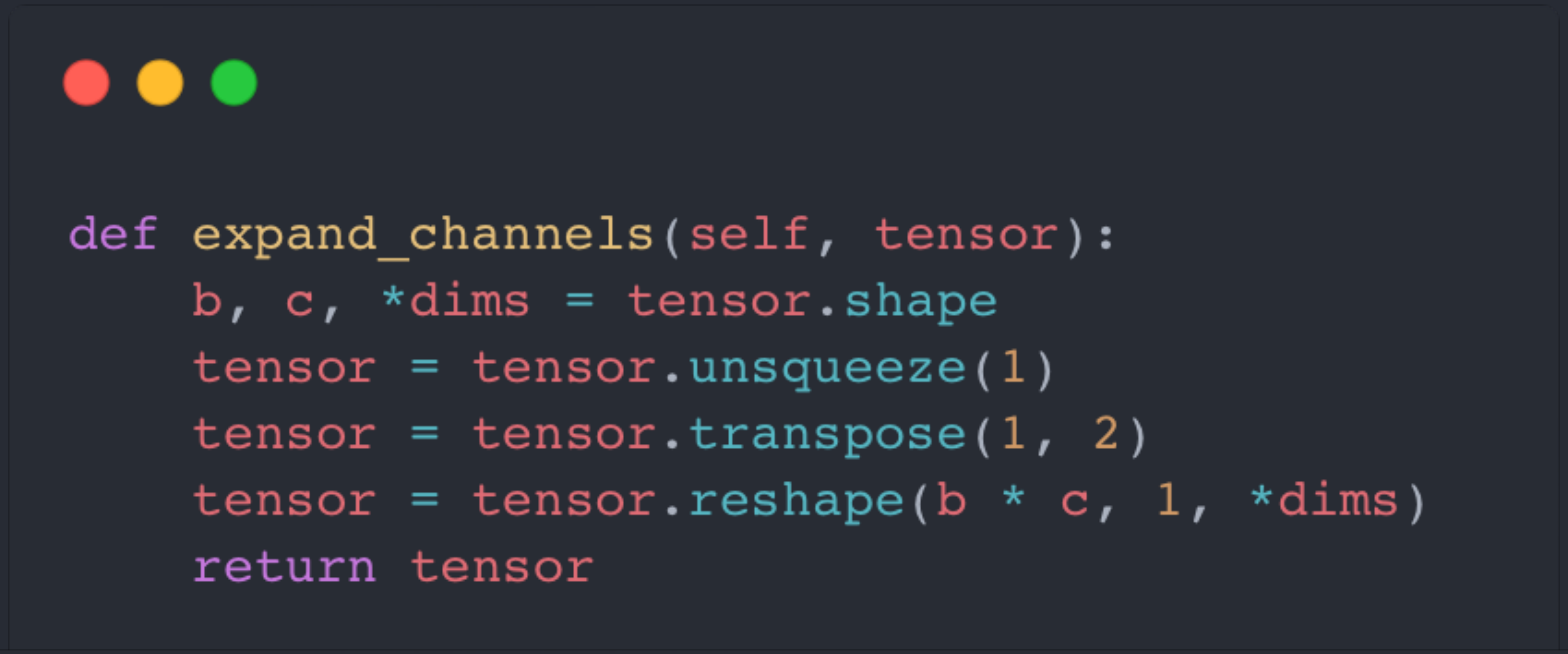


```
def forward(self, x, epoch=None):
    z = self.model.encoder(x["data"])
    z = self.model._pre_vq_conv(z)
    proper_shape = z.shape

    z = self.avgpool(z)
    z = z.permute(0, 2, 3, 1)

    loss, quantized, perplexity, encodings = self.model._vq_va
    z = quantized.flatten(1)

    quantized = quantized.permute(0, 3, 1, 2)
    quantized = quantized.expand(-1, *proper_shape[-3:])

    x_recon = self.model._decoder(quantized)
```

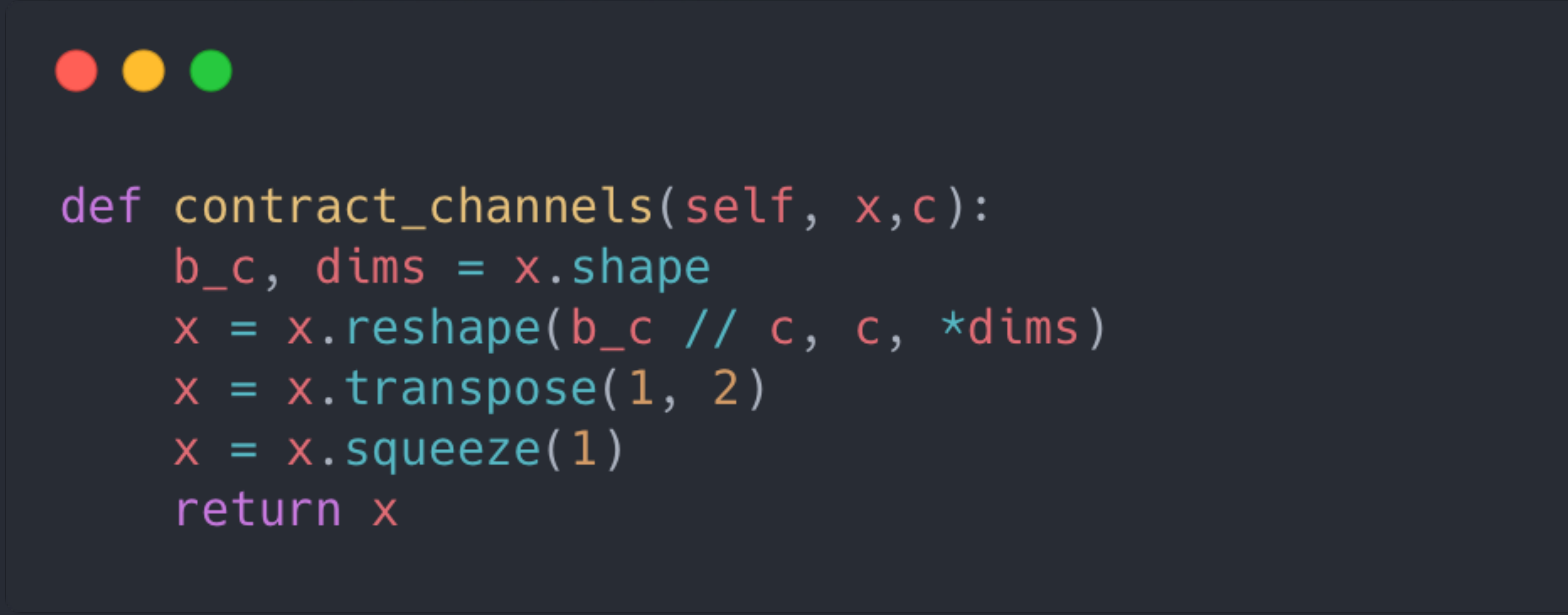
2. ND Colour channels

Expand channels into batch

- ***Issue:*** Most image models take inputs of x.shape
  - Biomage datasets can be
    - [batch,c,y,x] where $c \in \{n \in \mathbb{Z} \mid n \geq 1\}$
  - Most image models take inputs of x.shape
    - [batch,3,y,x]
- ***Trick***: Put colour channels in batch_dim, then expand c_dim to 3
  - Possible because batch is allowed to vary during training



```
def expand_channels(self, tensor):
    b, c, *dims = tensor.shape
    tensor = tensor.unsqueeze(1)
    tensor = tensor.transpose(1, 2)
    tensor = tensor.reshape(b * c, 1, *dims)
    return tensor
```



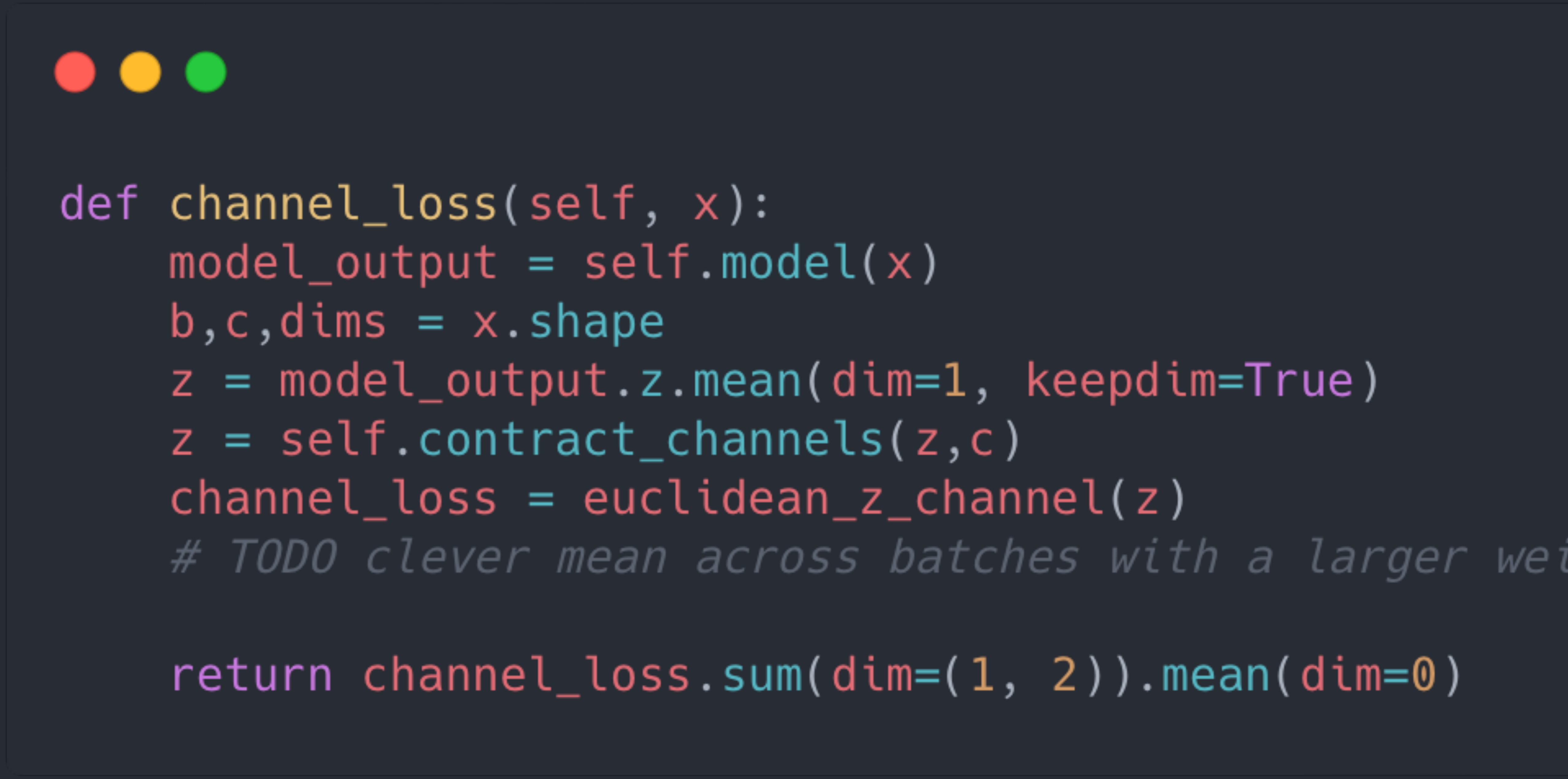
```
def contract_channels(self, x,c):
    b_c, dims = x.shape
    x = x.reshape(b_c // c, c, *dims)
    x = x.transpose(1, 2)
    x = x.squeeze(1)
    return x
```

3. + 4. Mixed data

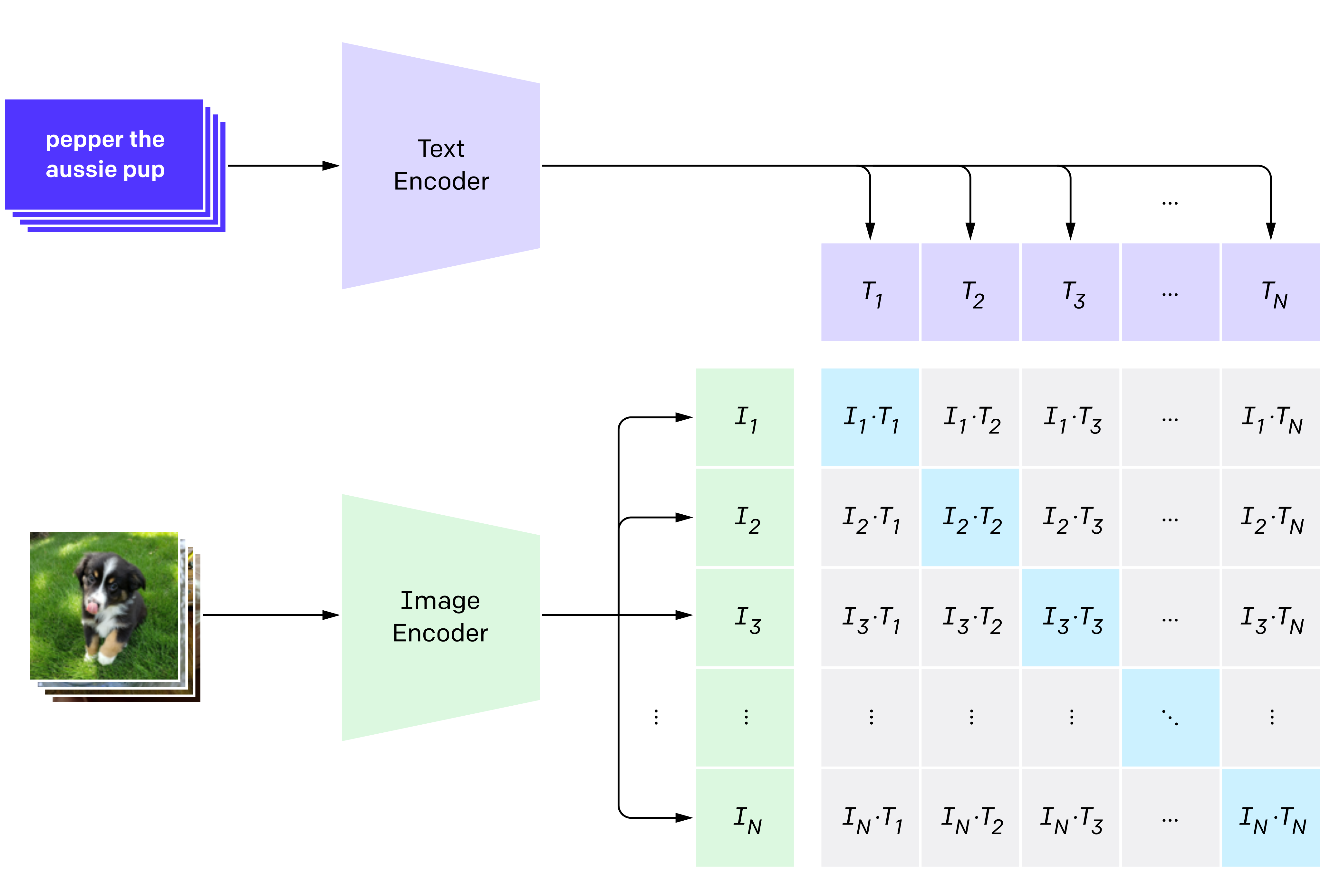
Full process

- ***Issue:*** Due to previous tricks the model doesn't know that additional dimensions are related, (`z,c,t,plate,well,batch` etc)

- ***Trick:*** Use contrastive learning to make sure their latent representations are similar



```
def channel_loss(self, x):
    model_output = self.model(x)
    b,c,dims = x.shape
    z = model_output.z.mean(dim=1, keepdim=True)
    z = self.contract_channels(z,c)
    channel_loss = euclidean_z_channel(z)
    # TODO clever mean across batches with a larger wei

    return channel_loss.sum(dim=(1, 2)).mean(dim=0)
```

