INSTITUTE OF COMPUTER SCIENCES
Master in Artificial Intelligence and Data Science

Universitätsstr. 1 D–40225 Düsseldorf

Heinrich Heine Universität Düsseldorf

AI-based fluorescent labeling for cell line development

Hanna Pankova

Master thesis

Date of issue: 01. April 2022
Date of submission: 29. August 2022
Reviewers: Prof. Dr. Markus Kollmann
Dr. Wolfgang Halter

Erklärung

Hiermit versichere ich, dass ich diese Master thesis selbstständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Düsseldorf, den 29. August 2022

Hanna Pankova

Abstract

Cell line development is an expensive and time-consuming process, however that is the most modern approach for producing the proteins needed in pharmaceuticals.

Contents

1 Introduction

1.1 Motivation

1.2 Notation

2 Domain knowledge

2.1 Biology

2.1.1 Cell line development process

General theory behind the cell line development process. Starting from what proteins are. How cells are developed. Difficulties of the cell line development process and timelines.

2.1.2 Project specifications of cell line development for Merck KgaA

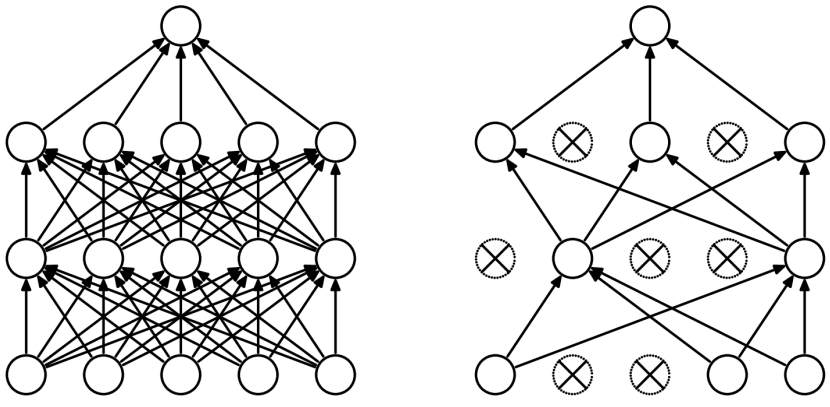
Description of my project, why is it useful, what are the processes here. How my neural network can be used for further stability predictions.

2.2 Deep learning and machine learning basics

Introduction of the notaiton for the dataset, parameters, predictions.

2.2.1 Neural networks

Convolutional neural network, Autoencoder, embedding, optimizers, regularization, descriptions of how each layer works.



(a) Standard Neural Net (b) After applying dropout.

Figure 1: Dropout

2.2.2 Clustering

Theory of clustering algorithms, DBSCAN, HDBSCAN, PCA

2.3 Imaging

2.3.1 Digital imaging

How image is stored in memory, which conventions there are (RGB, BGR (conventions are used in corruptions augmentations)).

2.3.2 Microscopy imaging

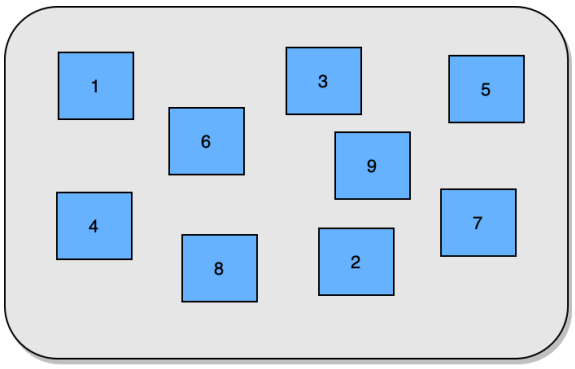


Figure 2: Way in which photos of the well-plate were taken

Which difficulties it may cause (validation loss is lower than train loss)

2.3.2.1 Crops combination technique

Improve this plot by showing the visible border explicitly, example of how it can influence a further segmentation perhaps?

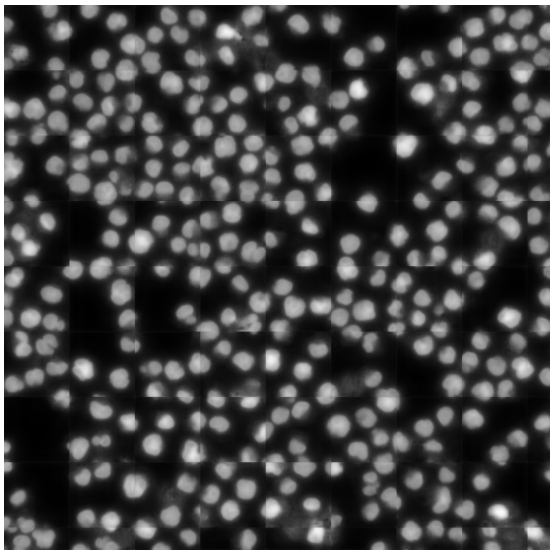


Figure 3: No overlap

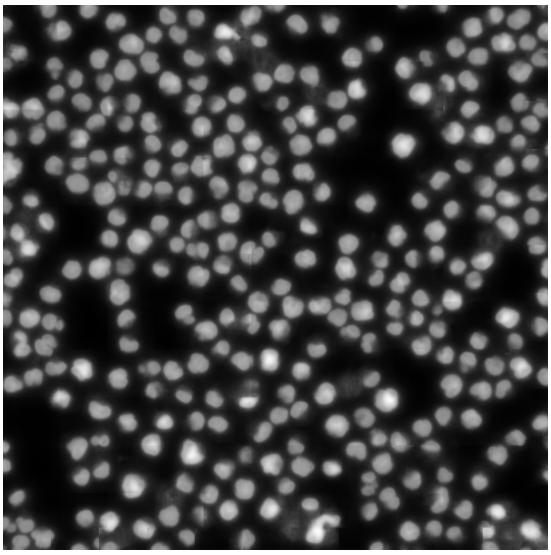


Figure 4: 30 pixels overlap

3 Model training

3.1 Neural network architecture

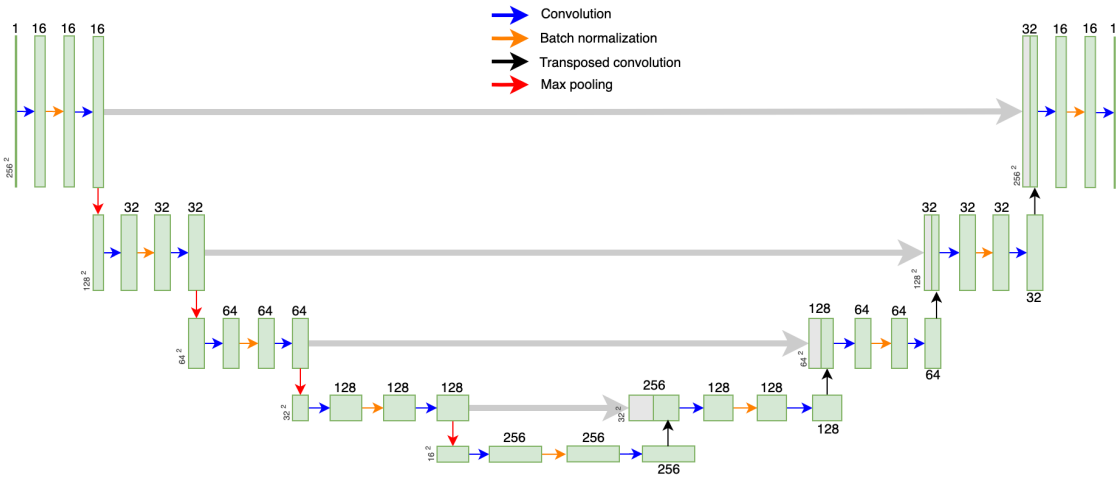


Figure 5: Unet

And information on the embeddings, output sizes, amount of parameters, etc.

