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

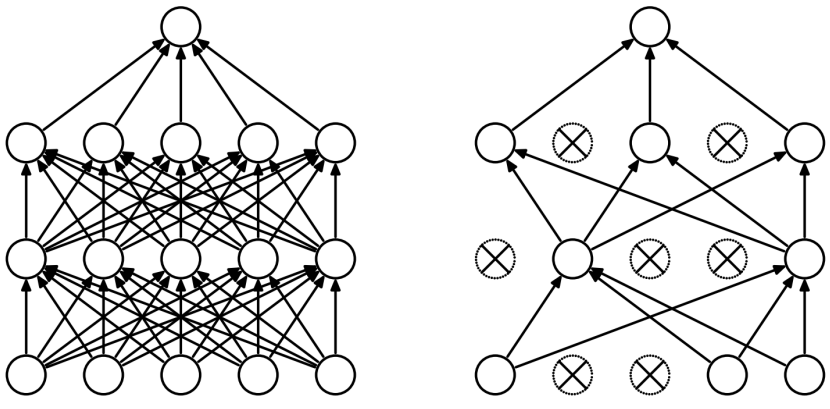
Description of my project, why is it useful, what are the processes here. How my neural network can be used for further stability predictions. My work as a part of the whole process of stability prediction

## 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 Dimensionality reduction methods

##### 2.2.2.1 UMAP

##### 2.2.2.2 PCA

##### 2.2.2.3 PacMAP

#### 2.2.3 Clustering methods

##### 2.2.3.1 DBSCAN

##### 2.2.3.2 HDBSCAN

##### 2.2.3.3 K-means

### 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

##### 2.3.2.1 Image acquisition peculiarity

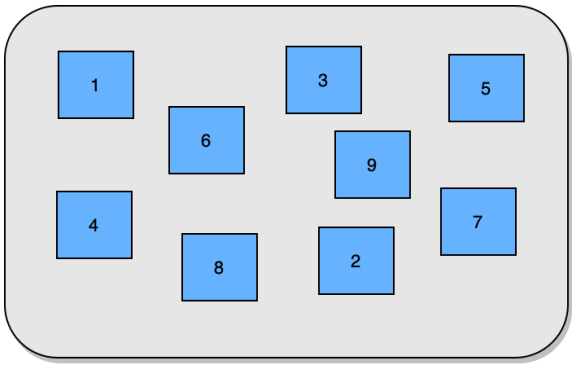



Figure 2: Way in which photos of the well-plate were taken

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

#### 2.3.2.2 Crops combination technique

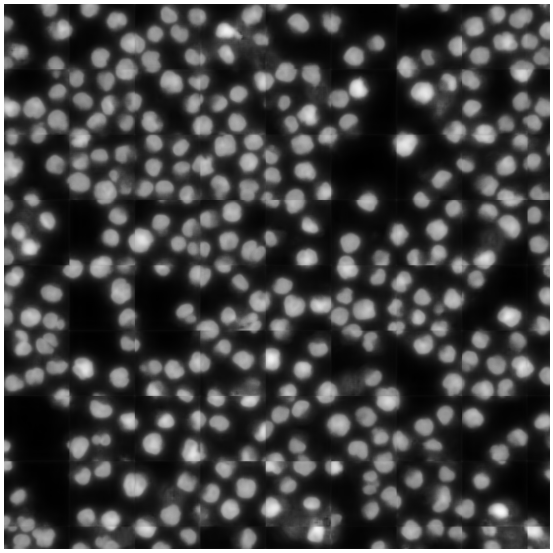


Figure 3: No overlap

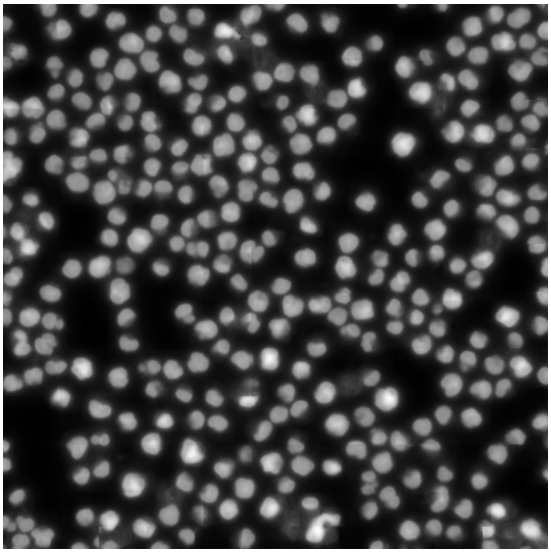


Figure 4: 30 pixels overlap

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

# 3 Implementation and experiments

## 3.1 Model training

### 3.1.1 Neural network architecture

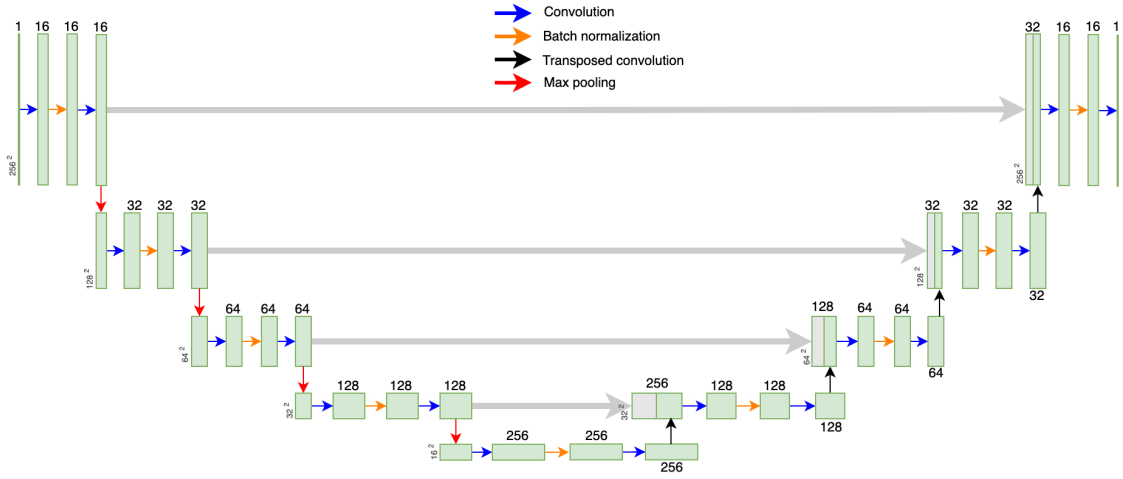



Figure 5: Unet

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

### 3.1.2 Loss functions

Which loss functions were used, Pearson correlation coefficient explained.