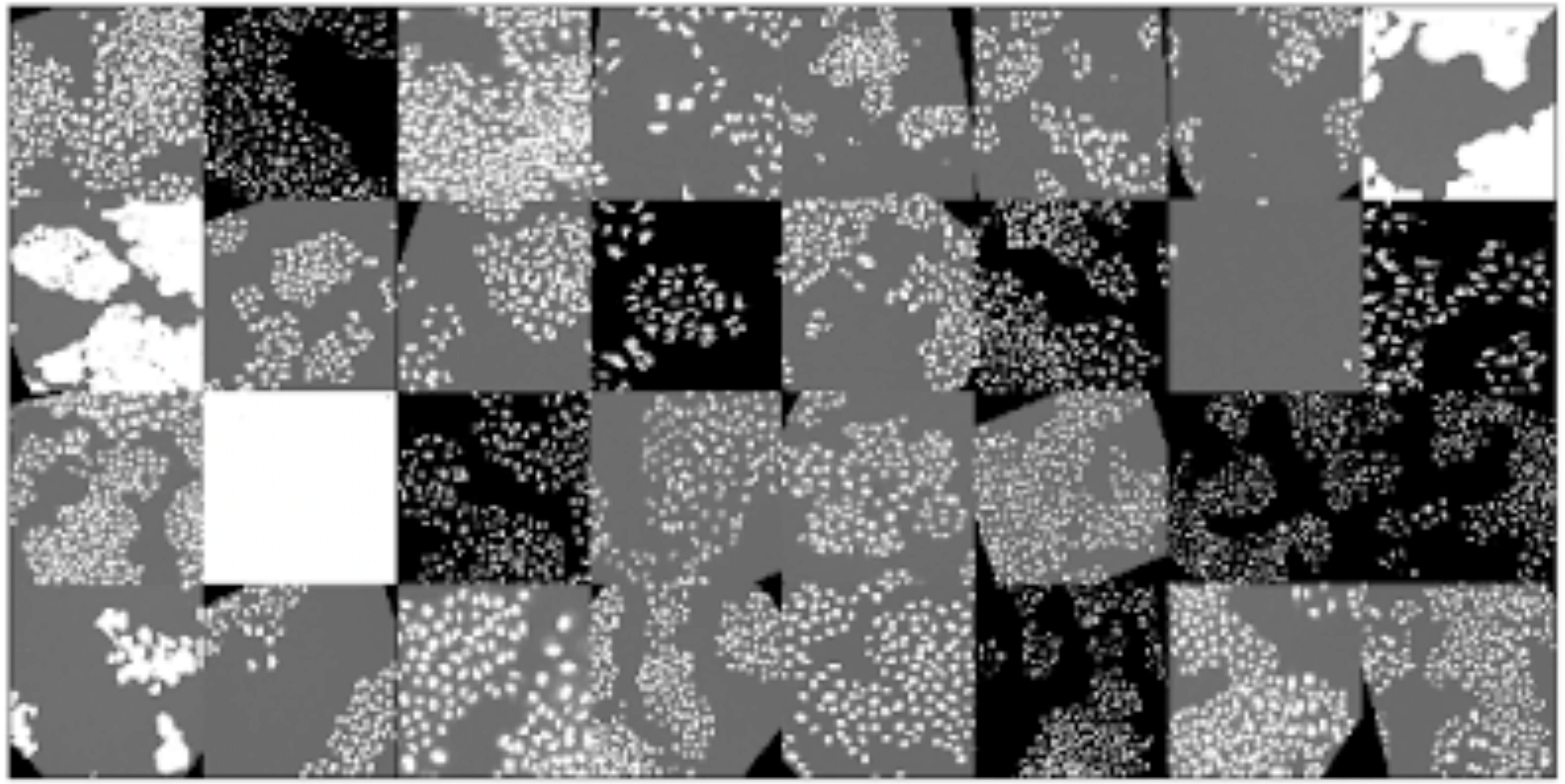


	T_1	T_2	T_3	...	T_N
I_1	$I_1 \cdot T_1$	$I_1 \cdot T_2$	$I_1 \cdot T_3$	...	$I_1 \cdot T_N$
I_2	$I_2 \cdot T_1$	$I_2 \cdot T_2$	$I_2 \cdot T_3$	...	$I_2 \cdot T_N$
I_3	$I_3 \cdot T_1$	$I_3 \cdot T_2$	$I_3 \cdot T_3$	...	$I_3 \cdot T_N$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
I_N	$I_N \cdot T_1$	$I_N \cdot T_2$	$I_N \cdot T_3$	...	$I_N \cdot T_N$

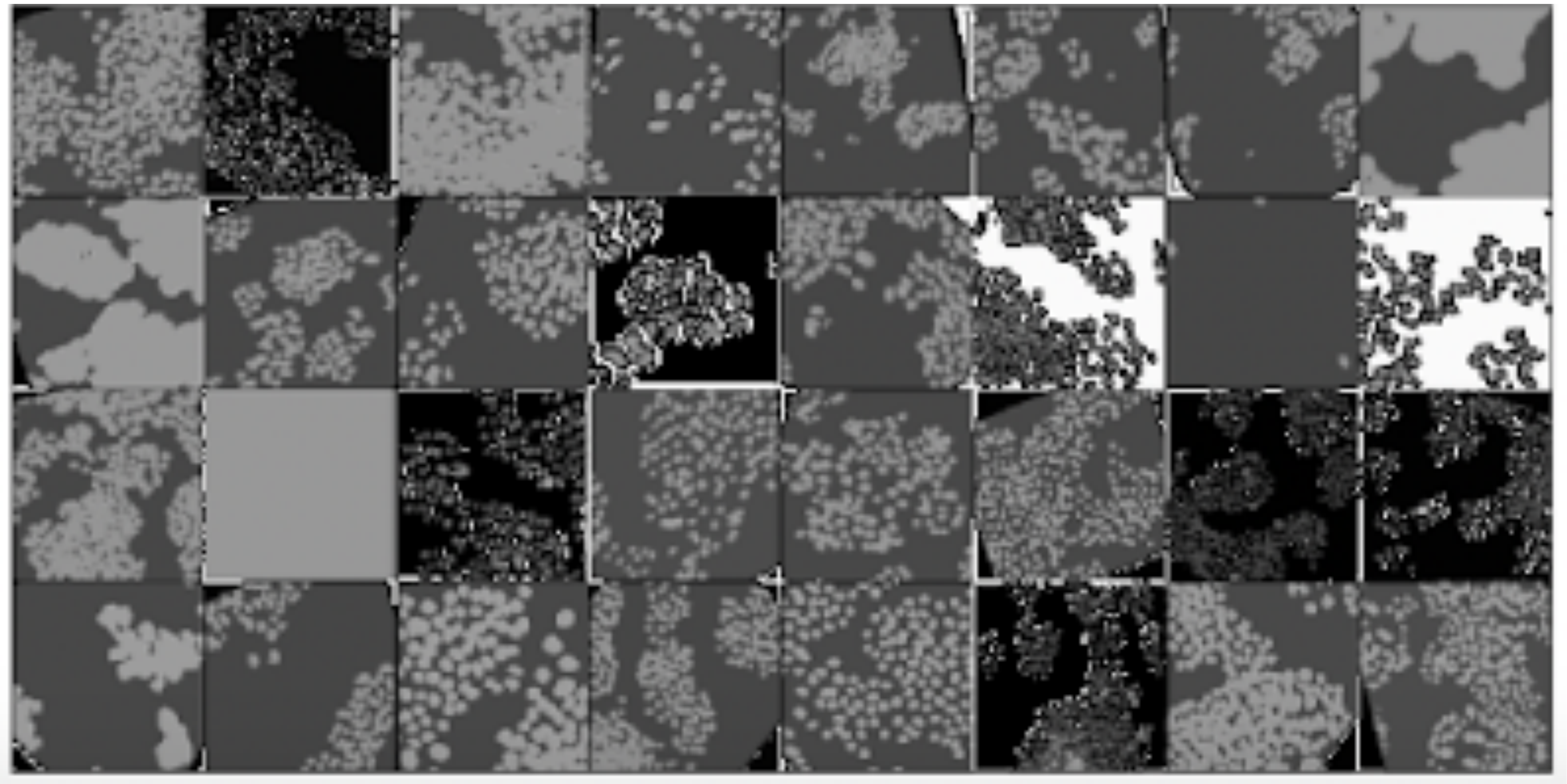
Visual assessment of model image reconstruction

Input and output of trained model

Image set from IDR00093



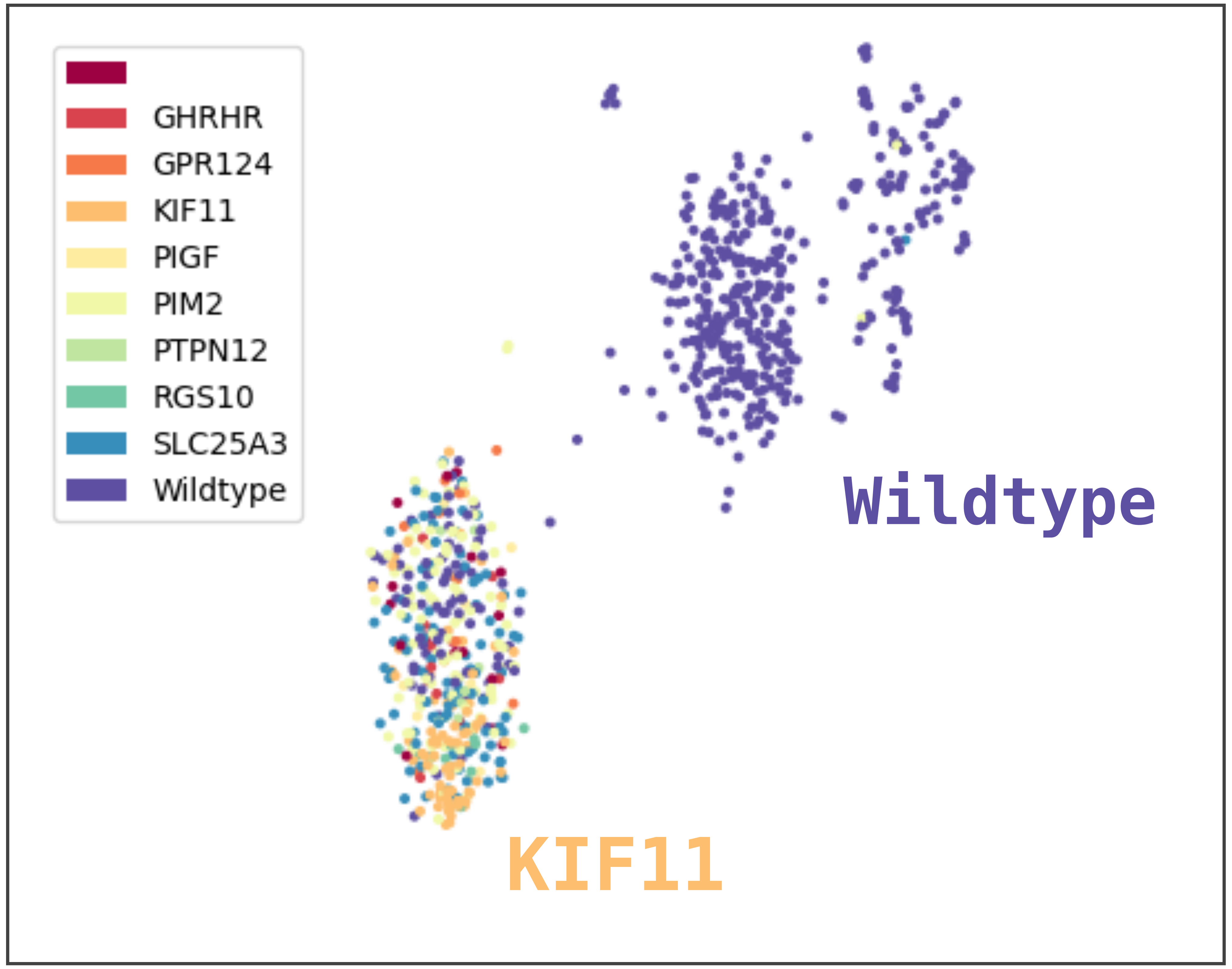
Reconstruction



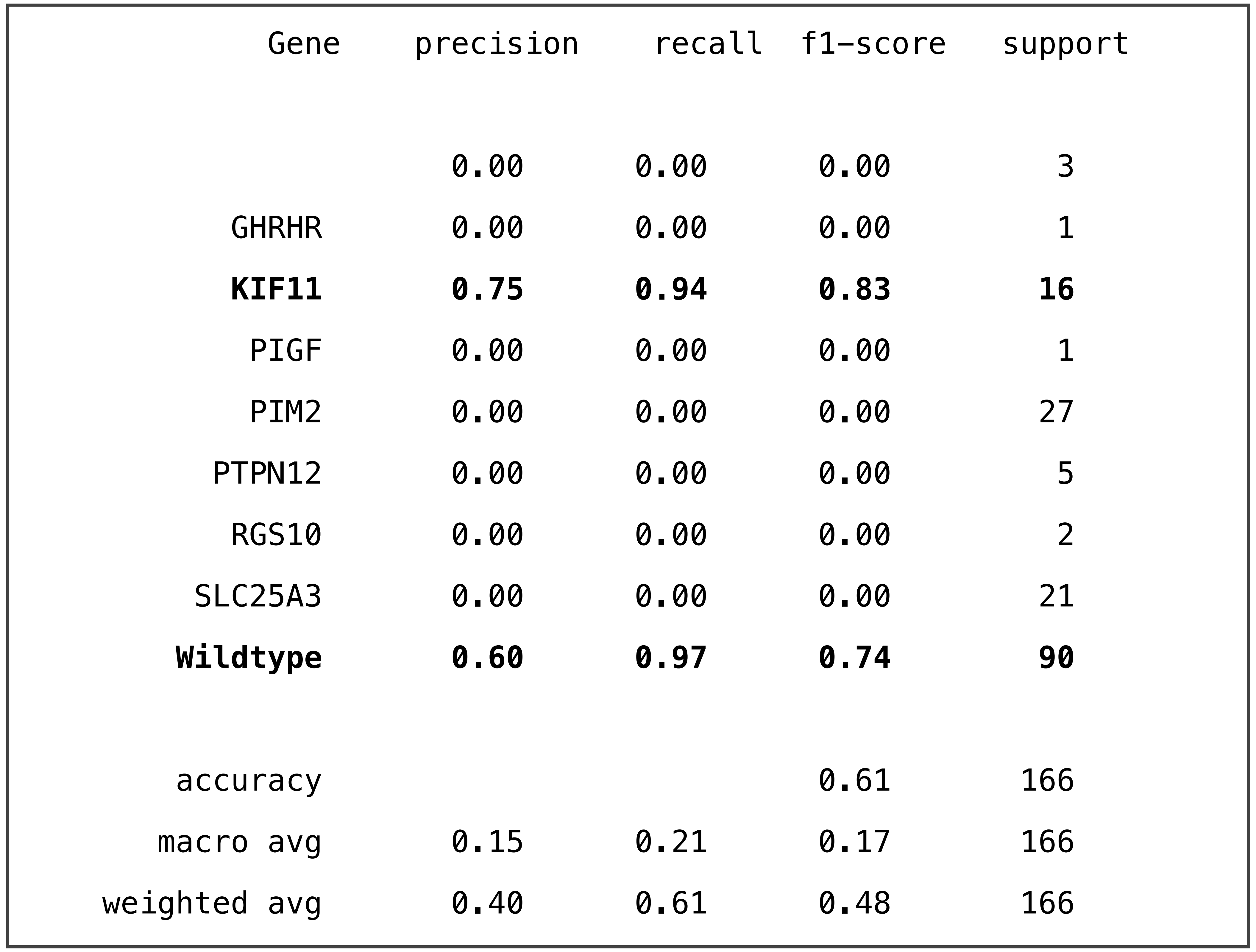
Model inference of gene labels in IDR00093