3.2 Loss functions

Which loss functions were used, Pearson correlation coefficient explained.

3.3 Available data

Description of the datasets and the amount of images in each category.

Table 1: Available data for each fo the organelles

	Total images	Training crops	Validation crops	Test crops
Nuclei	595	27,264	3,008	7,616
Actin	400	18,432	2,048	5120
Golgi	761	23,036	2,336	6,347
H19	400	27,264	3,008	7,61
Nucleolei	?	?	?	?

3.4 Training costs estimation

Table with the estimation of costs and times for AWS

3.5 Augmentations

Description of all augmentations used

3.5.1 Smart augmentations for rotation and scaling

3.6 Convergence

Images of train and validation loss.

3.7 Model setup

3.7.1 Weight Initialization

These plots represent MSE

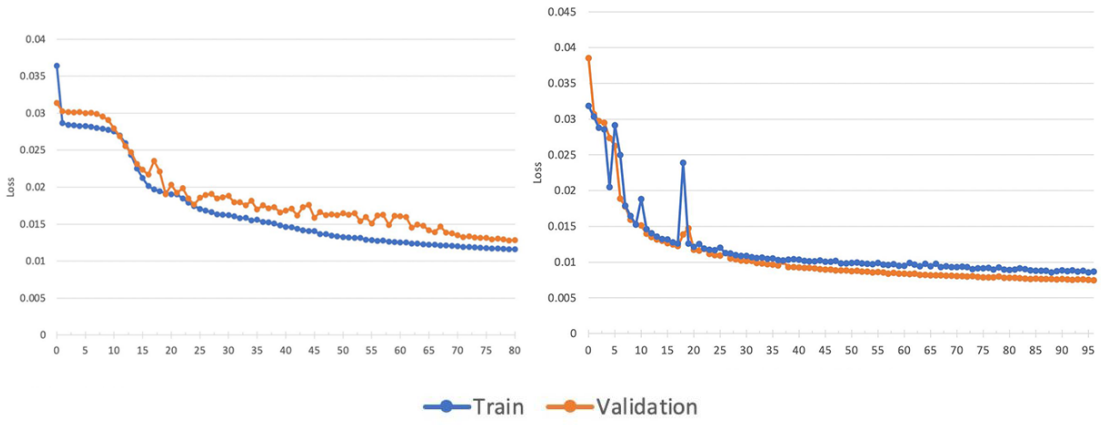


Figure 6: Nuclei training without and with custom weight initialization

3.7.2 Regularization

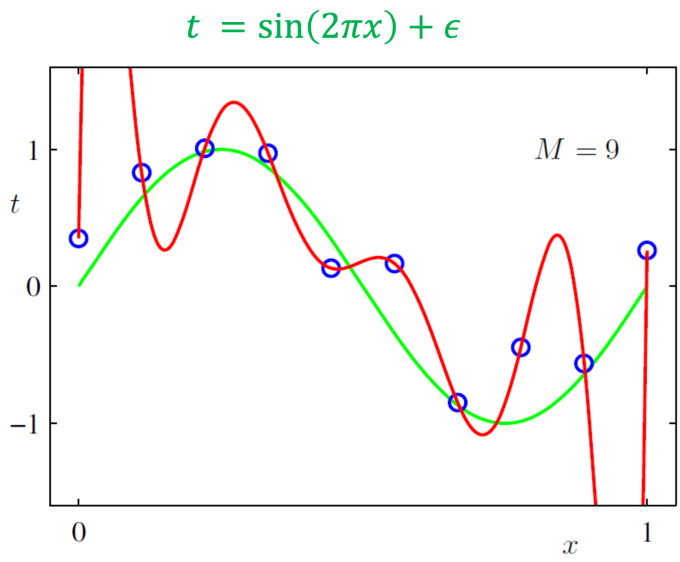


Figure 7: Overfitting

3.7.3 Optimizers

Comparison of different optimizers

4 Nuclei

4.1 Preprocessing

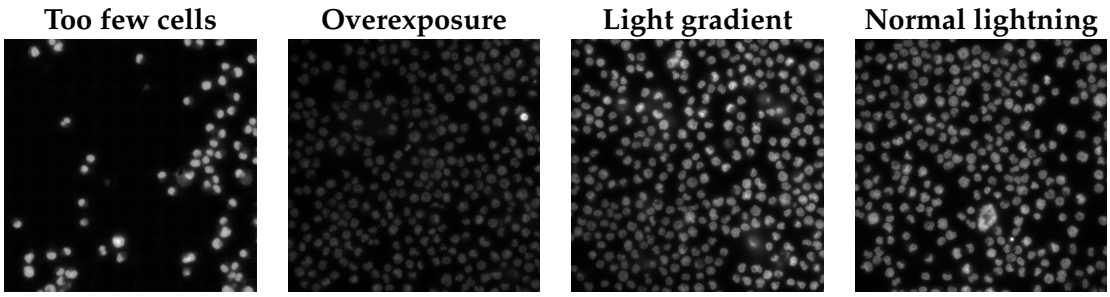


Figure 8: Different lightning conditions

4.1.1 Thresholding algorithms

Global and local thresholding

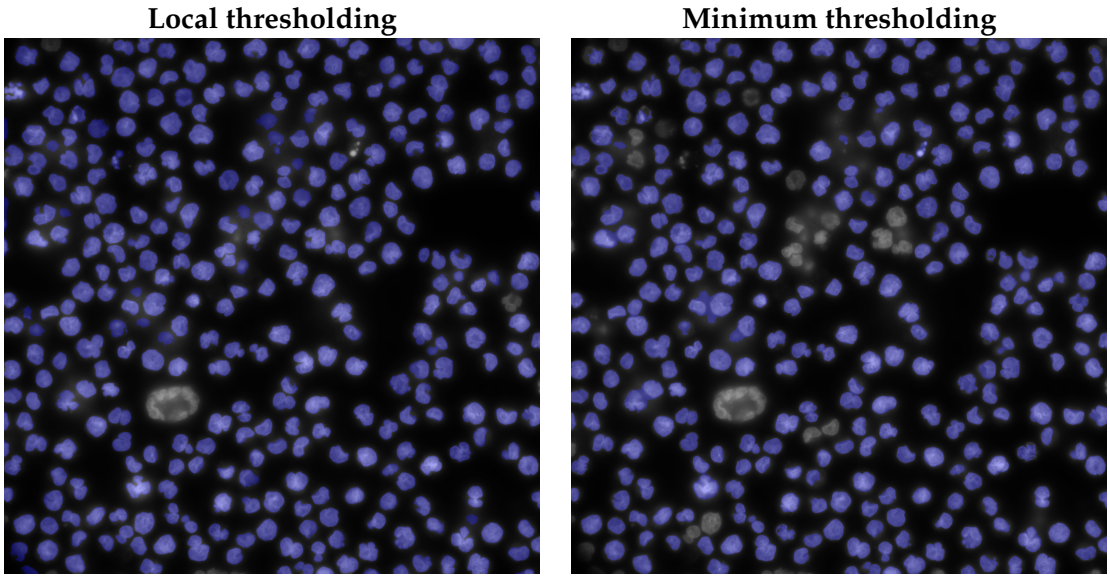
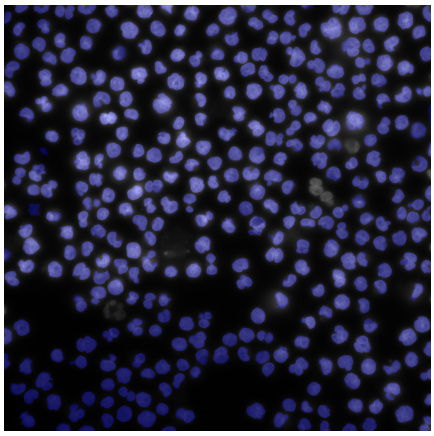
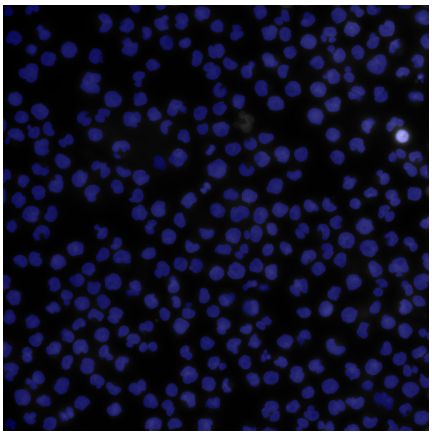


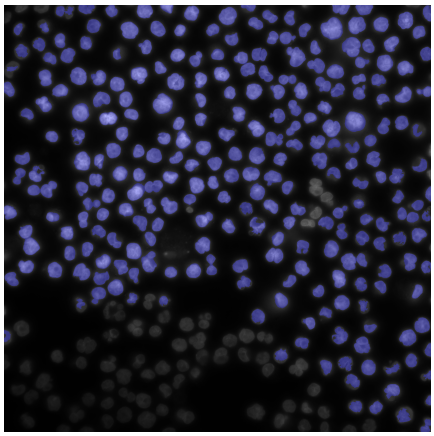
Figure 10: Local vs. Global thresholding (normal conditions)



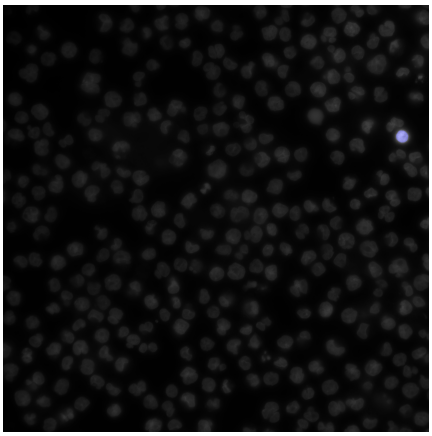
(a) Local



(b) Local



(c) Global, minimum



(d) Global, minimum

Figure 9: Local vs. Global thresholding

4.2 Training and predictions

4.2.1 Convergence

Has the model converged or not. Will more data help?