### 3.1.3 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 | ? | ? | ? | ? |

### 3.1.4 Training costs estimation

Table with the estimation of costs and times for AWS

### 3.1.5 Augmentations

Description of all augmentations used

#### 3.1.5.1 Special augmentations for rotation and scaling

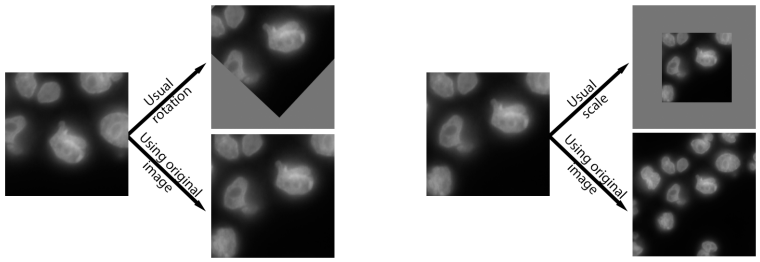



Figure 6: Using original image for rotation and scaling augmentations

### 3.1.6 Model setup

#### 3.1.6.1 Weight Initialization

These plots represent MSE

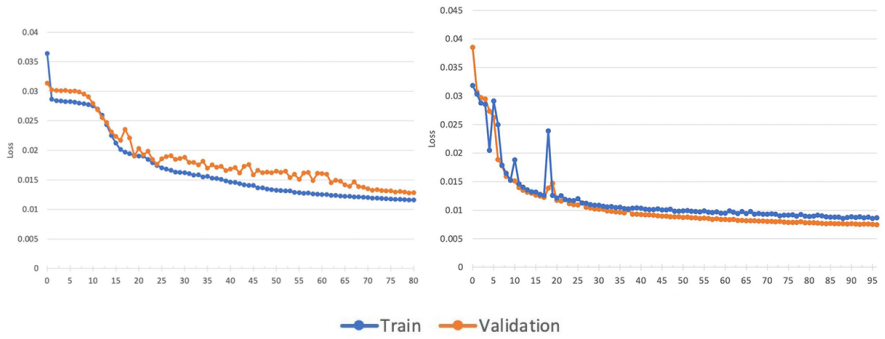



Figure 7: Nuclei training without and with custom weight initialization

#### 3.1.6.2 Regularization

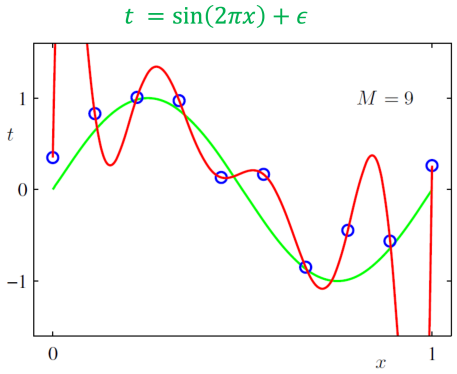