Using best model

UMAP of embeddings



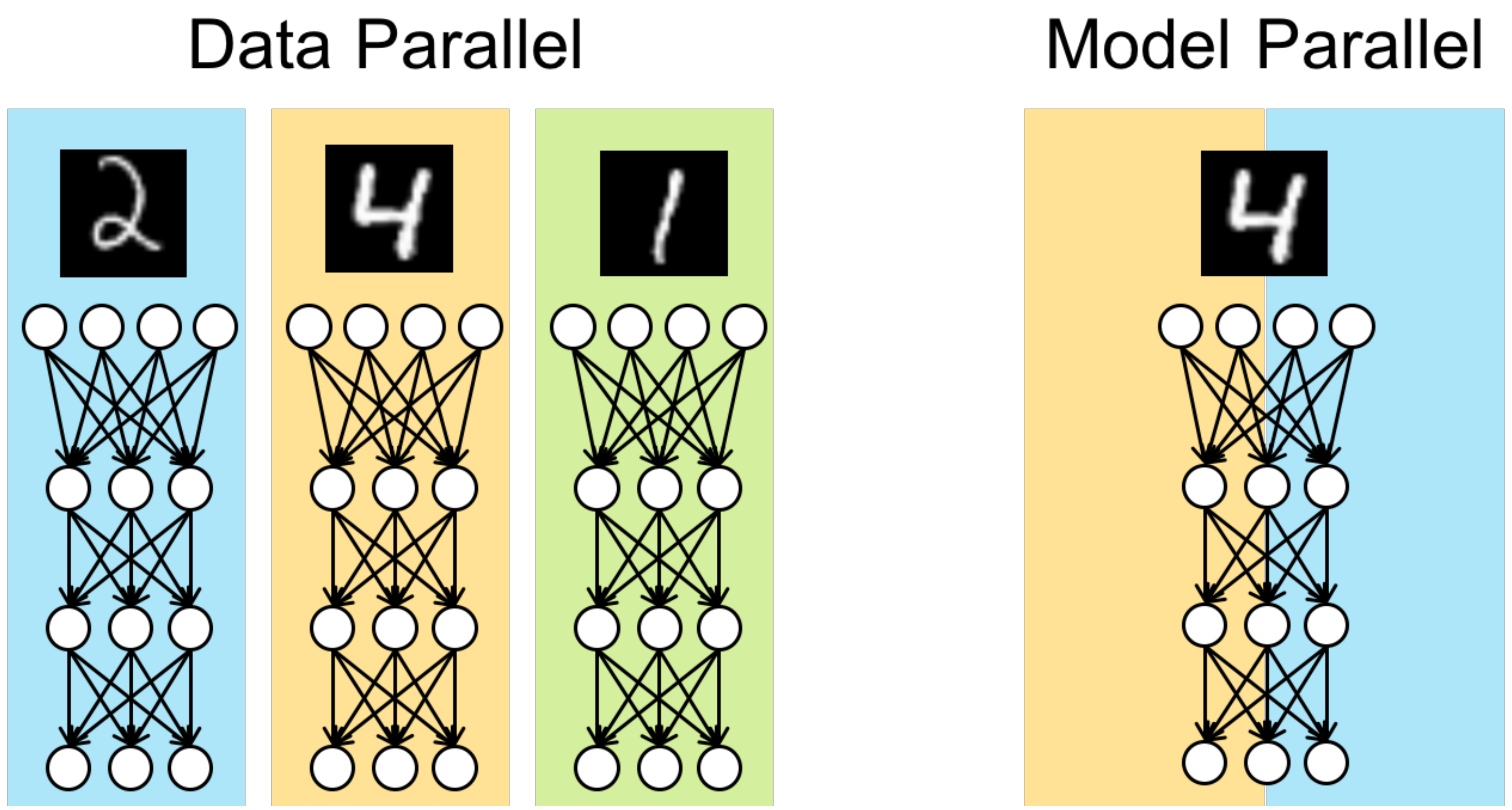
Random forest label prediction scores from subset embeddings



```
          Gene    precision    recall  f1-score   support


                       0.00      0.00      0.00         3
         GHRHR         0.00      0.00      0.00         1
         KIF11         0.75      0.94      0.83        16
          PIGF         0.00      0.00      0.00         1
          PIM2         0.00      0.00      0.00        27
        PTPN12         0.00      0.00      0.00         5
         RGS10         0.00      0.00      0.00         2
       SLC25A3         0.00      0.00      0.00        21
      Wildtype         0.60      0.97      0.74        90


      accuracy                             0.61       166
     macro avg         0.15      0.21      0.17       166
  weighted avg         0.40      0.61      0.48       166
```

Large model Training

- Data parallelism
  - Dataset is split per training node
  - Same model per node
  - Gradients can be accumulated
- Model parallelism
  - Model is split per training node
  - Gradients have to be synchronised quickly



Challenges + Tips

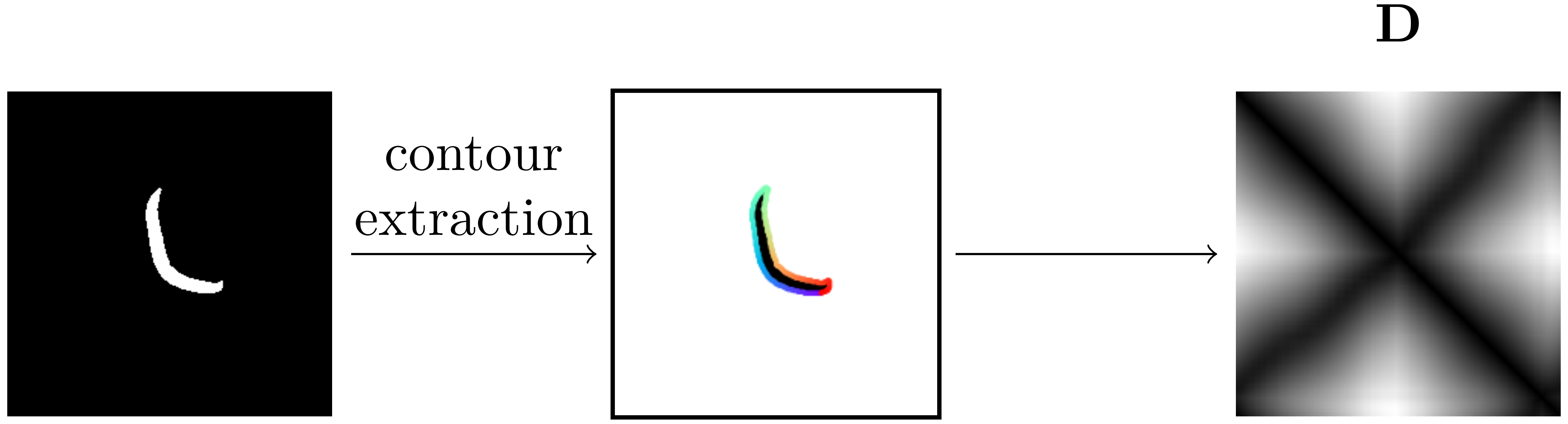
- Larger batches are always better
  - They converge faster
  - Choose largest that fits in memory
- CPU bottlenecks lead to GPU bottlenecks
  - Raw tiffs are slow, pngs are faster, zarr loading also slow

- GPU bottlenecks
  - 1 device per node is bad for model parallelism
  - Generally we see diminishing returns on speed for model sharding and data parallelism
- Hyperparameter tuning is embarrassingly parallel

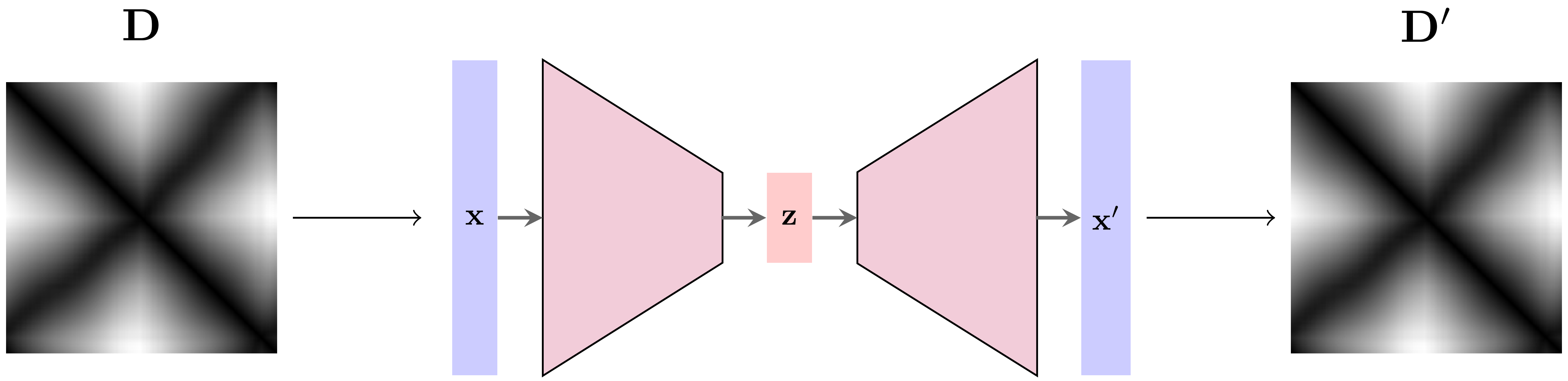
https://lightning.ai/docs/pytorch/stable/advanced/training_tricks.html

Biological applications

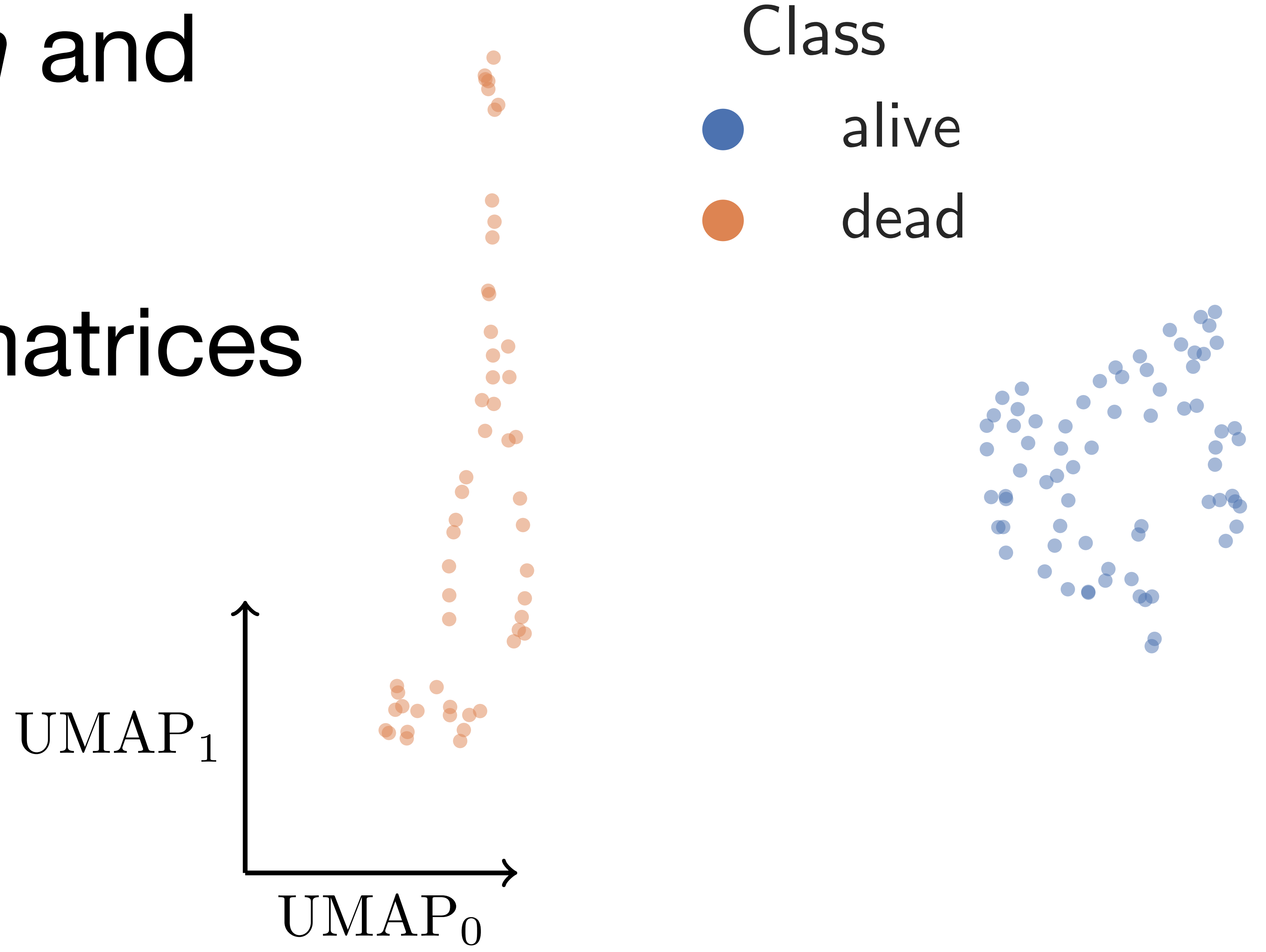
shape_embed

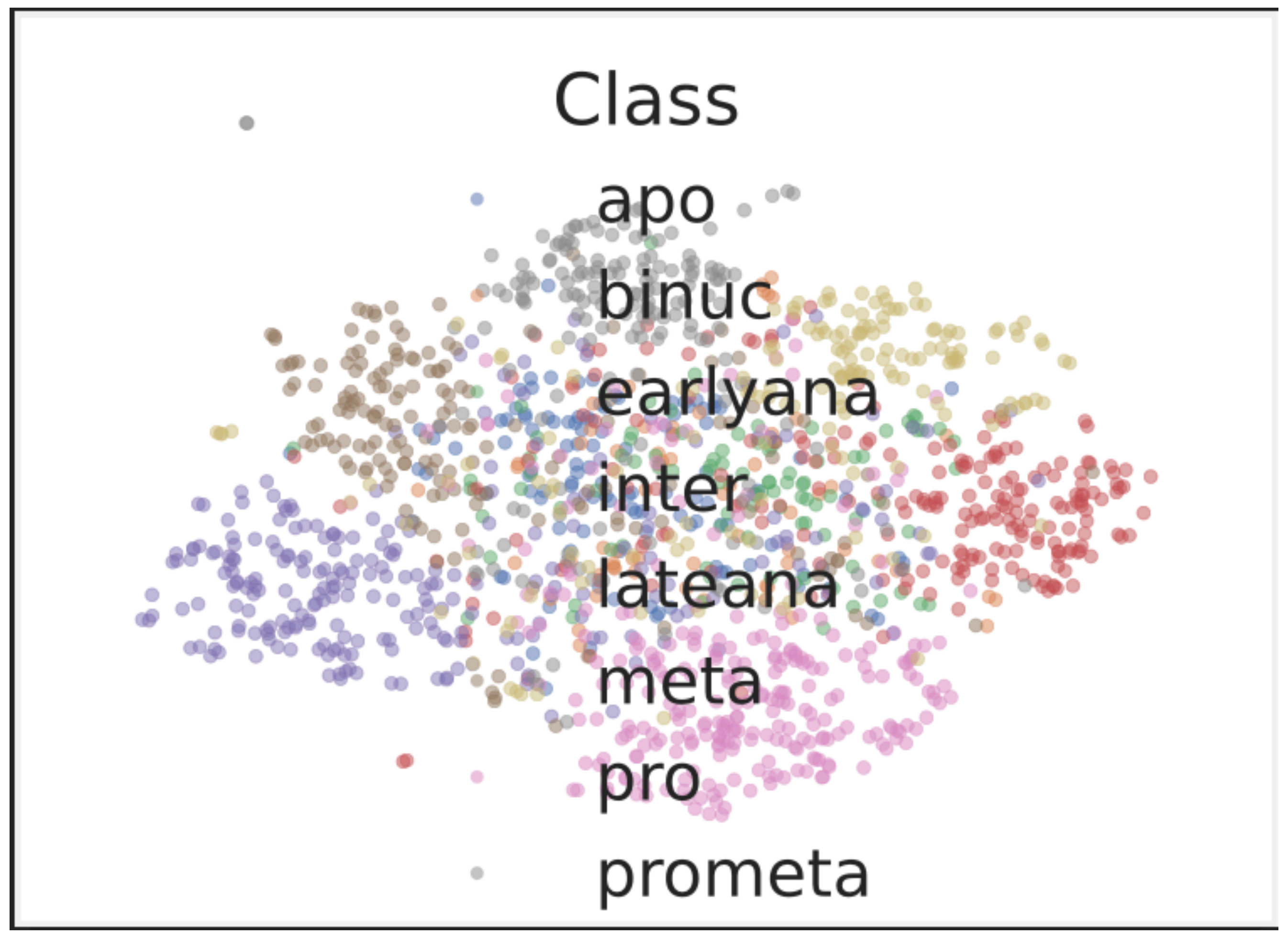


(a) Generation of distance matrix from a segmentation mask.