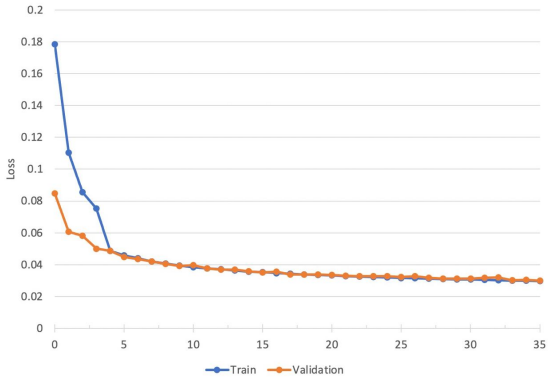


Figure 11: Having more data makes training more stable

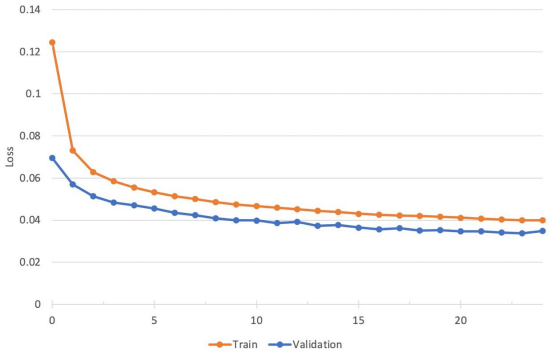


Figure 12: With regularization and augmentations PCC

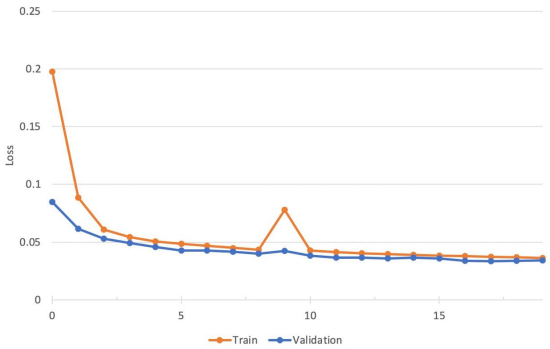


Figure 13: No regularization but augmentations

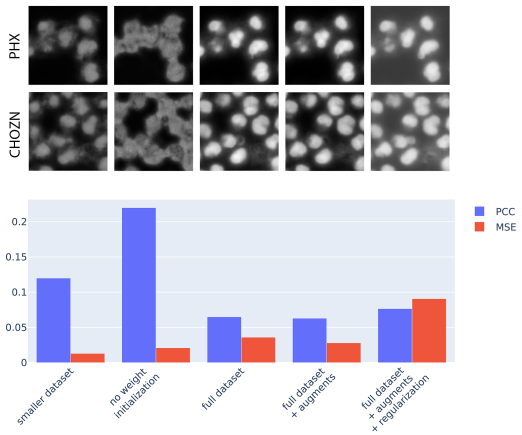


Figure 14: Difefrent models predictions and scores comparison

4.2.2 Predictions quality

Blurry, boundaries, not enough of details and possible improvements

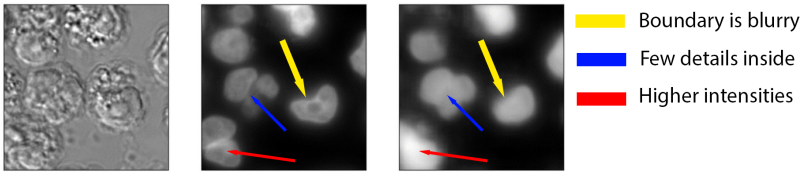


Figure 15: Some troubles in predictions

4.3 Postprocessing for nuclei segmentation

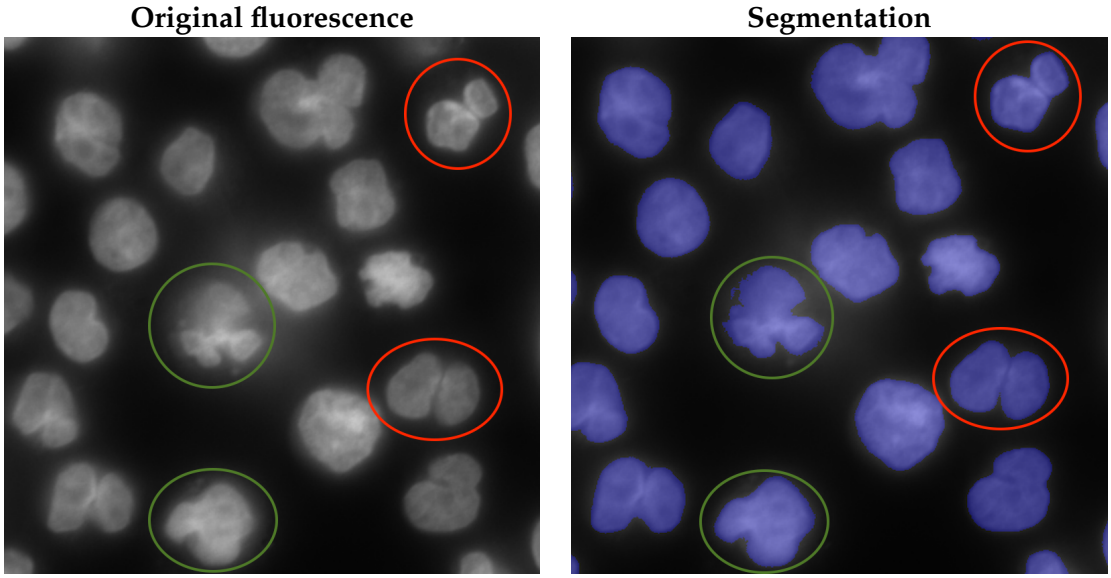


Figure 16: Closely located cells

Overall algorithm

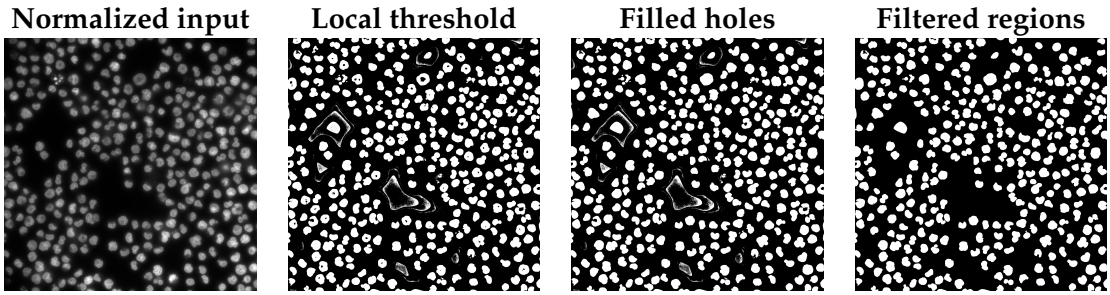


Figure 17: Fluorescence segmentation

4.4 Influence of scaling on predictions quality

Examples of predictions quality with different scales.

Table 2: Paerson correlation coefficients for downstream tasks for different scaling factors

	1.3 scale	0.7 scale	model with augmentations on 1.3 scale
Number of nuclei	0.987	0.995	0.975