Figure 8: Overfitting

#### 3.1.6.3 Optimizers

Comparison of different optimizers

## 3.2 Nuclei

### 3.2.1 Preprocessing

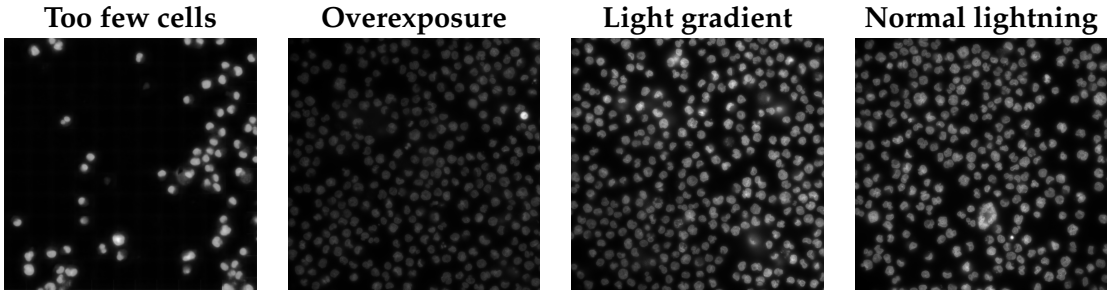



Figure 9: Different lightning conditions

#### 3.2.1.1 Thresholding algorithms

Global and local thresholding

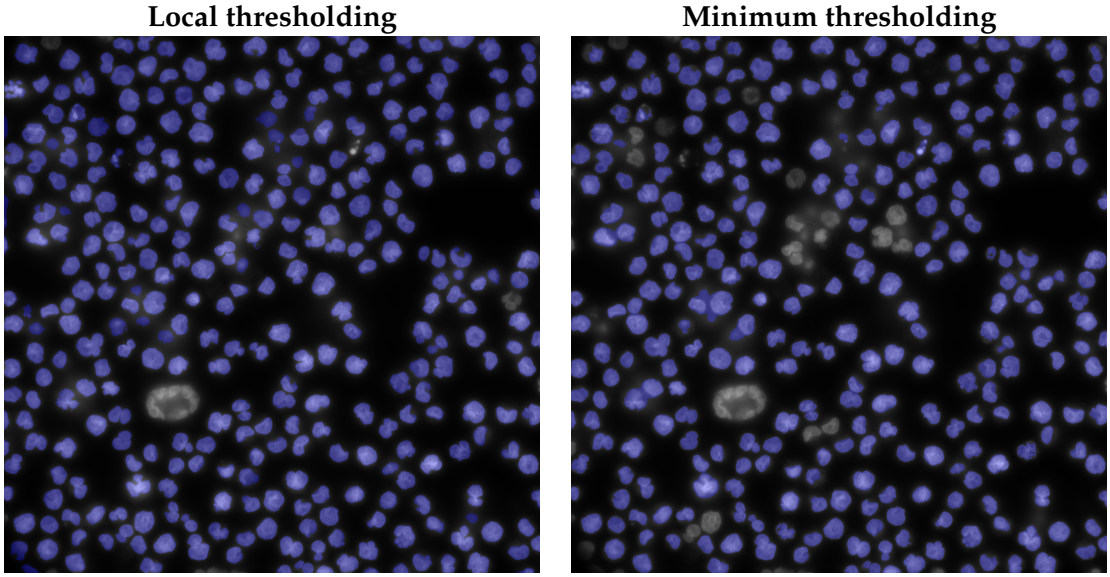
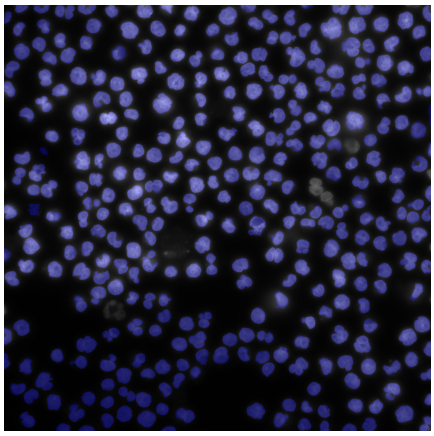



Figure 11: Local vs. Global thresholding (normal conditions)

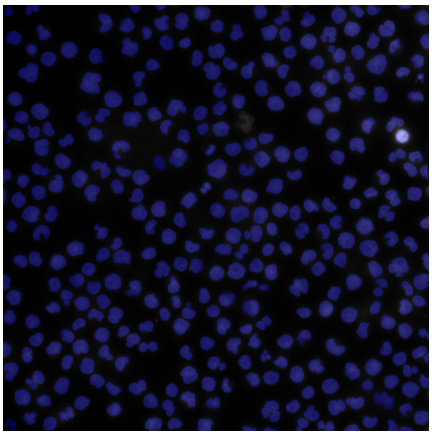
### 3.2.2 Training and predictions

#### 3.2.2.1 Convergence

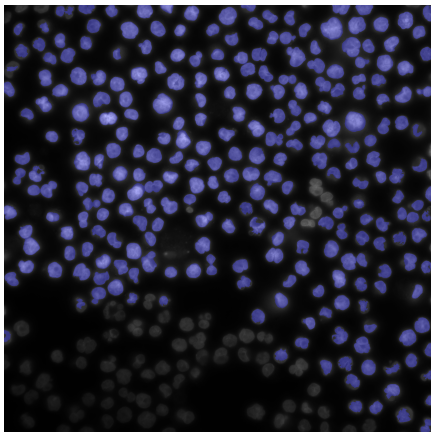
Has the model converged or not. Will more data help?