- Use distance matrix representation of masks/ contours
    - Agnostic to ***rotation*** and ***translation***
- Feed those distance matrices into the model
- Generate shape representation

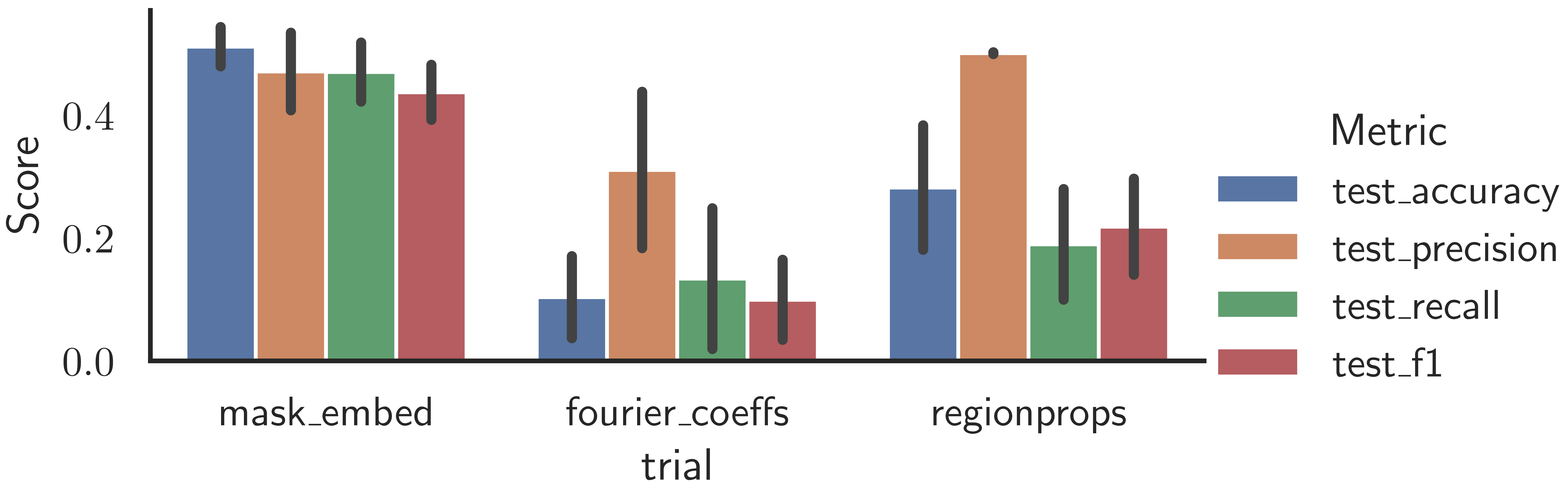


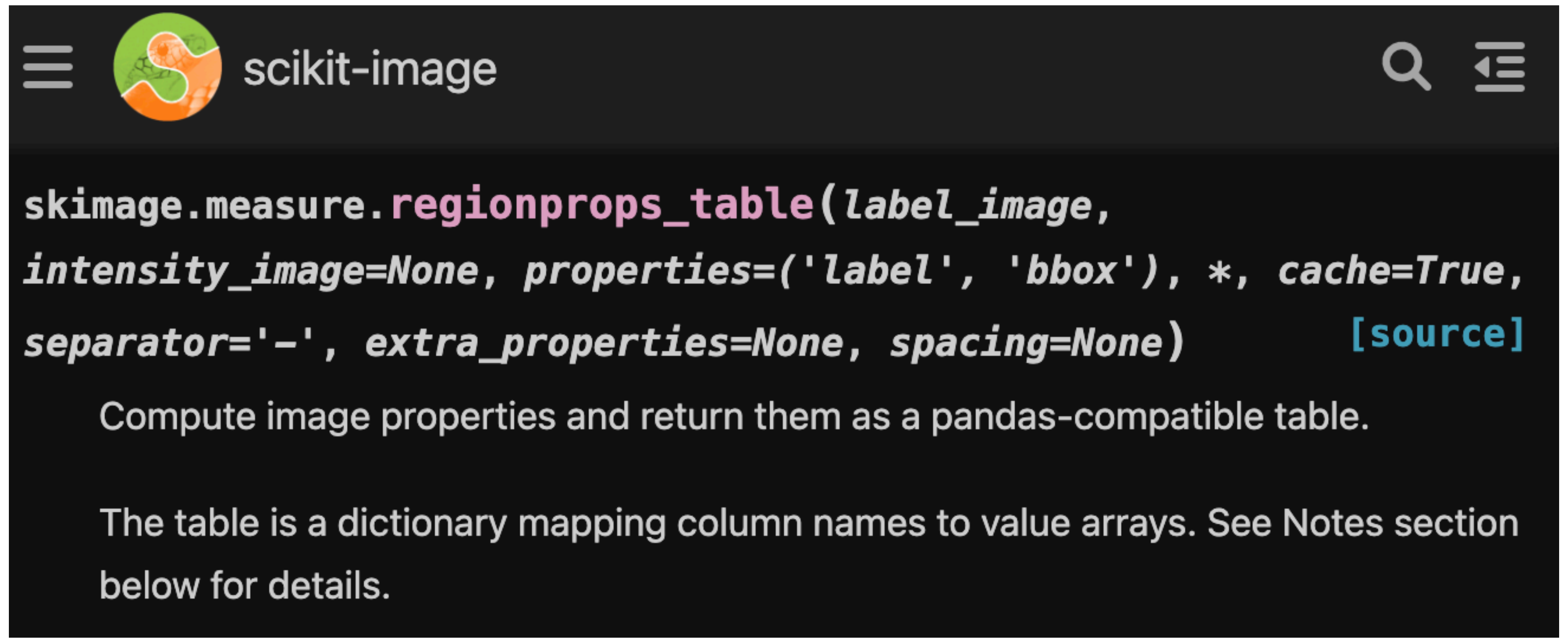


shape_embed results

Classification power

- Scoring with RandomForest
- Region props = [size, extent etc.]
- Broadly outperforms classical approaches

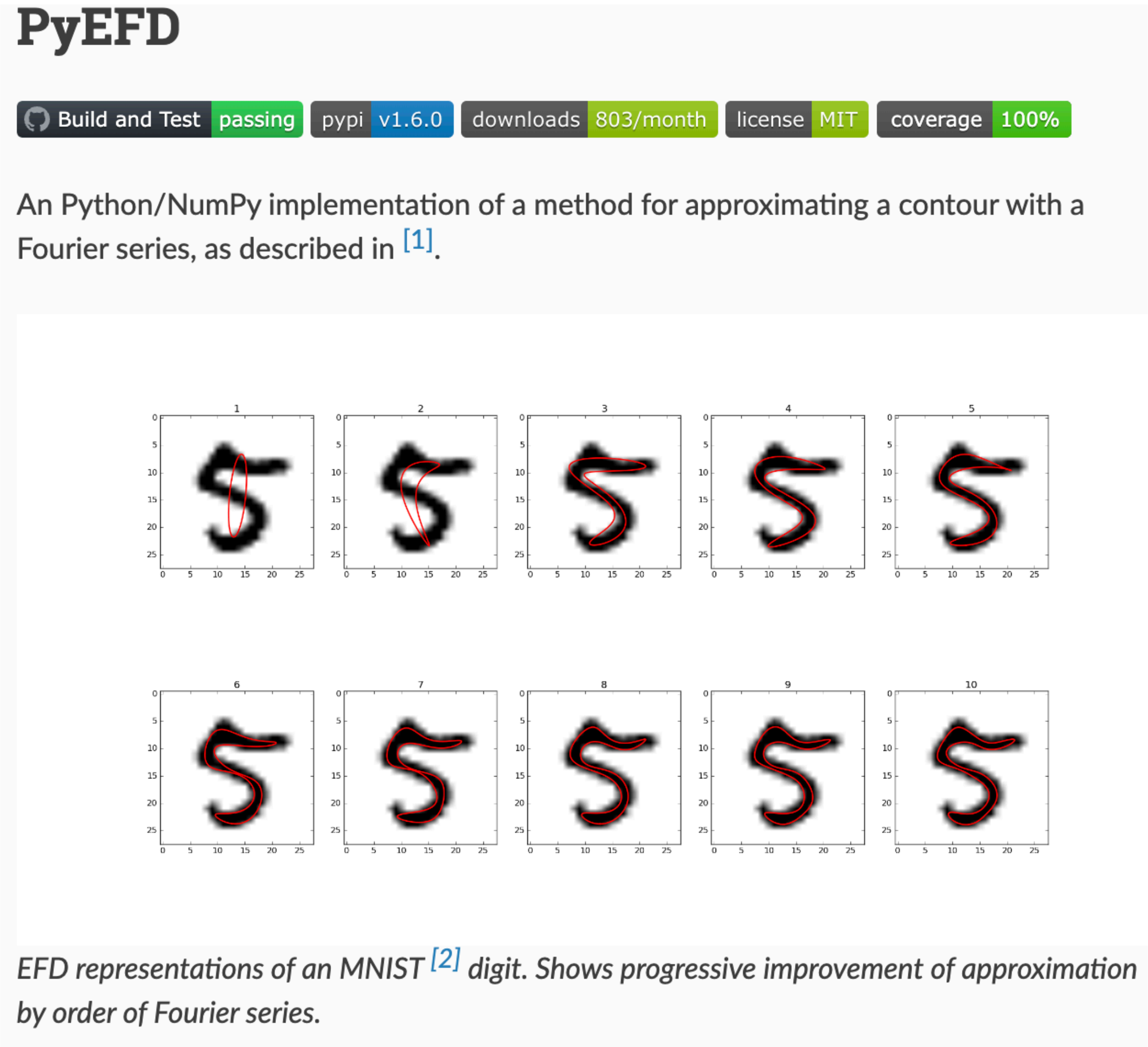




```
skimage.measure.regionprops_table(label_image, intensity_image=None, properties=('label', 'bbox'), *, cache=True, separator='-', extra_properties=None, spacing=None)    [source]
```

Compute image properties and return them as a pandas-compatible table.

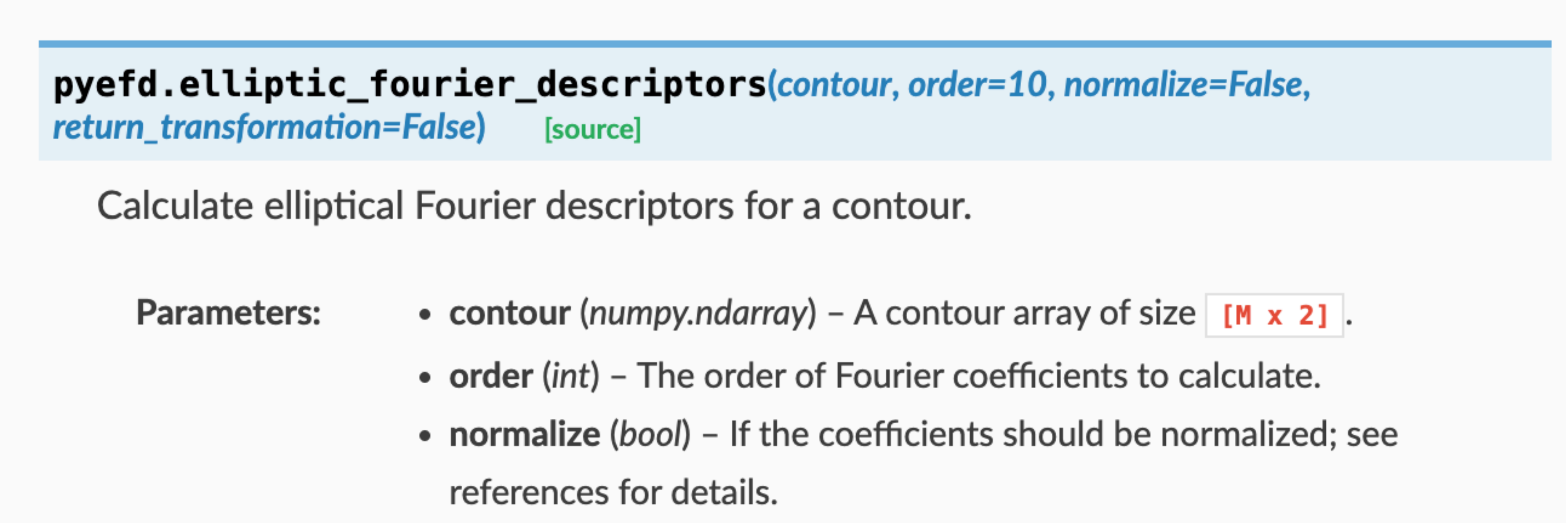
The table is a dictionary mapping column names to value arrays. See Notes section below for details.



PyEFD

An Python/NumPy implementation of a method for approximating a contour with a Fourier series, as described in [1].

EFD representations of an MNIST [2] digit. Shows progressive improvement of approximation by order of Fourier series.



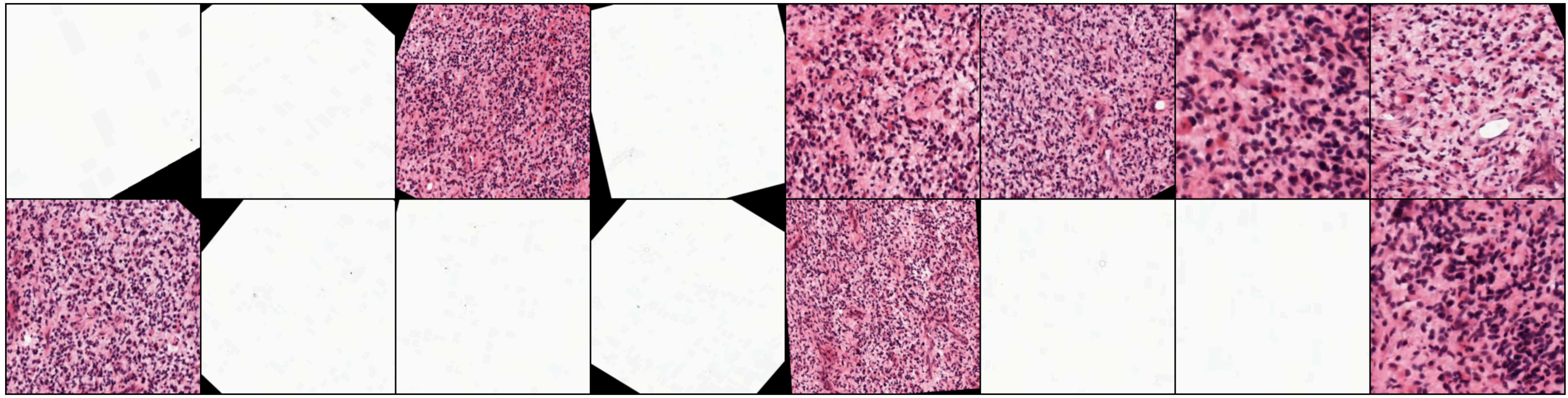
```
pyefd.elliptic_fourier_descriptors(contour, order=10, normalize=False, return_transformation=False)    [source]
```

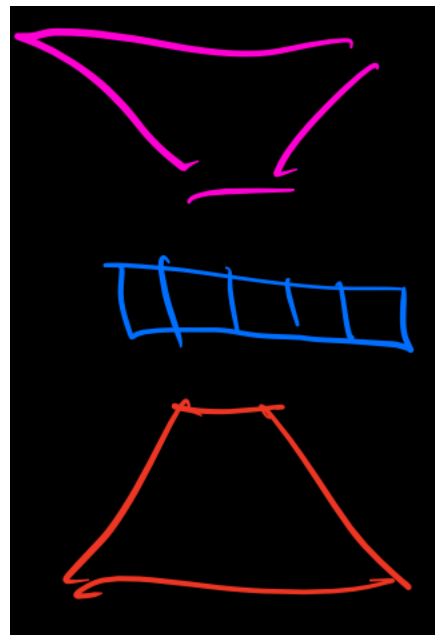
Calculate elliptical Fourier descriptors for a contour.

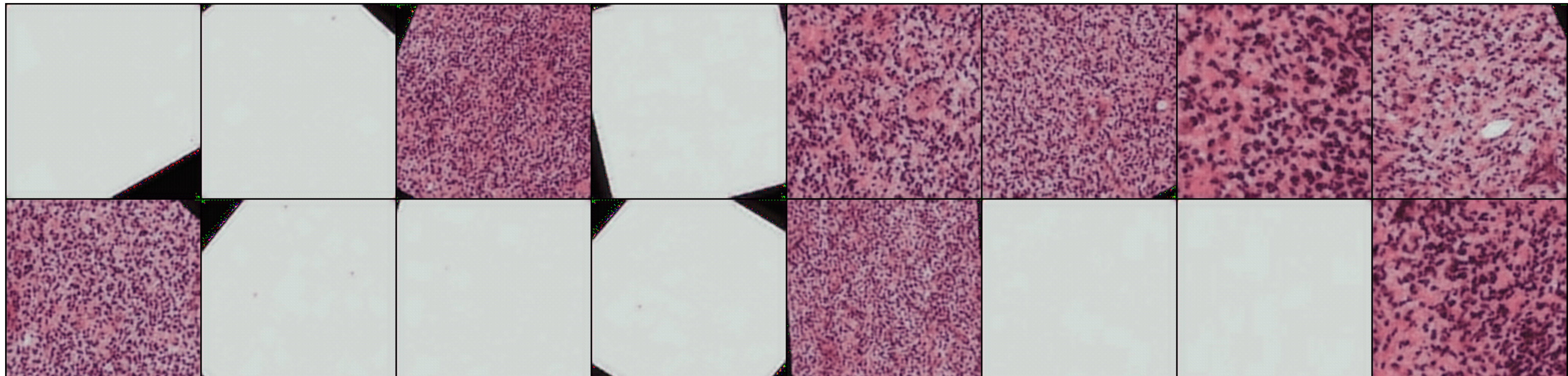
Parameters:
- **contour** (*numpy.ndarray*) – A contour array of size `[M x 2]`.
- **order** (*int*) – The order of Fourier coefficients to calculate.
- **normalize** (*bool*) – If the coefficients should be normalized; see references for details.