Total intensity	0.902	.88	0.86
Mean intensity	0.902	0.906	0.88
Area	0.991	0.992	0.96

5 ER

5.1 Preprocessing

Algorithm 1 Fluorescence segmentation

1. Normalize image
2. Apply global *threshold_mean* to receive initial mask.
3. Zero out pixels outside the mask
4. Apply local thresholding.
5. Apply *fill_holes* transformation.
6. Morphological opening from opencv and Gaussian blur.
7. Run *findContours* from opencv in order to obtain separate regions and filter out too small regions.

Segmentation steps are also illustrated in Figure

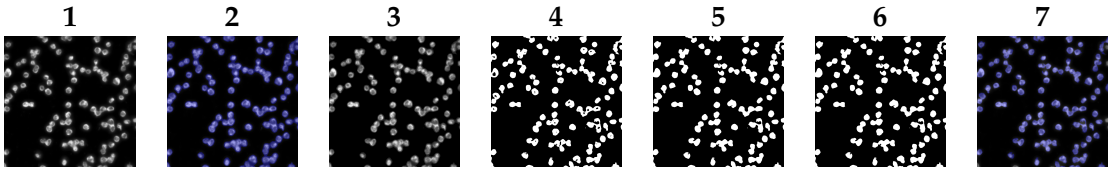


Figure 18: ER prediction

5.2 Training and predictions

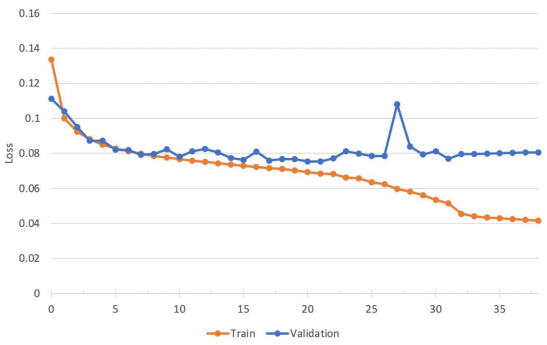


Figure 19: Overfit

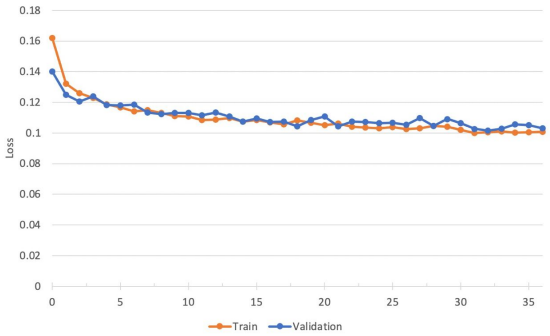


Figure 20: No overfit with augmentations

5.3 Combination of nuclei and actin predictions

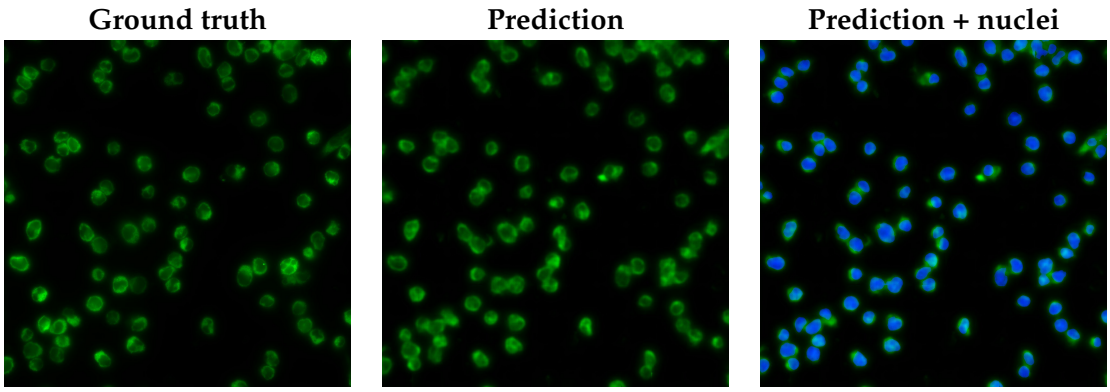


Figure 21: ER prediction

5.4 Generalizability across phenotypes

TODO train the model on one phenotype and predict on the other, compare predictions (visually?) postprocessing with metrics then?