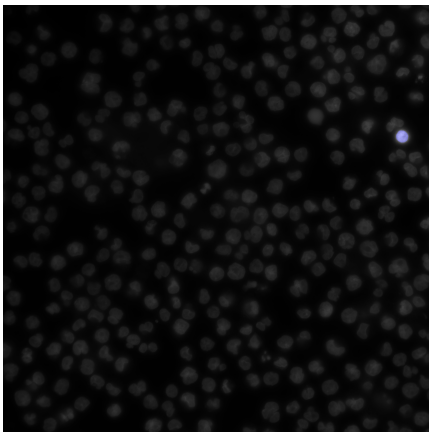
(a) Local



(b) Local



(c) Global, minimum



(d) Global, minimum

Figure 10: Local vs. Global thresholding

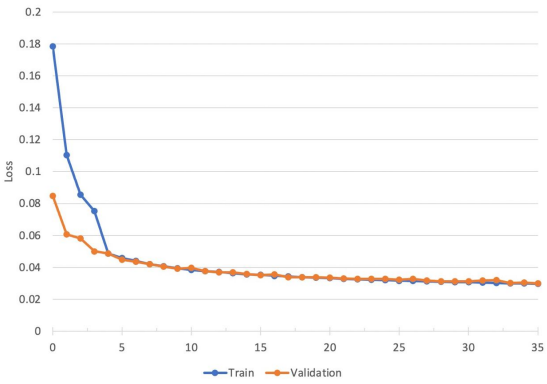



Figure 12: Having more data makes training more stable

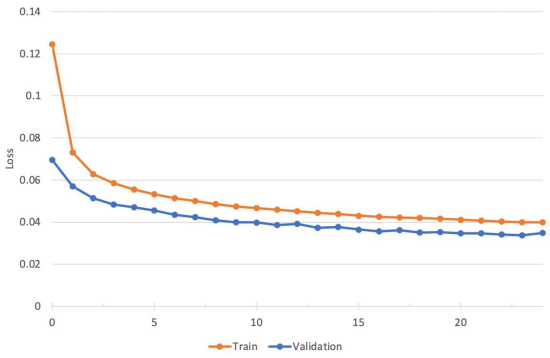



Figure 13: With regularization and augmentations PCC

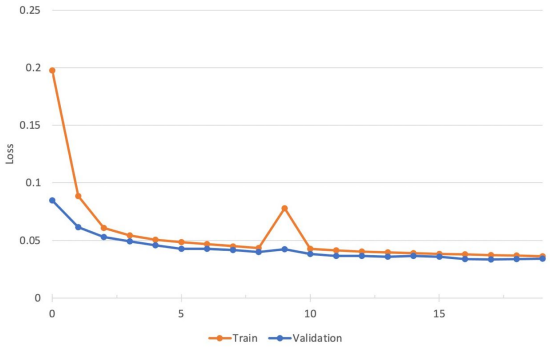



Figure 14: No regularization but augmentations

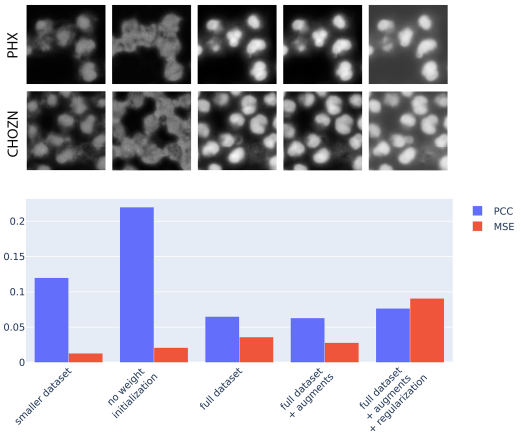



Figure 15: Difefrent models predictions and scores comparison

#### 3.2.2.2 Predictions quality

Blurry, boundaries, not enough of details and possible improvements

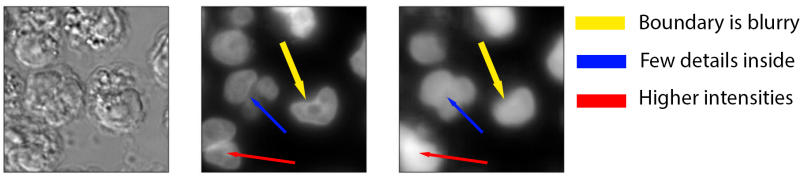