Histopathology patch similarity

- Large histopathology image
- Autoencoder encodes patches to patches
- Use latent representation and euclidean distance to find similar patches





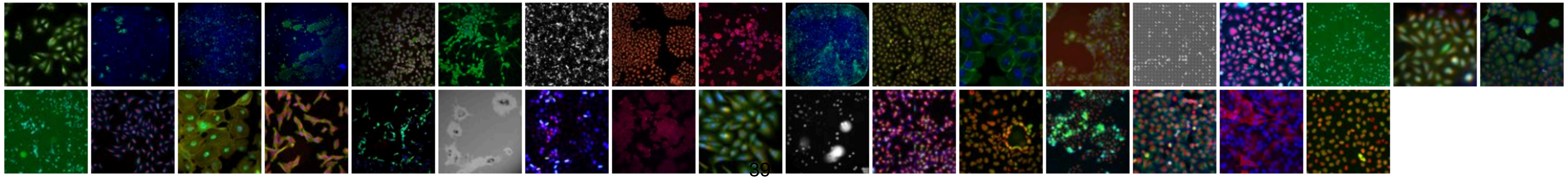


Future work

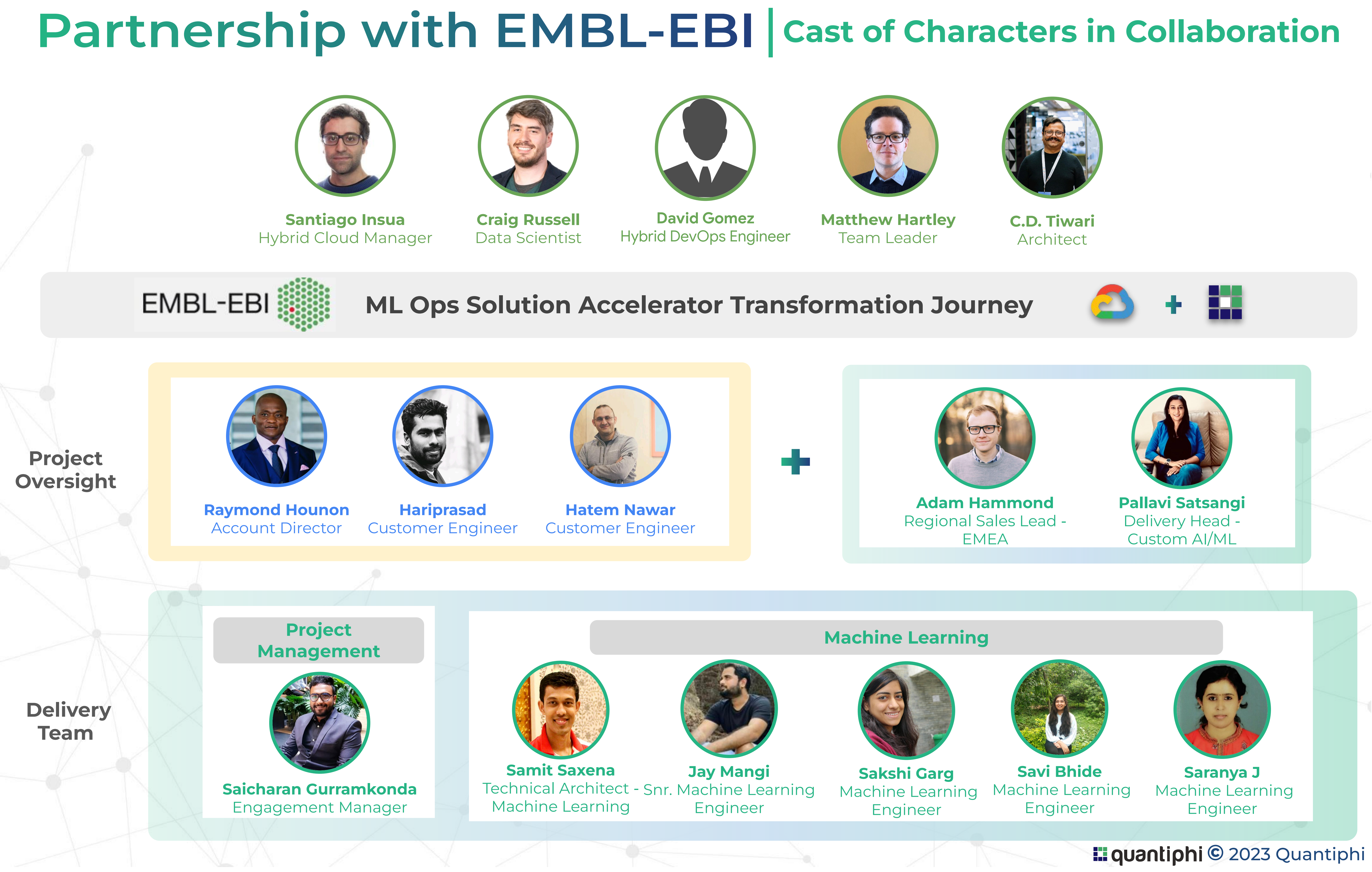
- Release challenge datasets
  - Pre-training + labelled
- MultiModal learning
  - Text - Dyes are extremely important
    - Metadata, papers etc
  - Shape - segmentations exist for some data

- More data - Include BIA
  - Larger pretraining
- More contrastive learning
  - More useful in finetuning
- Beta user public release
- More backend models
  - Transformers, HuggingFace models

High-content screening (human)



Acknowledgements



Partnership with EMBL-EBI | Cast of Characters in Collaboration

Santiago Insua
Hybrid Cloud Manager

Craig Russell
Data Scientist

David Gomez
Hybrid DevOps Engineer

Matthew Hartley
Team Leader

C.D. Tiwari
Architect

EMBL-EBI ML Ops Solution Accelerator Transformation Journey +

Project Oversight

Raymond Hounon
Account Director

Hariprasad
Customer Engineer

Hatem Nawar
Customer Engineer

+

Adam Hammond
Regional Sales Lead - EMEA

Pallavi Satsangi
Delivery Head - Custom AI/ML

Delivery Team

Project Management

Saicharan Gurramkonda
Engagement Manager

Machine Learning

Samit Saxena
Technical Architect - Machine Learning

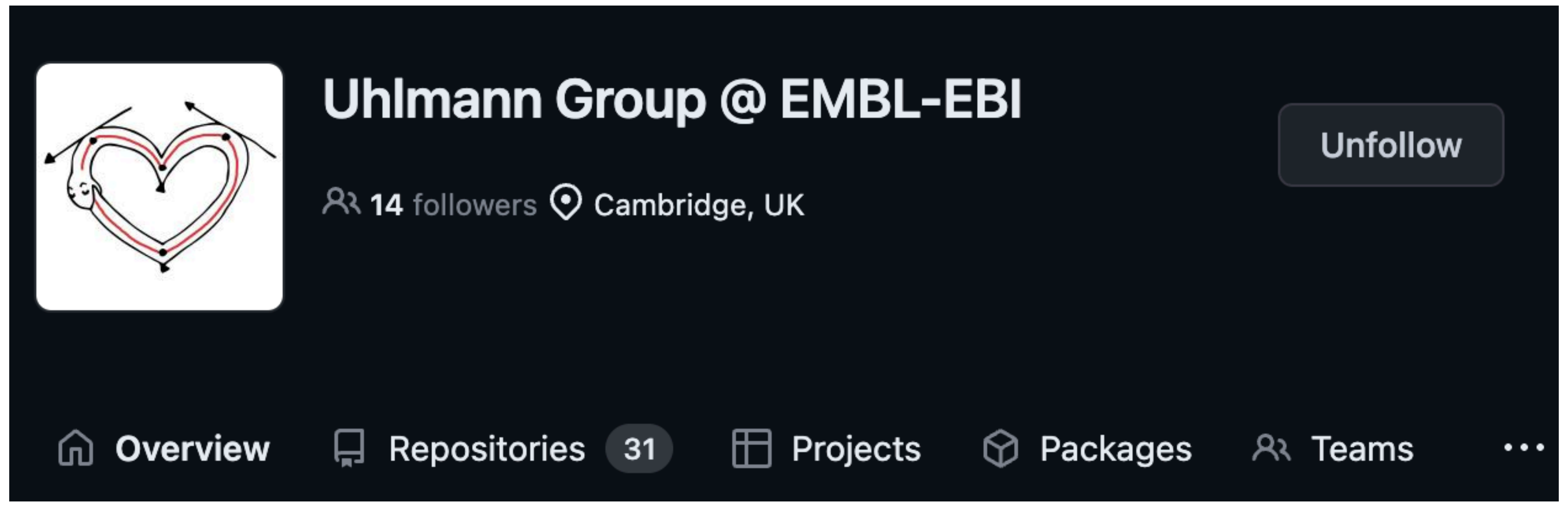
Jay Mangi
Snr. Machine Learning Engineer

Sakshi Garg
Machine Learning Engineer

Savi Bhide
Machine Learning Engineer

Saranya J
Machine Learning Engineer

quantiphi © 2023 Quantiphi

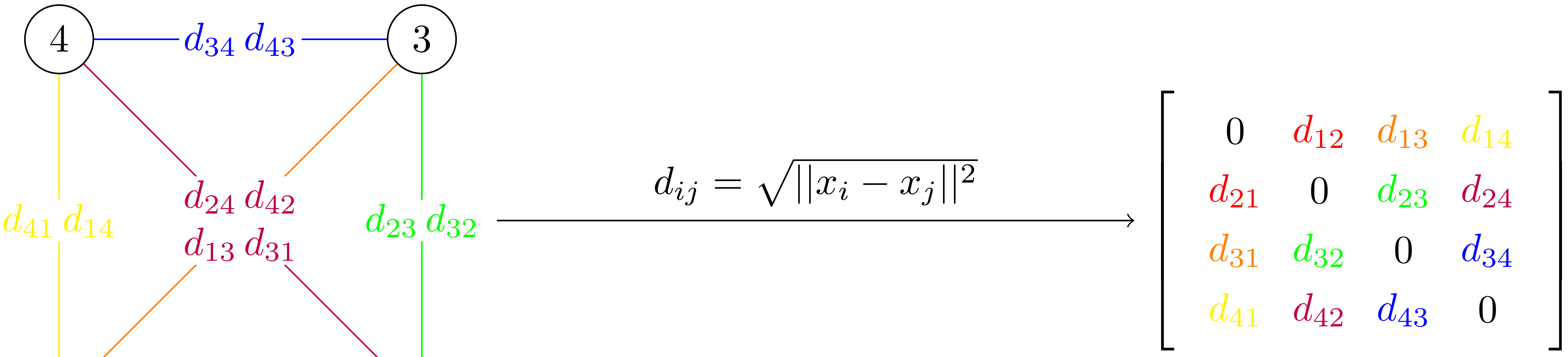


Uhlmann Group @ EMBL-EBI

14 followers Cambridge, UK

Distance matrices

- Invariant to:
  - Rotationally invariant
  - *Scale invariant*
    - With Frobenius norm
  - Translation invariance
  - Reflection invariance
  - Single shape prior



$$d_{ij} = \sqrt{||x_i - x_j||^2}$$

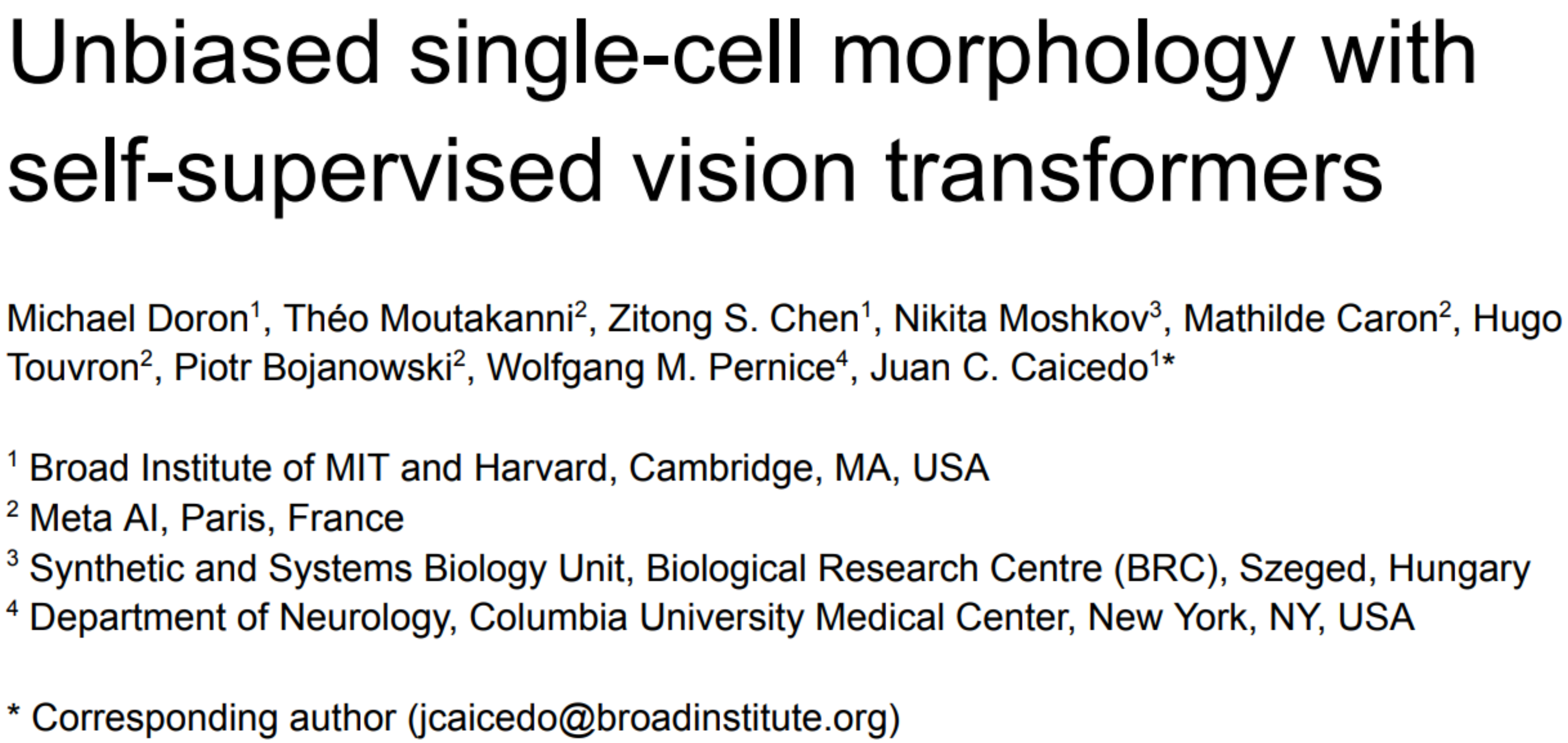
$$\begin{bmatrix} 0 & d_{12} & d_{13} & d_{14} \\ d_{21} & 0 & d_{23} & d_{24} \\ d_{31} & d_{32} & 0 & d_{34} \\ d_{41} & d_{42} & d_{43} & 0 \end{bmatrix}$$