6 Golgi

6.1 Preprocessing

Enhancement

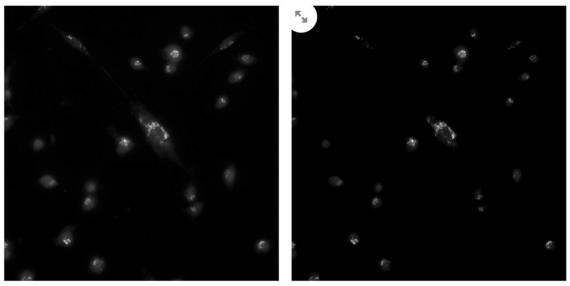


Figure 22: Golgi enhancement

6.1.1 Background removal algorithms

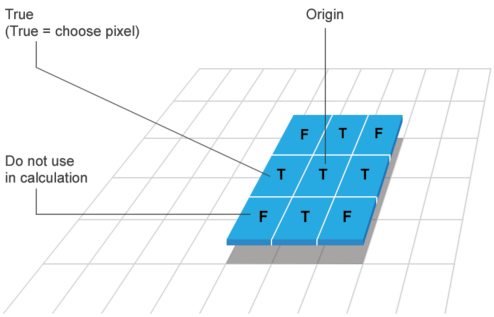


Figure 23: Structuring Element

Rolling ball algorithms

Rolling ball still leaves some noise

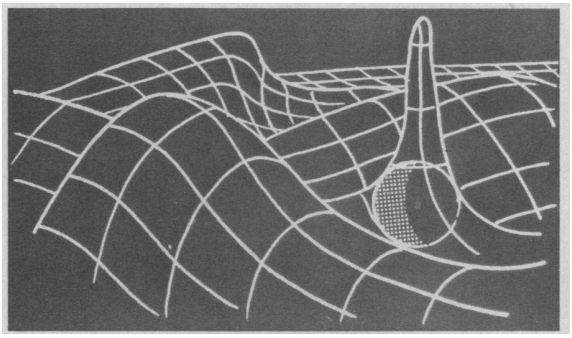
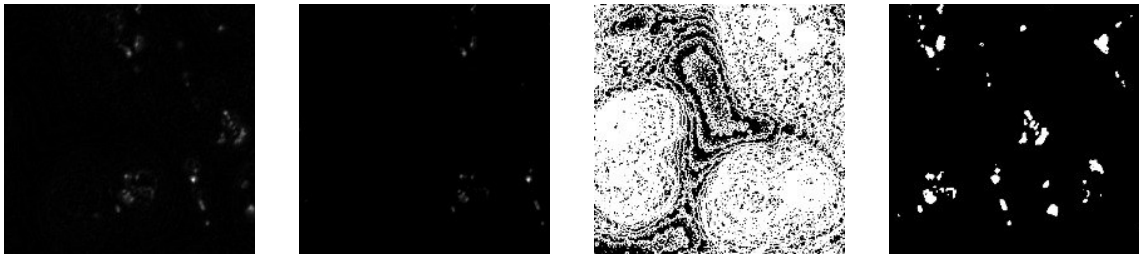


Figure 24: Rolling Ball



(a) (b) (c) (d)

Figure 25: (a) Vanilla pre-processing with automatic background removal algorithm only; (b) Additional clipping of lower intensities after vanilla pre-processing; (c) masked or subfigure (a); (d) mask of subfigure (b)