Figure 16: Some troubles in predictions

### 3.2.3 Postprocessing for nuclei segmentation

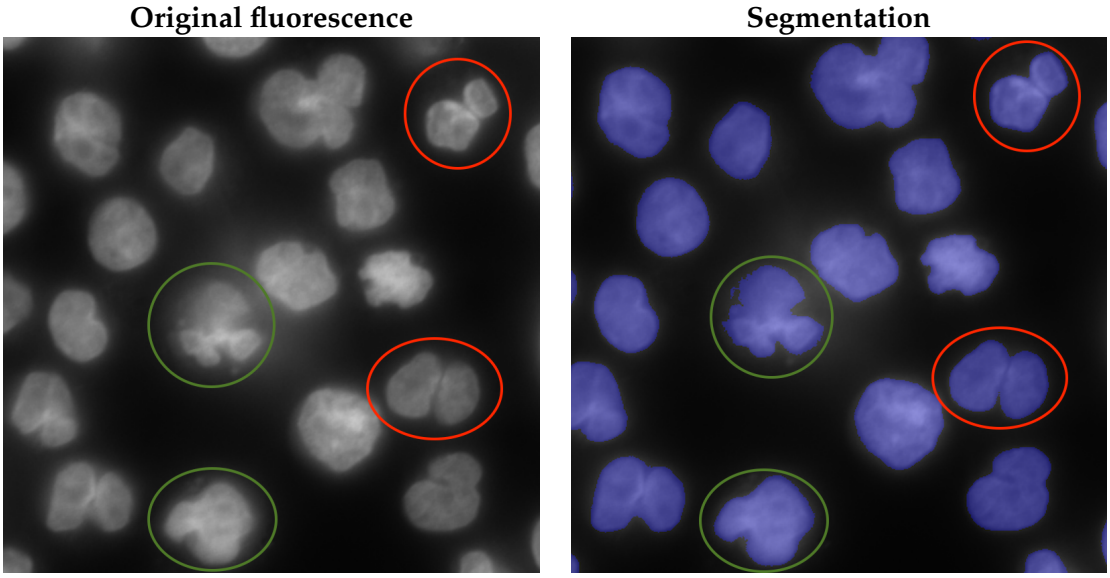



Figure 17: Closely located cells

Overall algorithm

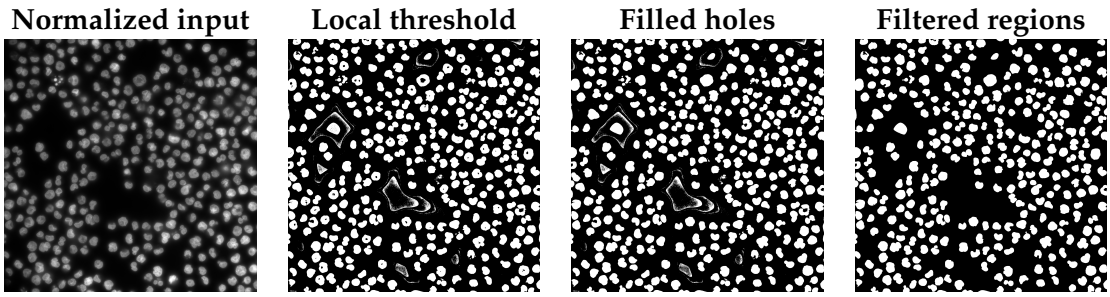



Figure 18: Fluorescence segmentation

### 3.2.4 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 | Train (1.0 scale + augments) Predict (1.3 scale) | Train (1.3 scale) Predict (1.0 scale) | Train (1.3 scale) Predict (0.7 scale) |
|---|---|---|---|---|---|
| Number of nuclei | 0.987 | 0.995 | 0.975 | 0.971 | ? |
| Total intensity | 0.902 | .88 | 0.861 | 0.856 | ? |
| Mean intensity | 0.922 | 0.906 | 0.88 | 0.872 | ? |
| Area | 0.991 | 0.992 | 0.961 | 0.952 | ? |