Distance matrices

Indexation invariance

- Achieved through augmentation
- Possible number of augmentations
    - $\|\mathrm{diag}(\mathbf{D})\|$

$$\begin{tikzcd}[ampersand replacement=\&]
\begin{bmatrix} 0 & d_{12} & d_{13} & d_{14} \\ d_{21} & 0 & d_{23} & d_{24} \\ d_{31} & d_{32} & 0 & d_{34} \\ d_{41} & d_{42} & d_{43} & 0 \end{bmatrix} \arrow[r] \& \begin{bmatrix} 0 & d_{41} & d_{42} & d_{43} \\ d_{14} & 0 & d_{12} & d_{13} \\ d_{24} & d_{21} & 0 & d_{23} \\ d_{34} & d_{31} & d_{32} & 0 \end{bmatrix} \arrow[d] \\
\begin{bmatrix} 0 & d_{23} & d_{24} & d_{21} \\ d_{32} & 0 & d_{34} & d_{31} \\ d_{42} & d_{43} & 0 & d_{41} \\ d_{12} & d_{13} & d_{14} & 0 \end{bmatrix} \arrow[u] \& \begin{bmatrix} 0 & d_{34} & d_{31} & d_{32} \\ d_{43} & 0 & d_{41} & d_{42} \\ d_{13} & d_{14} & 0 & d_{12} \\ d_{23} & d_{24} & d_{21} & 0 \end{bmatrix} \arrow[l]
\end{tikzcd}$$

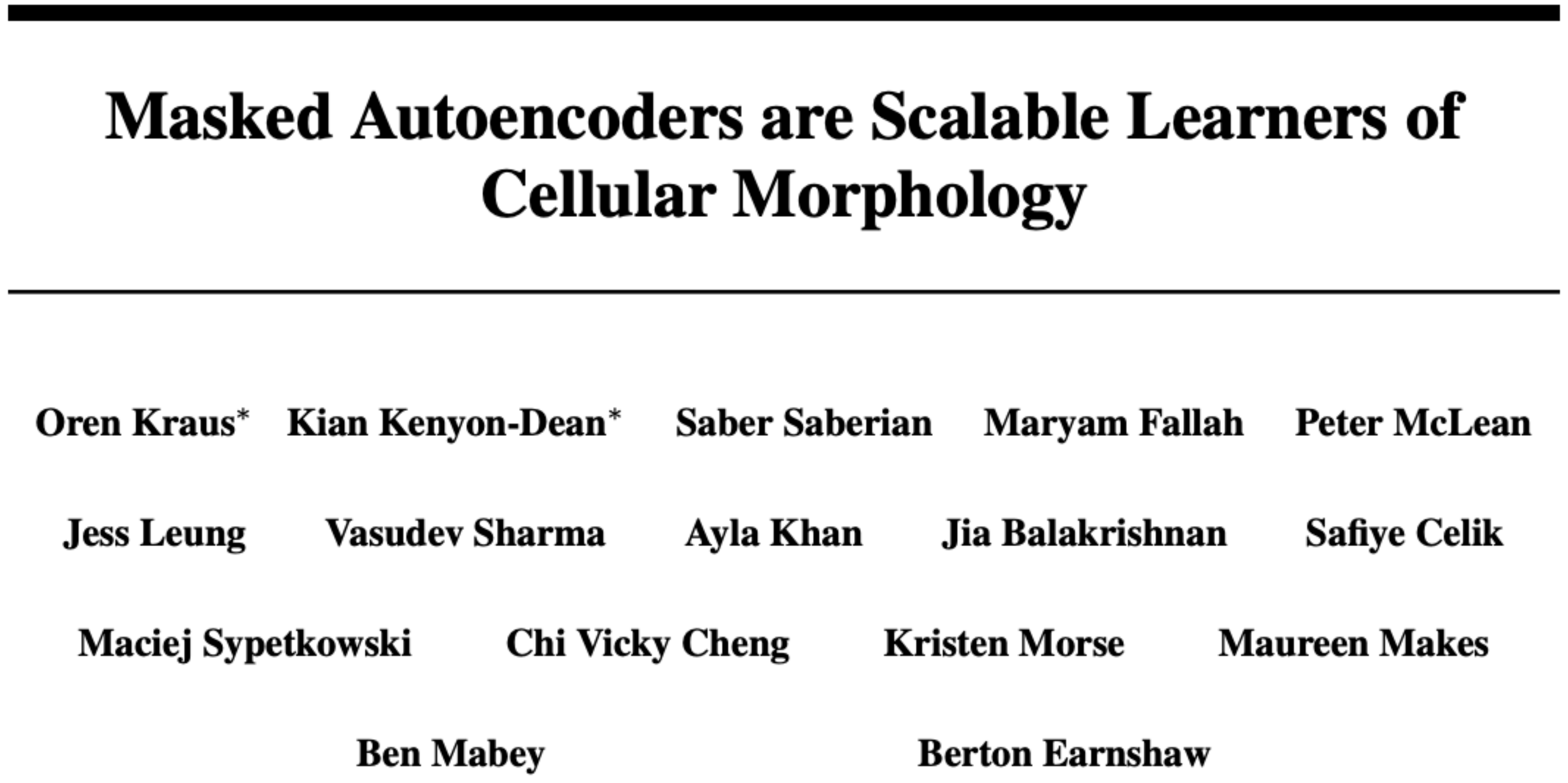


Unbiased single-cell morphology with self-supervised vision transformers

Michael Doron[1], Théo Moutakanni[2], Zitong S. Chen[1], Nikita Moshkov[3], Mathilde Caron[2], Hugo Touvron[2], Piotr Bojanowski[2], Wolfgang M. Pernice[4], Juan C. Caicedo[1]*

[1] Broad Institute of MIT and Harvard, Cambridge, MA, USA
[2] Meta AI, Paris, France
[3] Synthetic and Systems Biology Unit, Biological Research Centre (BRC), Szeged, Hungary
[4] Department of Neurology, Columbia University Medical Center, New York, NY, USA

* Corresponding author (jcaicedo@broadinstitute.org)



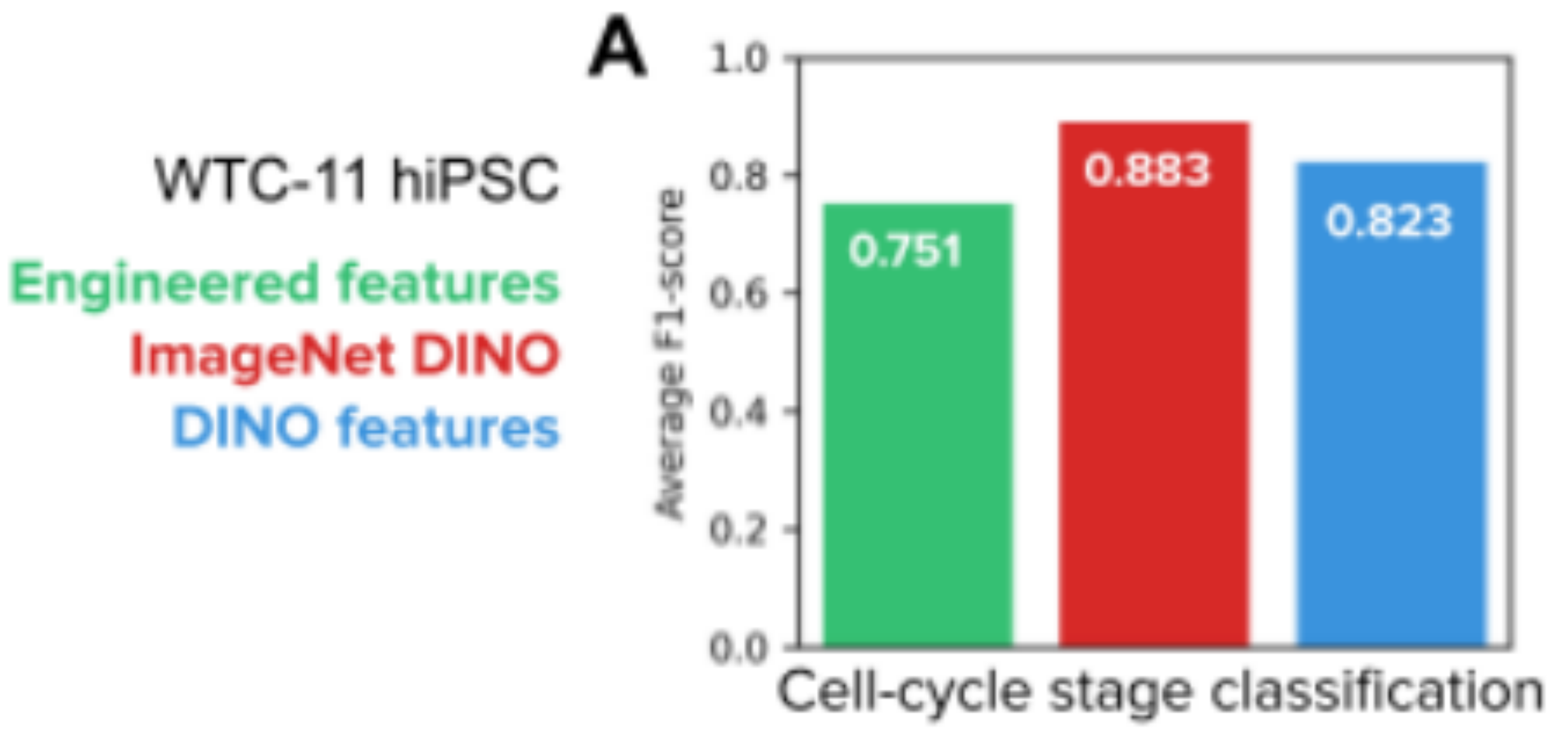
Masked Autoencoders are Scalable Learners of Cellular Morphology

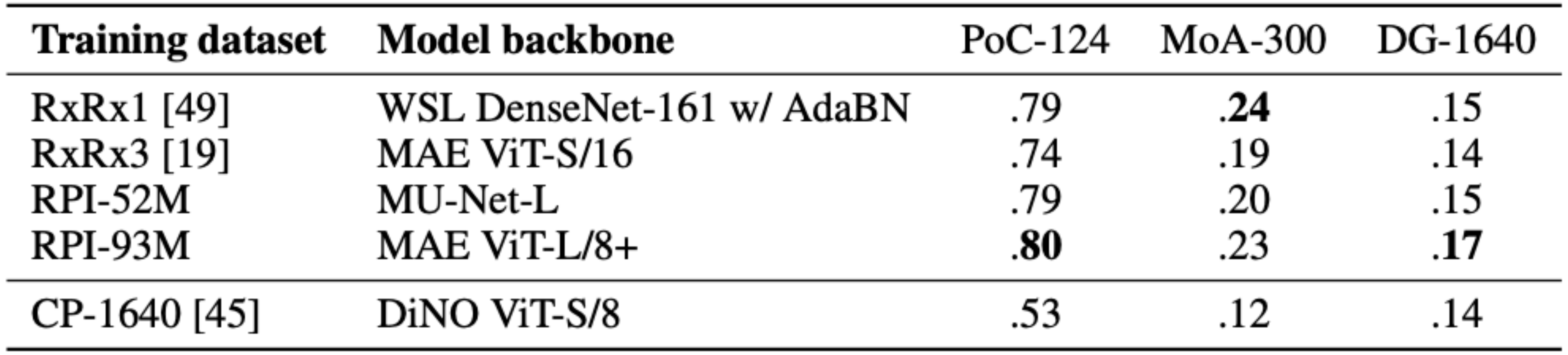
Oren Kraus* Kian Kenyon-Dean* Saber Saberian Maryam Fallah Peter McLean

Jess Leung Vasudev Sharma Ayla Khan Jia Balakrishnan Safiye Celik

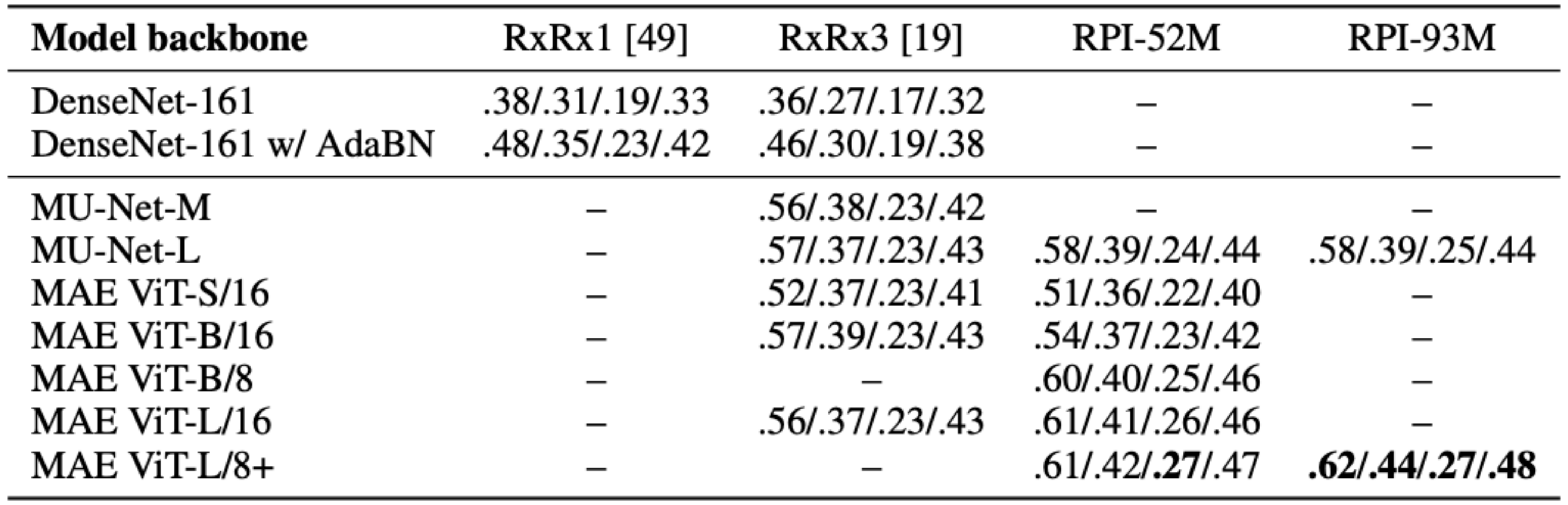
Maciej Sypetkowski Chi Vicky Cheng Kristen Morse Maureen Makes

Ben Mabey Berton Earnshaw

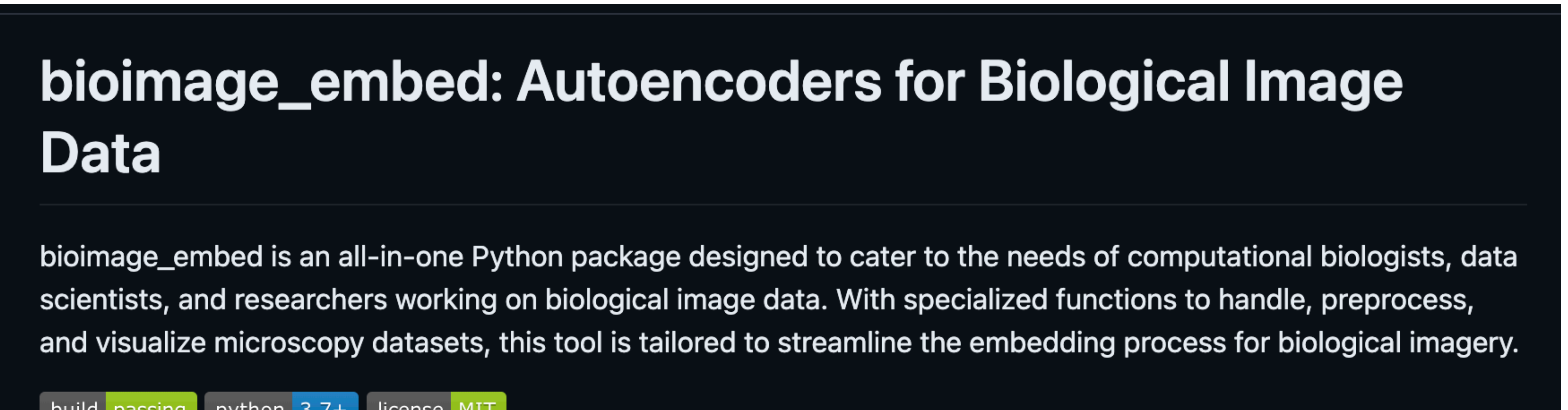




Training dataset	Model backbone	PoC-124	MoA-300	DG-1640
RxRx1 [49]	WSL DenseNet-161 w/ AdaBN	.79	**.24**	.15
RxRx3 [19]	MAE ViT-S/16	.74	.19	.14
RPI-52M	MU-Net-L	.79	.20	.15
RPI-93M	MAE ViT-L/8+	**.80**	.23	**.17**
CP-1640 [45]	DiNO ViT-S/8	.53	.12	.14