6.2 Training and predictions

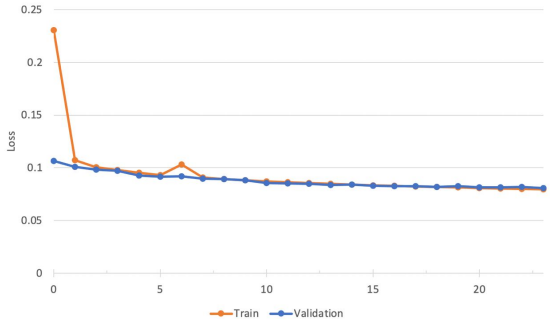


Figure 26: Straightforward training doesn't work

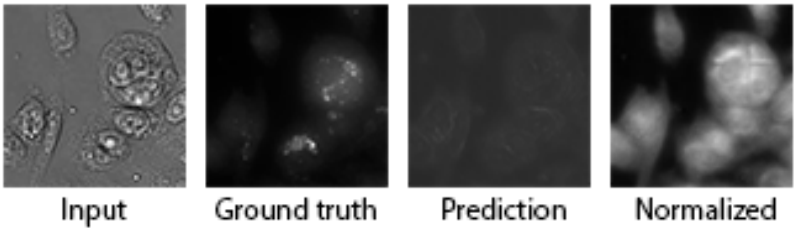


Figure 27: Training on original data

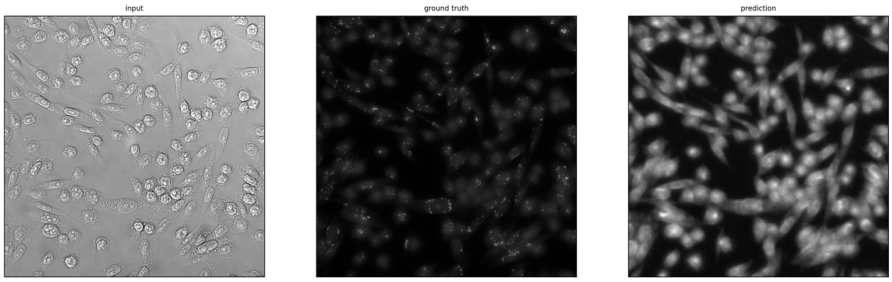


Figure 28: Full size predictions

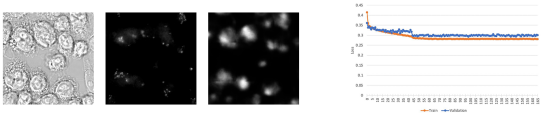


Figure 29: Training on the enhanced data

6.3 Alternative ways to improve predictions

6.3.1 Asymmetrical losses

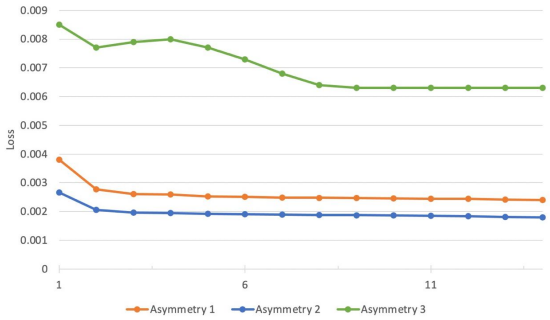


Figure 30: Asymmetrical training

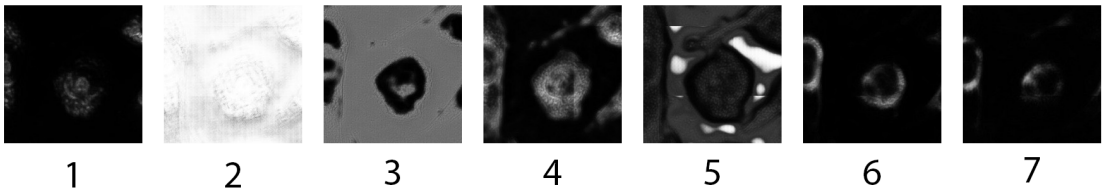


Figure 31: Asymmetrical training predictions

6.3.2 Use of gradient in loss

6.3.3 Noise reduction methods

7 GFP prediction

7.1 Preprocessing

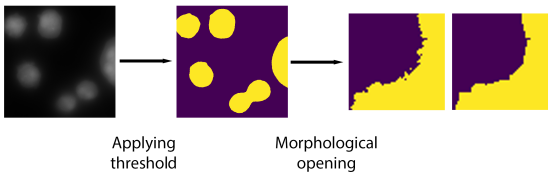


Figure 32: Converting GFP to a binary mask

7.2 Predictions

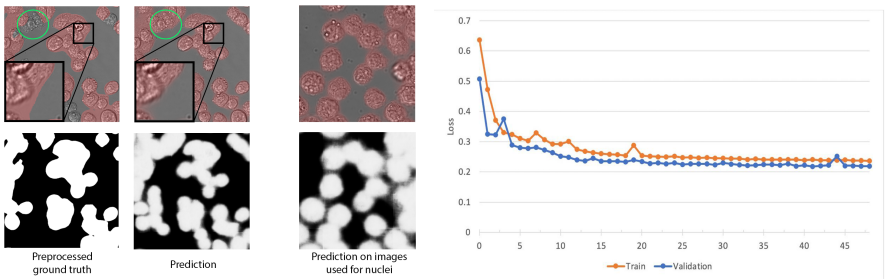


Figure 33: Training with BCE loss

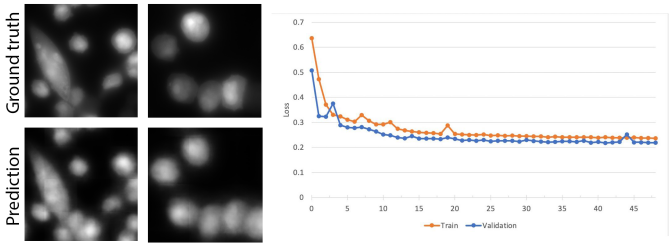


Figure 34: Training with Pearson correlation loss

Table 3: Correlation coefficients for downstream tasks

Binary training	Pearson	Spearman
Number of ER	0.67	0.64
Area	0.82	0.75
Continuos training	Pearson	Spearman
Number of ER	0.57	0.55
Area	0.26	0.64

7.3 Downstream metrics

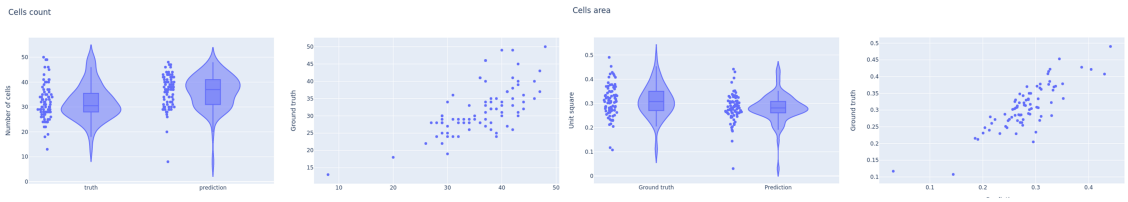


Figure 35: Downstream metrics

7.4 Combination of GFP, nuclei and ER

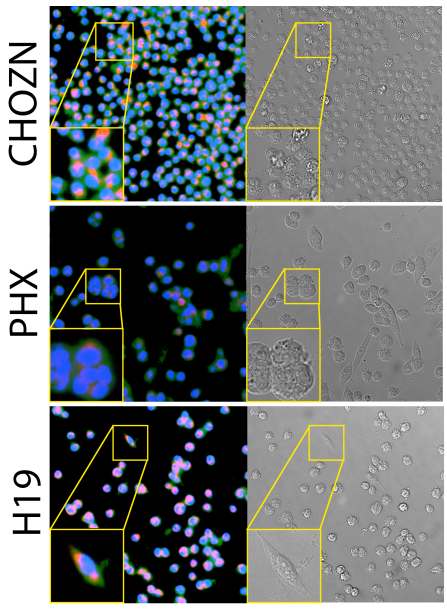


Figure 36: GFP, Nuclei and ER combined

8 Model Evaluation

8.1 Metrics for downstream tasks

Figure 37: Metrics for downstream tasks on nuclei

Table 4: Correlation coefficients for downstream tasks

	Pearson	Spearman
Number of nuclei	0.995	0.994
Total intensity	0.902	.911
Mean intensity	0.907	0.904
Area	0.992	0.990

8.2 Influence of different loss functions on metrics for downstream tasks

9 Stability study

9.1 Artificial corruptions

Description of artificial corruptions.

Table 5: Hyperparameterization for different artificial corruption severities

Corruption \ Severity	-5	-4	-3	-2	-1	0	1	2	3	4	5
Defocus blur (radius)	-	-	-	-	-	0	0.5	1.0	1.5	2	3
Contrast (gain)	3.5	3.0	2.5	2.0	1.5	1	0.9	0.8	0.7	0.5	0.3
Brightness (bias)	-150	-135	-120	-90	-50	0	50	90	120	135	150