## 3.3 ER

### 3.3.1 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

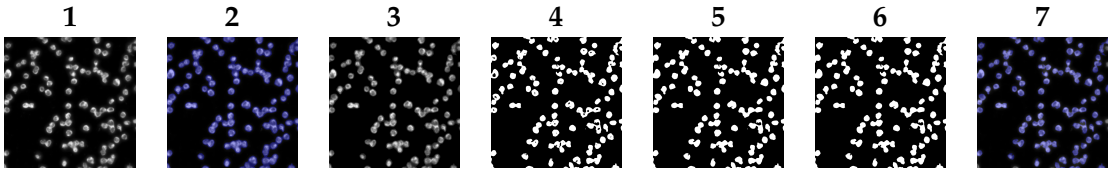



Figure 19: ER prediction

### 3.3.2 Training and predictions

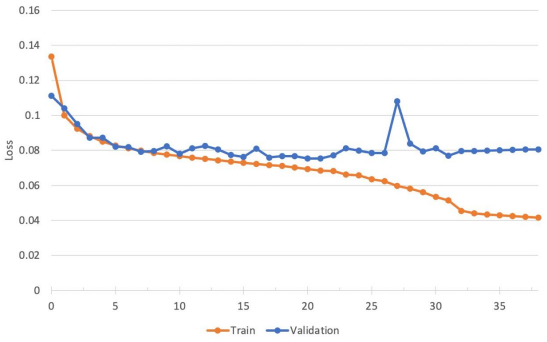



Figure 20: Overfit

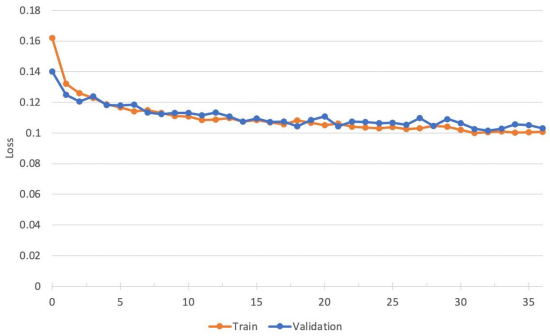



Figure 21: No overfit with augmentations

### 3.3.3 Combination of nuclei and actin predictions

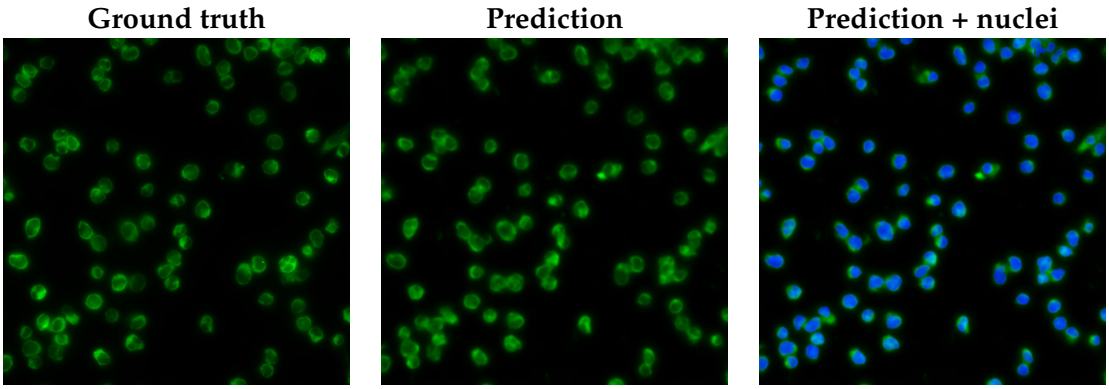



Figure 22: ER prediction

### 3.3.4 Generalizability across phenotypes

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

## 3.4 Golgi

### 3.4.1 Preprocessing

Enhancement

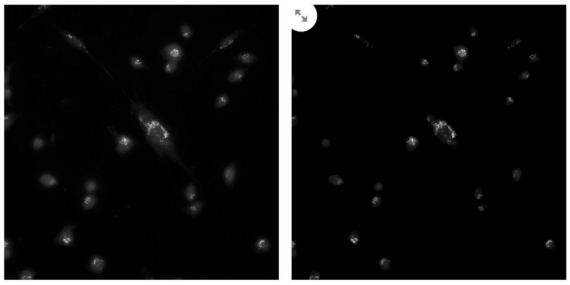


Figure 23: Golgi enhancement

#### 3.4.1.1 Background removal algorithms

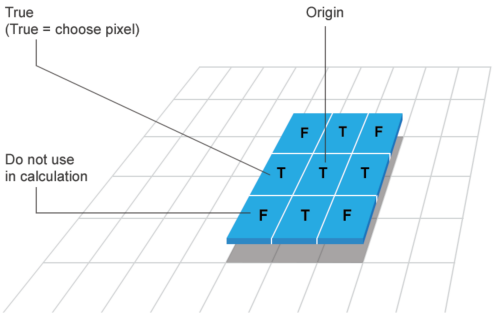



Figure 24: Structuring Element

Rolling ball algorithms

Rolling ball still leaves some noise

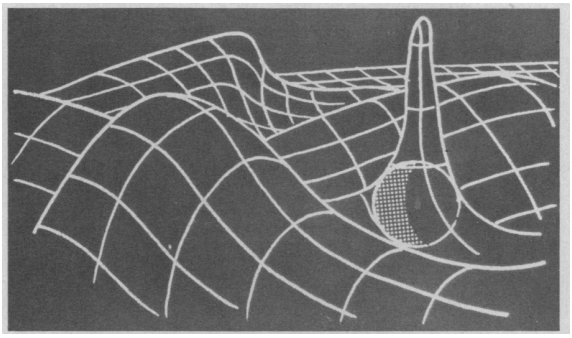
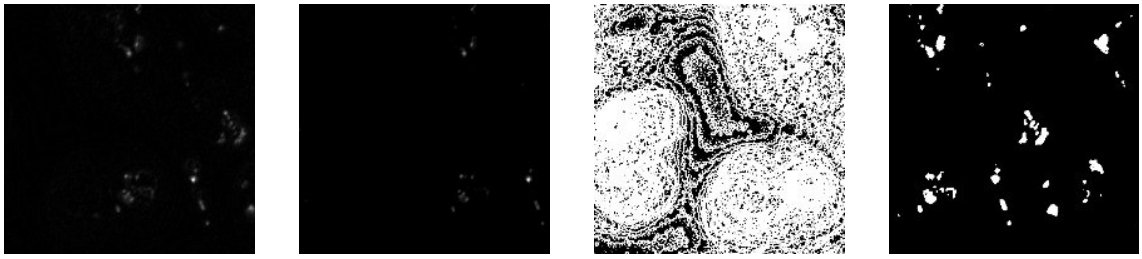