Model backbone	RxRx1 [49]	RxRx3 [19]	RPI-52M	RPI-93M
DenseNet-161	.38/.31/.19/.33	.36/.27/.17/.32	–	–
DenseNet-161 w/ AdaBN	.48/.35/.23/.42	.46/.30/.19/.38	–	–
MU-Net-M	–	.56/.38/.23/.42	–	–
MU-Net-L	–	.57/.37/.23/.43	.58/.39/.24/.44	.58/.39/.25/.44
MAE ViT-S/16	–	.52/.37/.23/.41	.51/.36/.22/.40	–
MAE ViT-B/16	–	.57/.39/.23/.43	.54/.37/.23/.42	–
MAE ViT-B/8	–	–	.60/.40/.25/.46	–
MAE ViT-L/16	–	.56/.37/.23/.43	.61/.41/.26/.46	–
MAE ViT-L/8+	–	–	.61/.42/**.27**/.47	**.62/.44/.27/.48**

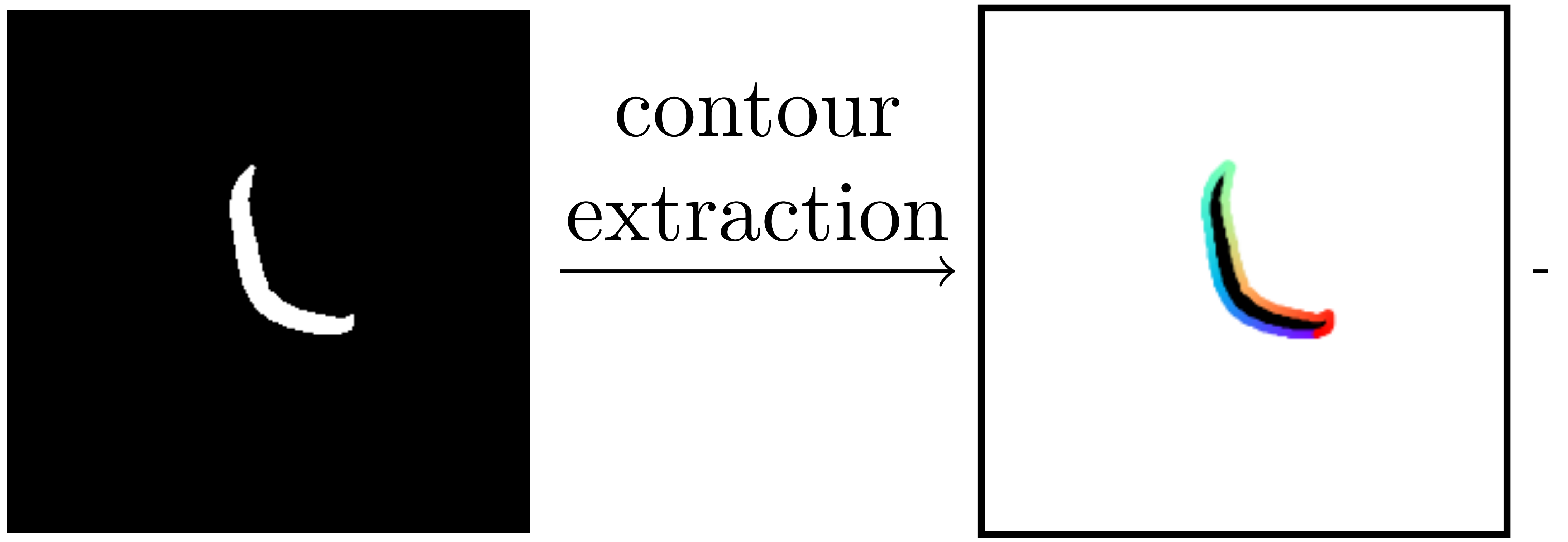


bioimage_embed: Autoencoders for Biological Image Data

bioimage_embed is an all-in-one Python package designed to cater to the needs of computational biologists, data scientists, and researchers working on biological image data. With specialized functions to handle, preprocess, and visualize microscopy datasets, this tool is tailored to streamline the embedding process for biological imagery.

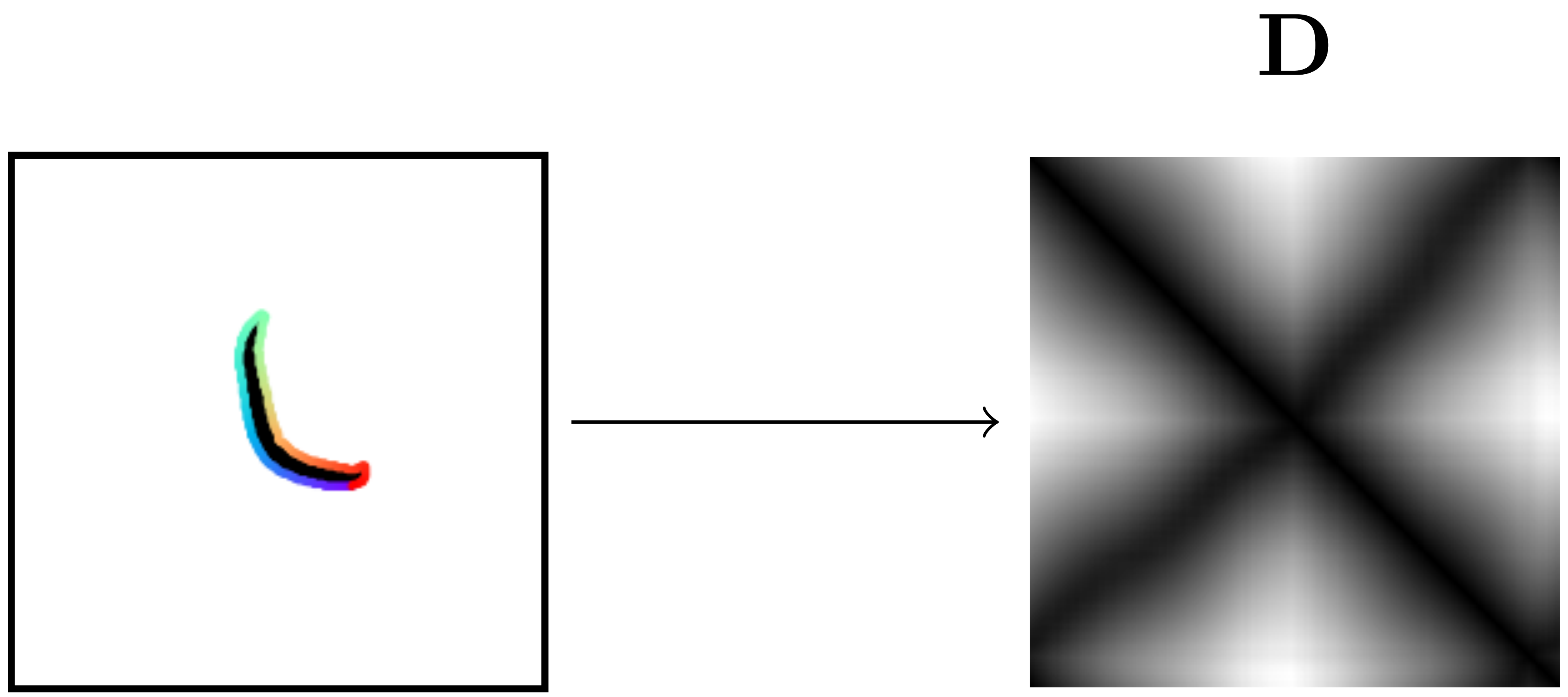
Contour extraction

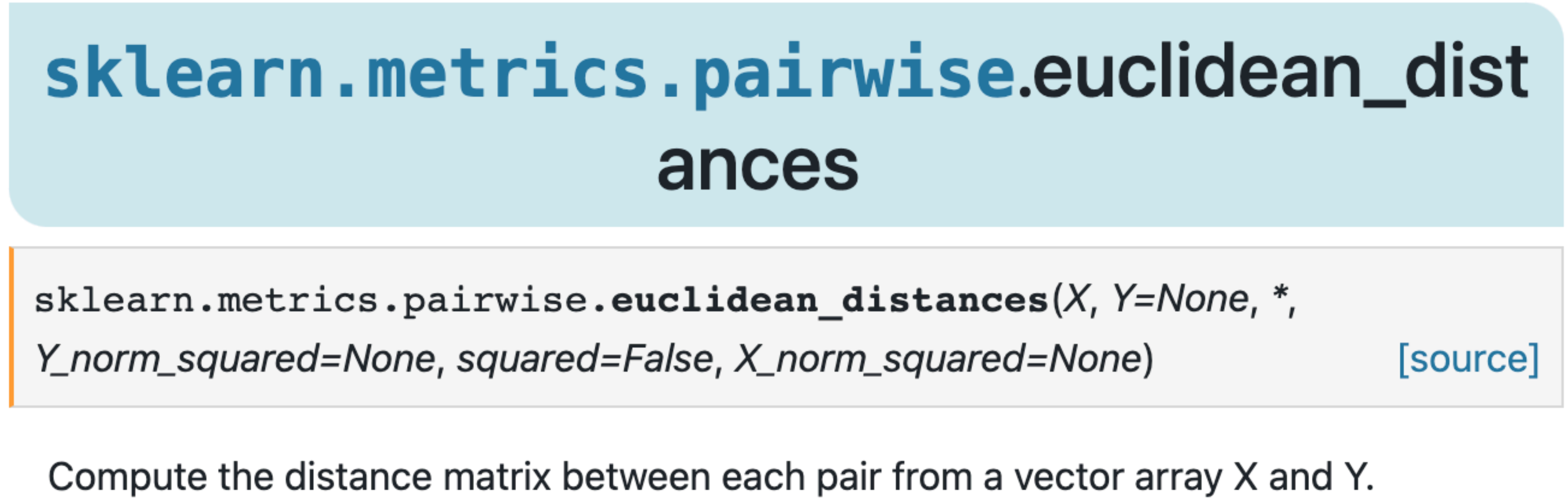
- Crop centroid
  - **Window size hyperparam**
- Marching squares
- Fit cubic spline
- Resample
  - Currently uses a stardist style ray casting
  - Should use spline resampling



Contour to distance matrix

- Resample to 256 points (hparam)
  - 256 is roughly 224 -> ImageNet size
  - Rainbow -> samples
- Normalise coords to window_size [!]
- Run euclidean distance on contour points
  - Use euclidean because operation can be inverted

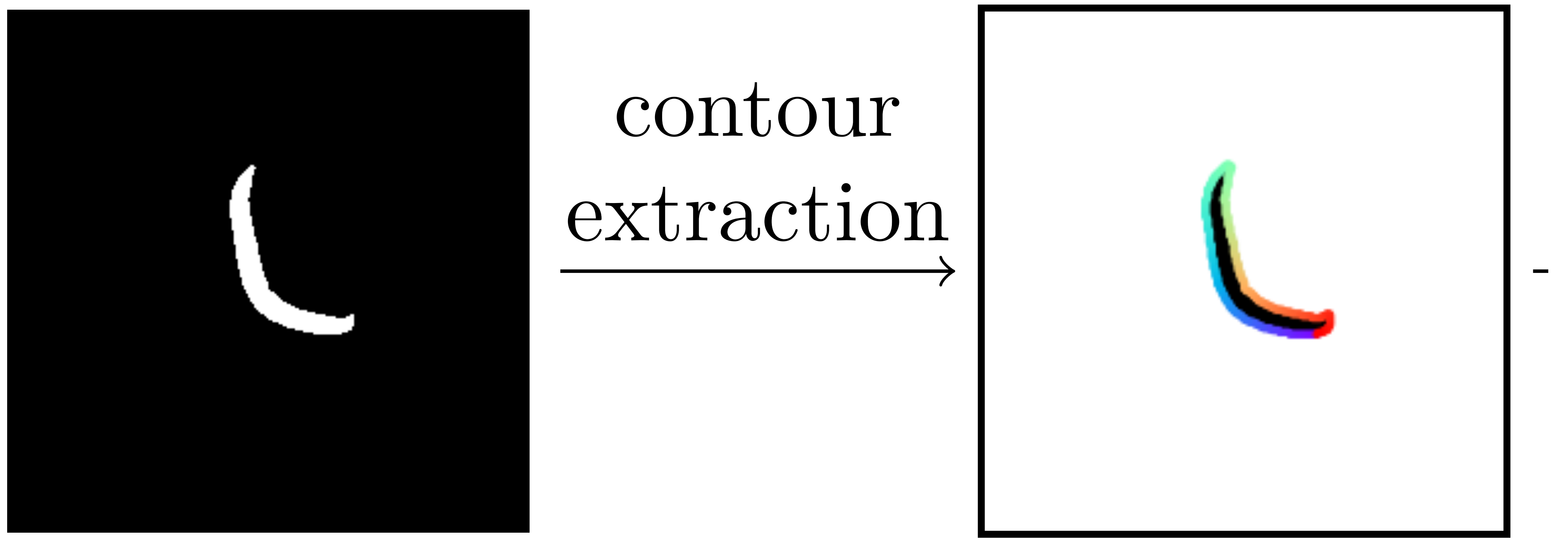




sklearn.metrics.pairwise.euclidean_distances

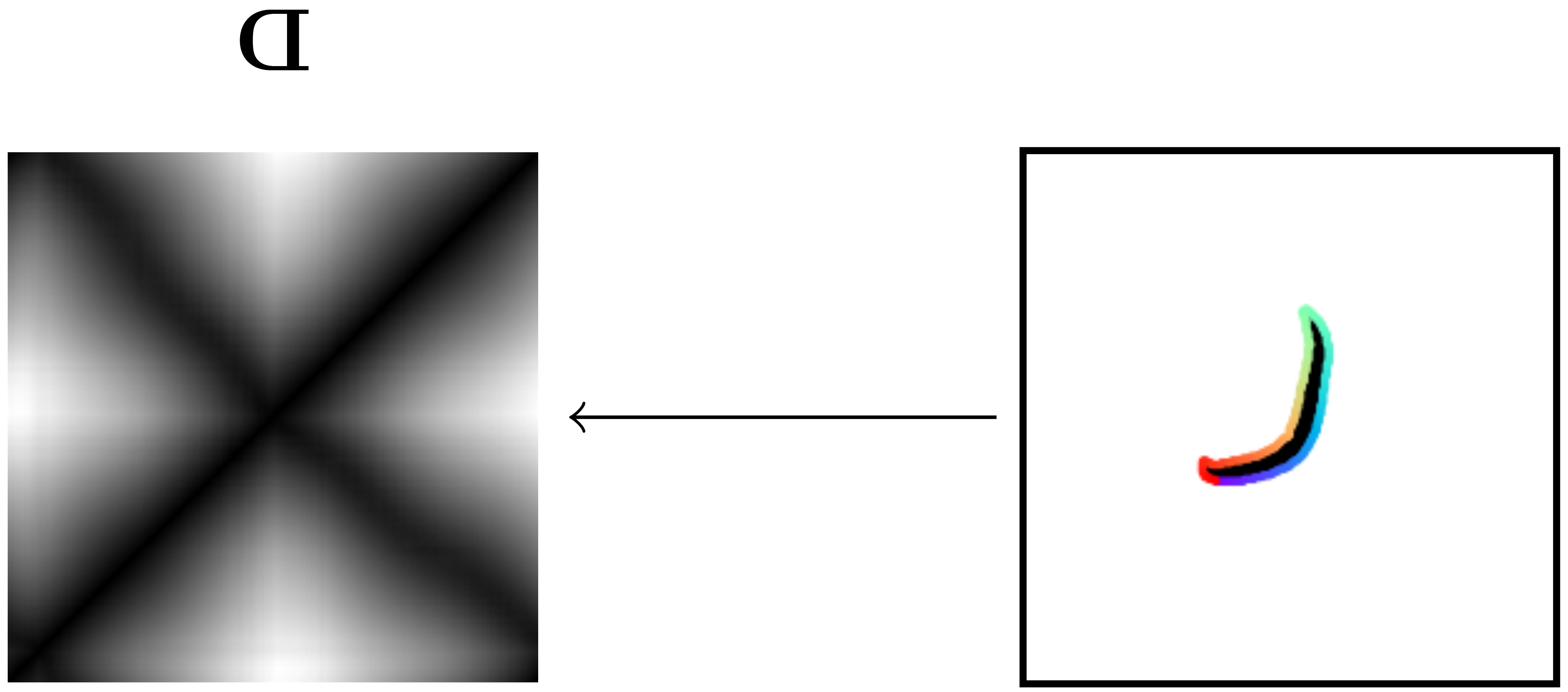
`sklearn.metrics.pairwise.euclidean_distances(X, Y=None, *, Y_norm_squared=None, squared=False, X_norm_squared=None)` [source]