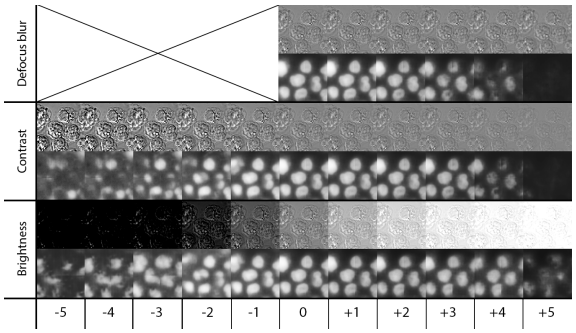


Figure 38: Influence of artificial corruptions on the predictions

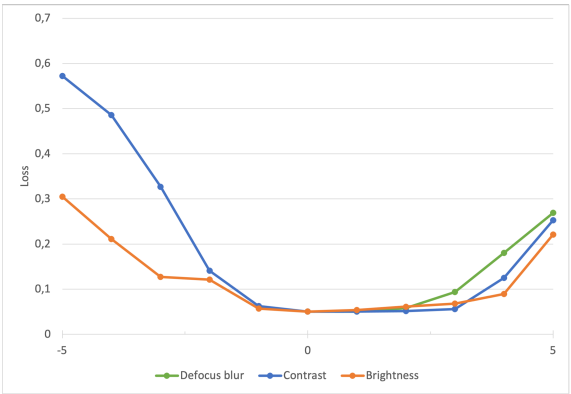


Figure 39: Change of Pearson correlation loss for artificial corruptions

9.2 Real corruptions

9.2.1 Not fixed cells imaging as corrupted input

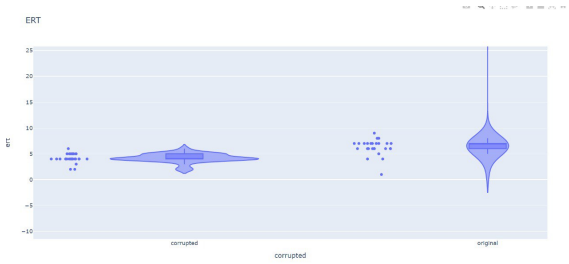


Figure 40: Online drift detection of not fixated cells

Scores of 0.91 however the threshold is 6, not corrupted data (fixed cells) mostly ert of 7 whereas corrupted data (not fixed cells) have an ert of 4. The threshold is therefore 6.

9.2.2 Real-world examples of corruptions

9.3 Influence of corruptions on metrics for downstream tasks

Calculate how metrics worsen when the evaluation stays the same, but the input is corrupted.

9.4 Improving predictions with additional corruption augmentations

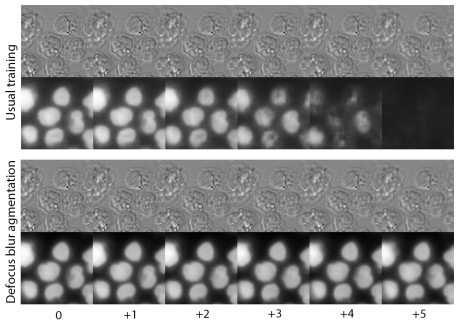


Figure 41: Using corruptions as augmentations improves predictions

10 UNET embeddings study

10.1 Dimensionality reduction and clustering methods

10.1.1 UMAP, t-SNE, PCA, PacMAP

10.1.2 Clustering methods (HDBSCAN, DBSCAN, K-means)

10.2 Application of various dimentionality reduction methods

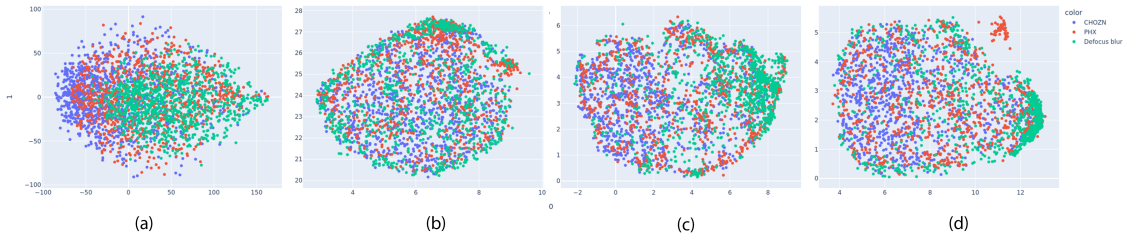


Figure 42: (a) PCA, (b) UMAP, (c) combination of PCA and UMAP with 10 and (d) 50 components

10.3 Autoencoder embeddings as an alternative

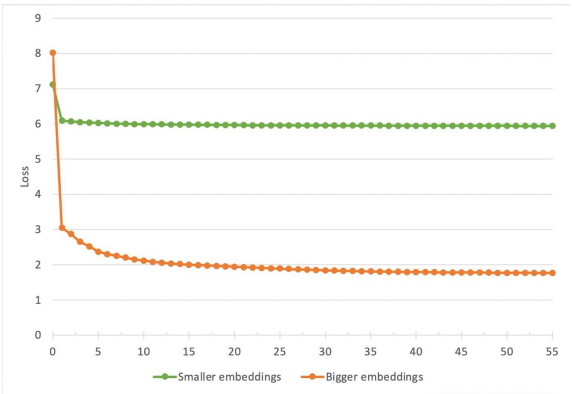


Figure 43: Autoencoders training convergence

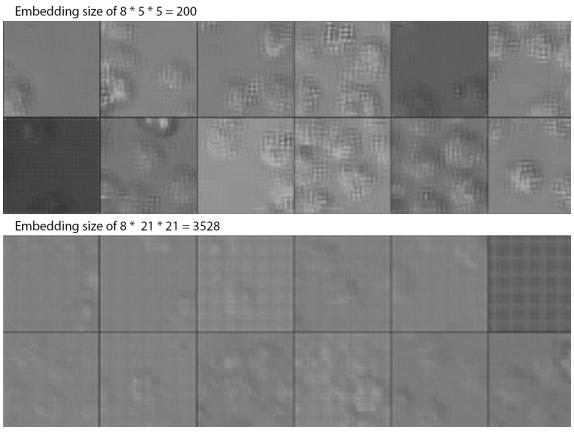


Figure 44: Samples drawn from the trained autoencoder

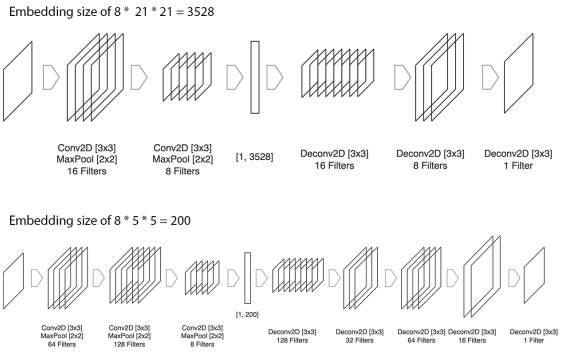


Figure 45: Architectures of two autoencoders

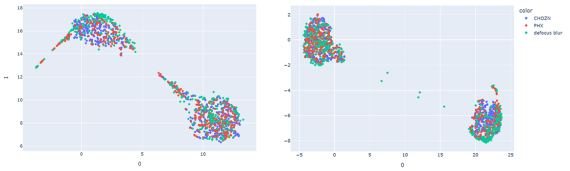


Figure 46: Autoencoder embeddings after applying PCA with 10 components and UMAP afterwards. Earlier epoch VS later epoch