Figure 25: Rolling Ball



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

Figure 26: (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)

### 3.4.2 Training and predictions

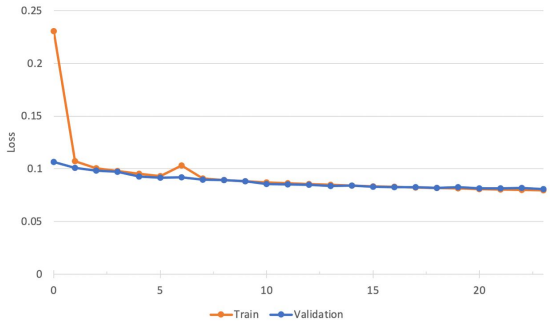



Figure 27: Straightforward training doesn't work

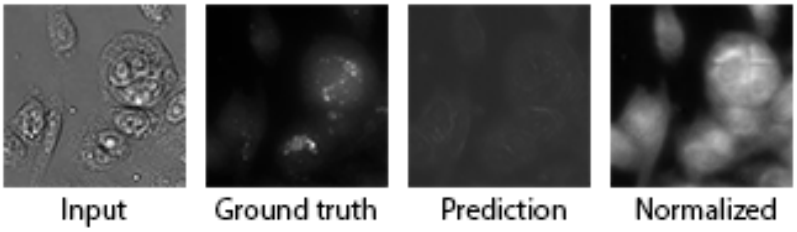



Figure 28: Training on original data

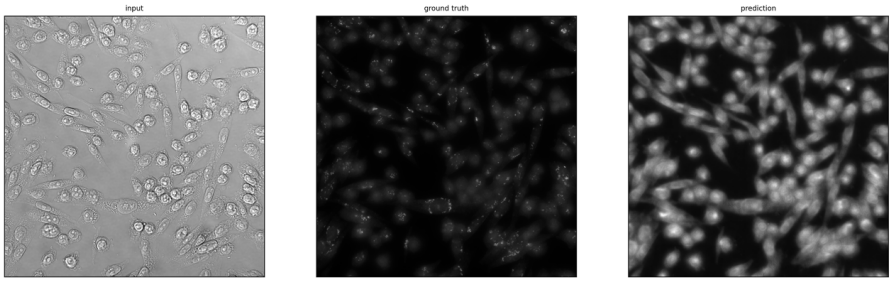



Figure 29: Full size predictions

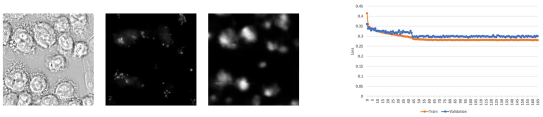


Figure 30: Training on the enhanced data

### 3.4.3 Alternative ways to improve predictions

#### 3.4.3.1 Asymmetrical losses

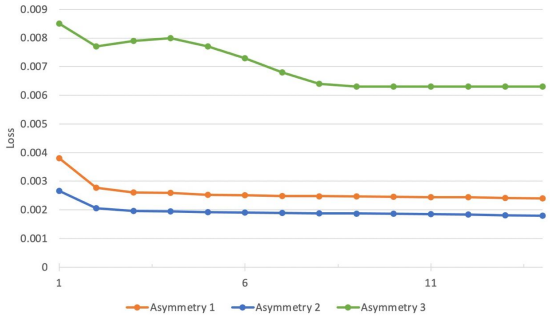



Figure 31: Asymmetrical training

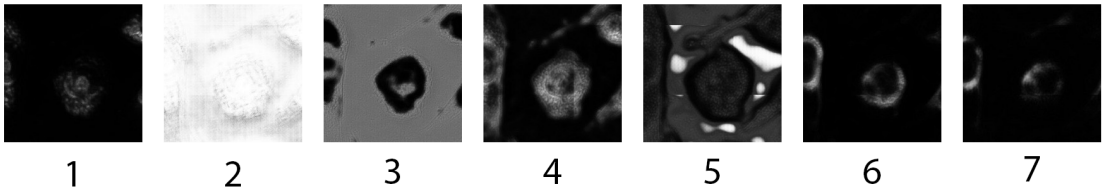



Figure 32: Asymmetrical training predictions

#### 3.4.3.2 Use of gradient in loss

#### 3.4.3.3 Noise reduction methods

## 3.5 GFP

#### 3.5.0.1 Preprocessing

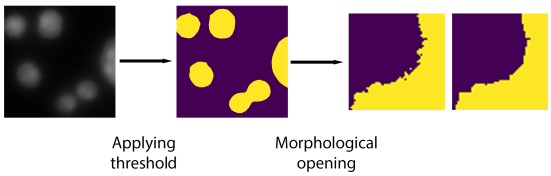