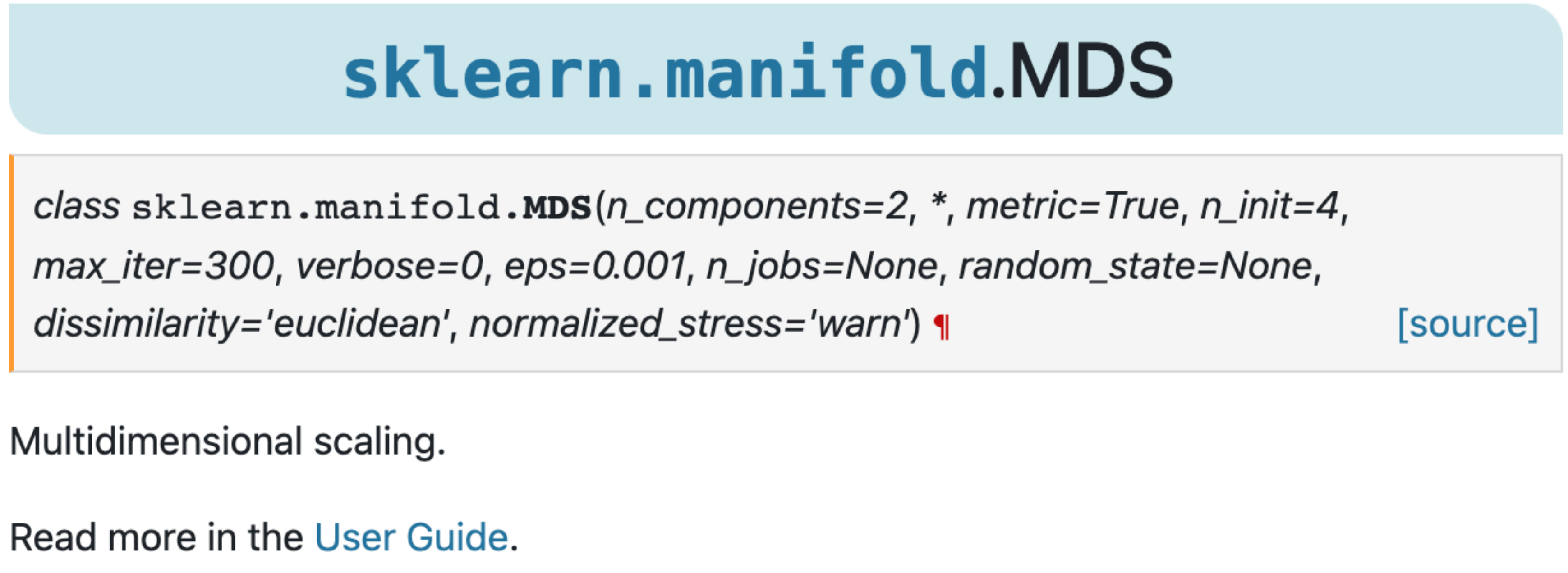
Compute the distance matrix between each pair from a vector array X and Y.


contour
extraction

Distance matrix to contour

- Multi-Dimensional-Scaling
    - Magically iterative algorithm
    - Does not get stuck in local minima
    - Will always converge
- The seed value essentially only randomly controls
    - The rotation
    - The indexation
- The essence of the shape is *always* recovered





sklearn.manifold.MDS

class `sklearn.manifold.MDS`(*n_components=2*, ***, *metric=True*, *n_init=4*, *max_iter=300*, *verbose=0*, *eps=0.001*, *n_jobs=None*, *random_state=None*, *dissimilarity='euclidean'*, *normalized_stress='warn'*) ¶ [source]

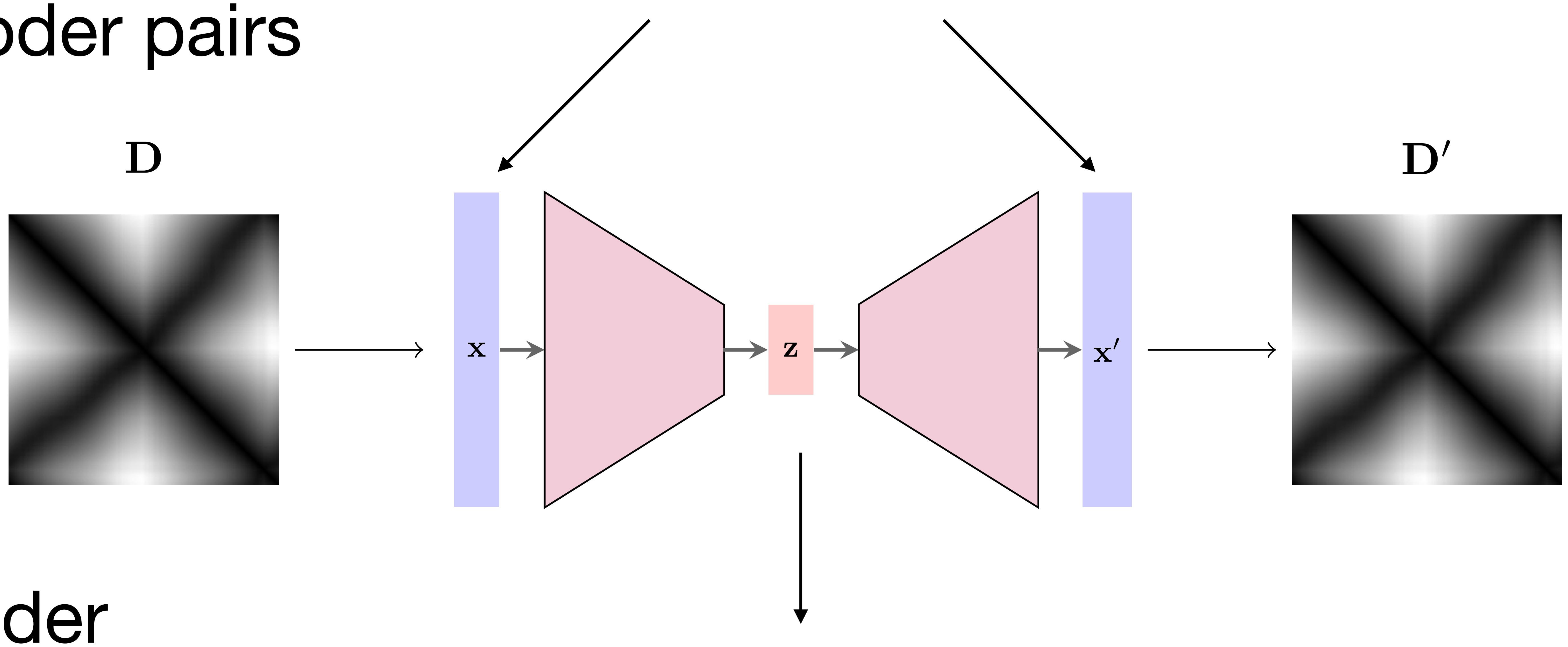
Multidimensional scaling.

Read more in the User Guide.

Autoencoding shapes

- We use matched encoder-decoder pairs
- Currently available
  - ResNet{18,50,110}
- **Future :**
  - Segmentation anything encoder
  - Mask auto encoder
  - (These might be the same or similar)

Batched tensor



Latent representation

Distance matrix losses

1. $\mathscr{L}_{\text{diagonal}}(\mathbf{D}') = \text{MSE}\left(\text{diag}(\mathbf{D}'), \mathbf{0}\right)$
    - All zero leading diagonal -> discourage deviation of diagonals from zero

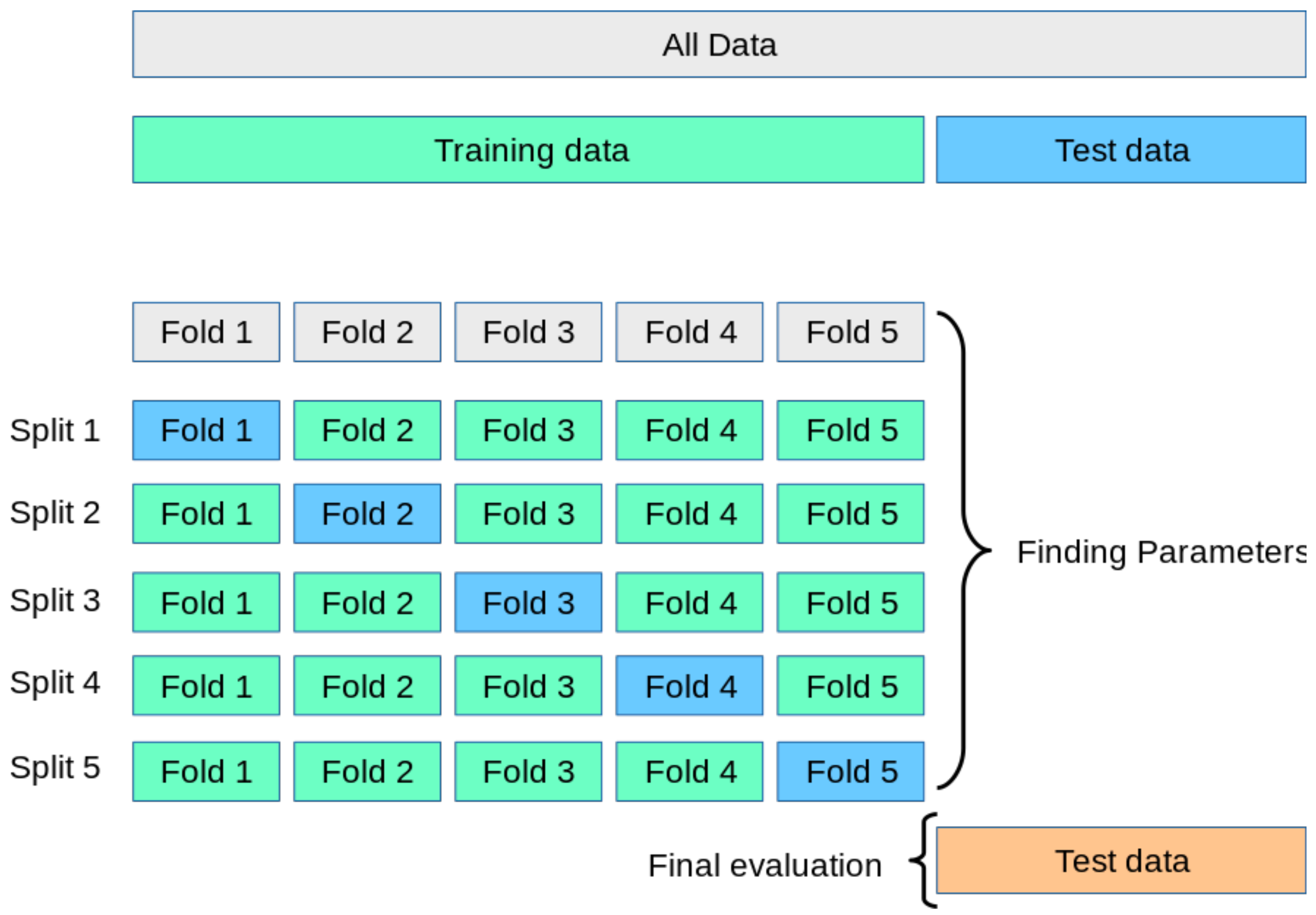
2. $\mathscr{L}_{\text{non-negativity}}(\mathbf{D}, \mathbf{D}') = \text{MSE}\left(\max(0, \mathbf{D}' - \mathbf{D})\right)$
    - All off-diagonal values are positive.

3. $\mathscr{L}_{\text{symmetry}}(\mathbf{D}, \mathbf{D}') = \text{MSE}\left(\mathbf{D}', \mathbf{D}^{\prime T}\right)$
    - Penalize any discrepancy between the matrix and its transpose

4. $\mathscr{L}_{\text{triangle}}(\mathbf{D}') = \text{ReLU}\left(\frac{1}{N}\sum_{i=1}^{N}[d_{ij} + d_{jk} - d_{ik}]_{+}\right)$
    - Euclidean points mean triangle inequality is valid

Results

- Data is k-folds split and stratified
- Worm masks are labelled: $[\mathrm{alive}, \mathrm{dead}]$
  - `BBBC010_v1_foreground_eachworm_live_dead/{label}/{image}.png`



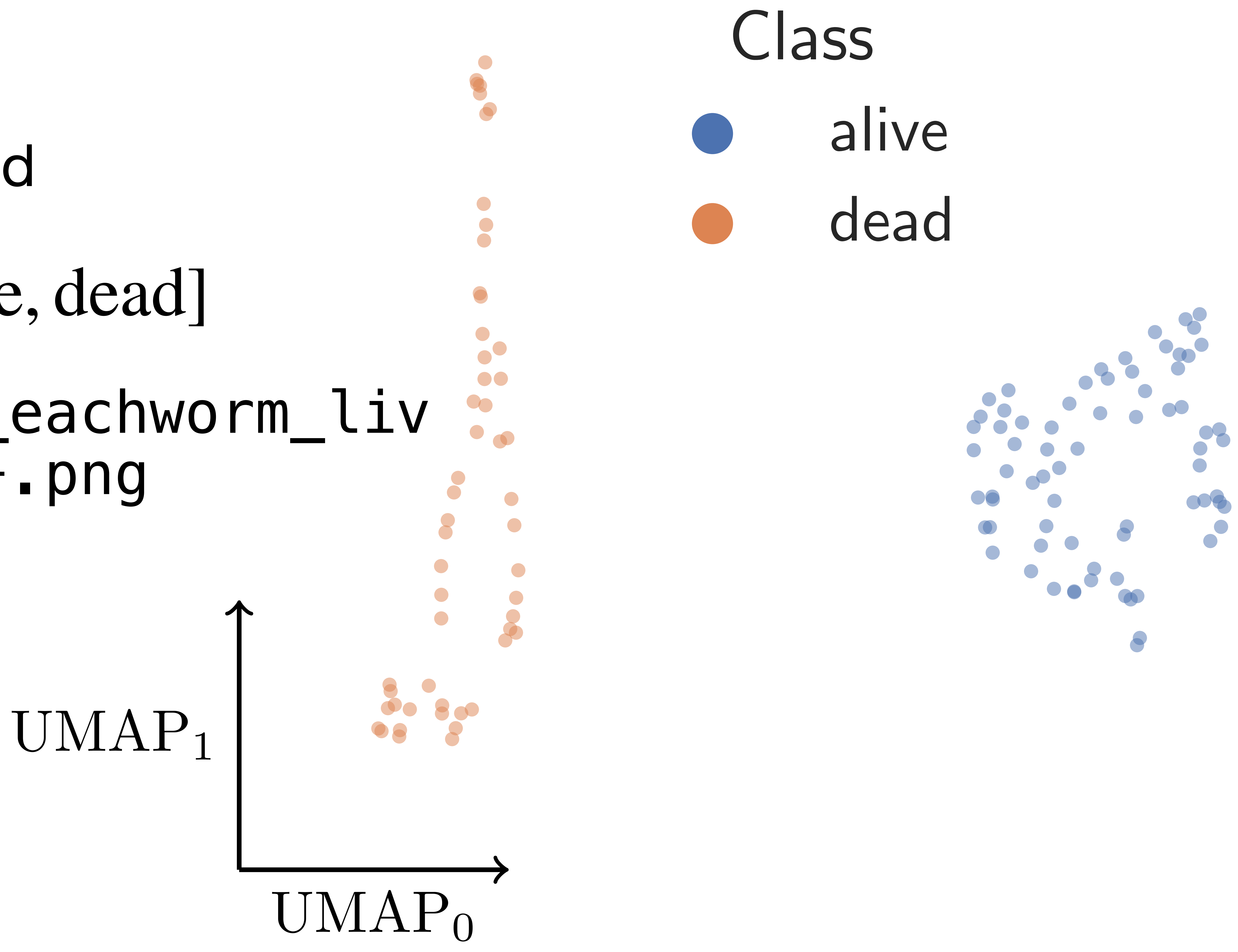
All Data

Training data | Test data

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

Finding Parameters

Final evaluation — Test data



Conclusions

- Alternative **shape representer** that has useful shape priors and **invariances baked-in**
    - Rotation, translation, [opt.] scale
- **Model agnostic** -> No special invariant layers
- Information spread generally better through the image vs black
- **Outperforms simple classical methods** on biological shape data
- **Complimentary to image features** e.g. easy to concatenate $\mathbf{D}$ onto an image
- Tight **control over latent space**, useful for shape generation