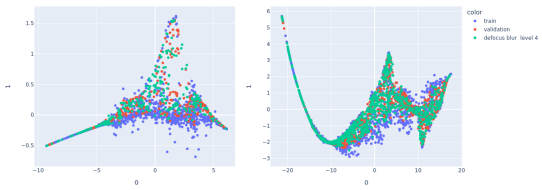


Figure 47: PacMAP does not provide information on the coruption

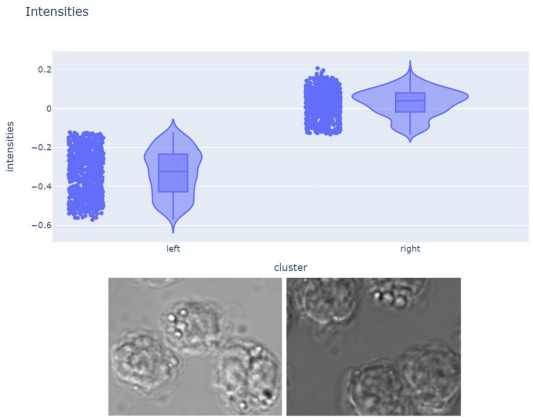


Figure 48: What do two UMAP clusters represent

10.4 Clustering of PacMAP embeddings

10.4.1 Clustering on UNet embeddings

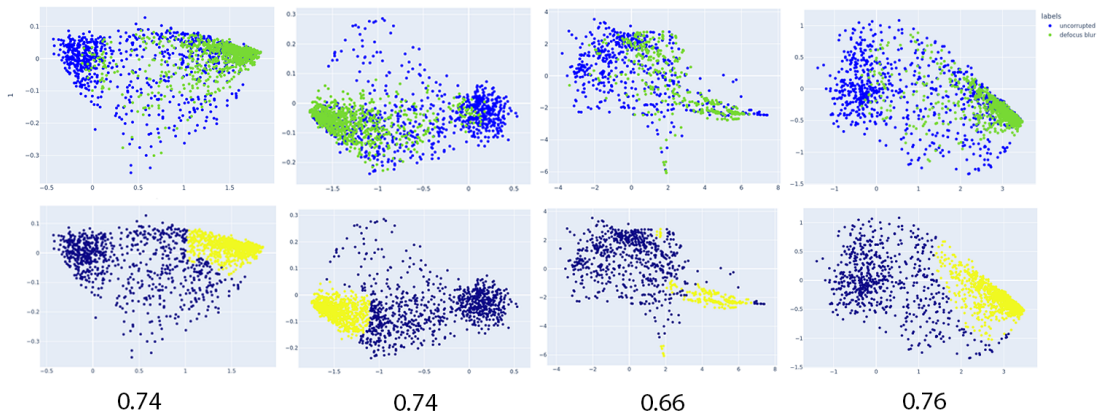


Figure 49: Clustering of UNet embeddings after PacMAP

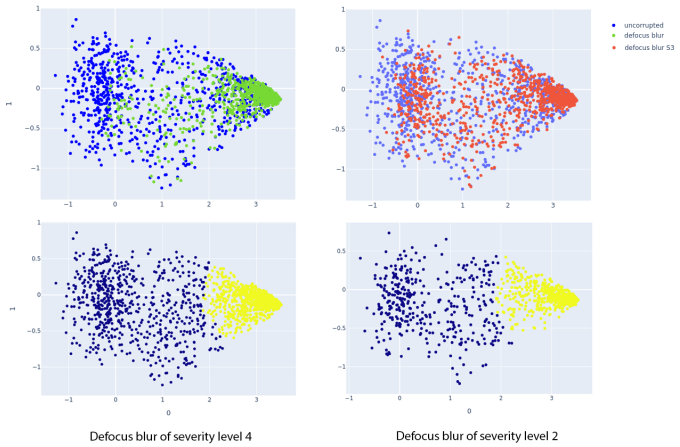


Figure 50: Clustering of UNet embeddings after PacMAP for different severities levels

TABLE with F1-score: 0.76 VS 0.64

11 Drift detection

11.1 A need to detect drift

11.2 Maximum mean discrepancy for drift detection

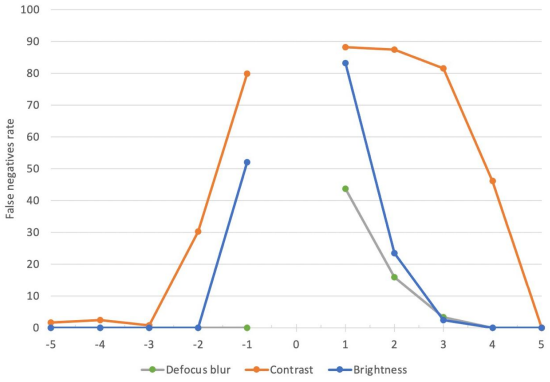


Figure 51: False negatives rate for drift detection on artificial corruptions

11.3 Online version of MMD algorithm

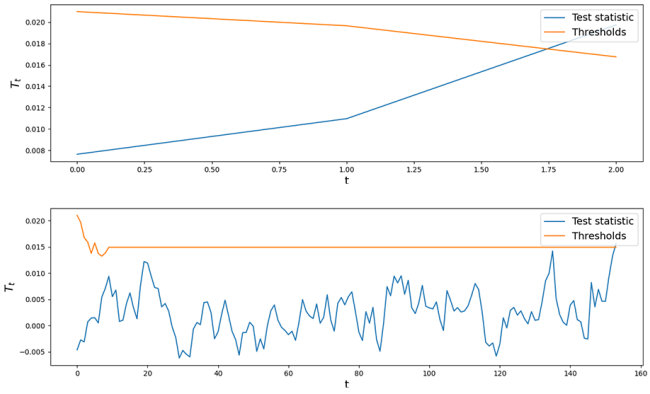


Figure 52: Expected runtime (ERT) for corrupted and in-distribution data

Table 6: Test window size influence on separability

W	2	5	10	15	20
Auc-Roc	0.85	0.92	0.98	0.90	0.88

Table 7: ERT influence on separability

W	32	64	128	256
Auc-Roc	0.90	0.95	0.98	0.98

Table 8: Severity of corruptions on separability

W	Level 2	Level 3	Level 4
Auc-Roc	0.84	0.92	0.98

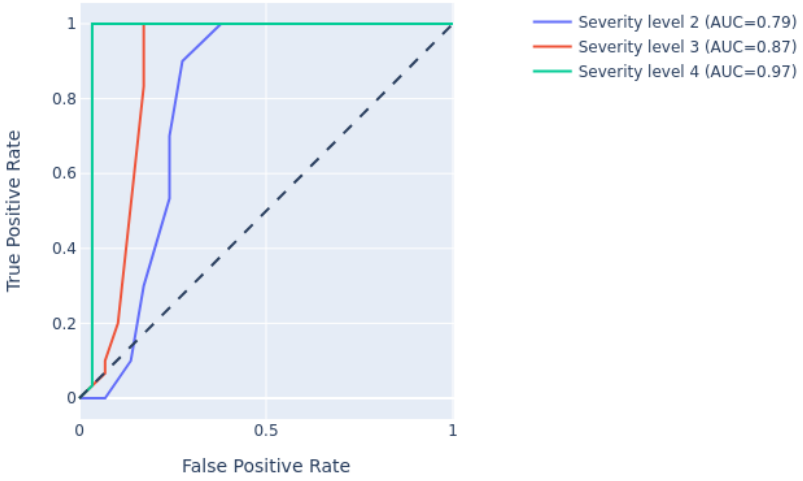


Figure 53: AUC ROC scores for various defocus corruptions severities

12 Software Tools

12.1 Foundry. Palantir

12.2 AWS

12.3 Streamlit

Further research

13 Summary

List of Figures

List of Tables