Figure 33: Converting GFP to a binary mask

#### 3.5.0.2 Predictions

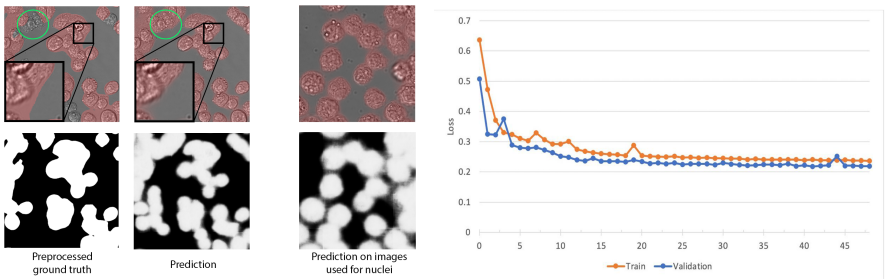



Figure 34: Training with BCE loss

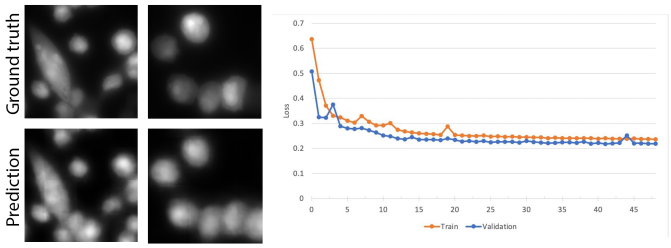



Figure 35: Training with Pearson correlation loss

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 |

#### 3.5.0.3 Downstream metrics

TODO move to separate chapter?

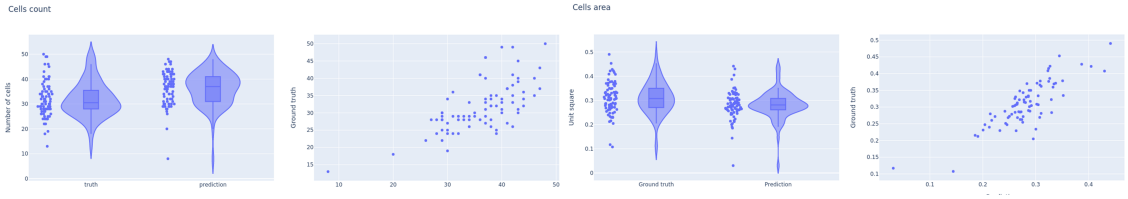



Figure 36: Downstream metrics

#### 3.5.0.4 Combination of GFP, nuclei and ER

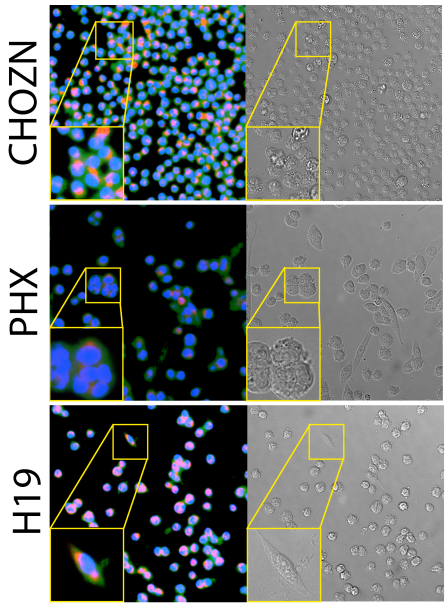



Figure 37: GFP, Nuclei and ER combined

## 3.6 Model evaluation

### 3.6.1 Metrics for downstream tasks

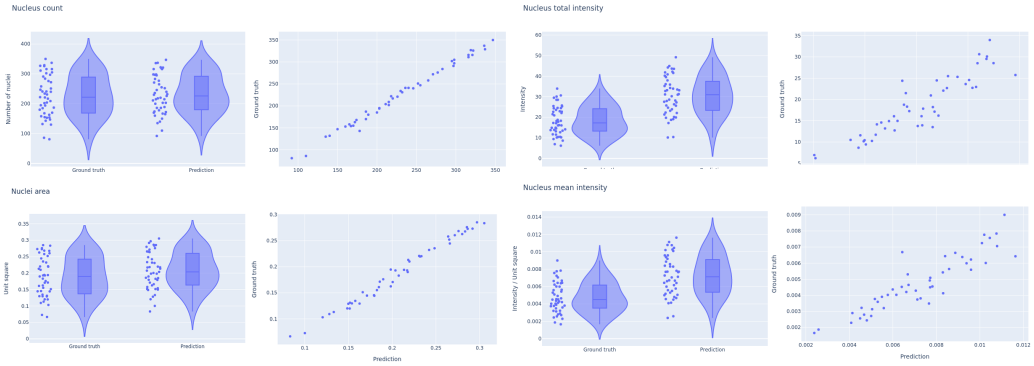



Figure 38: Metrics for downstream tasks on nuclei

Table 4: Correlation coefficients for downstream tasks on nuclei

| | 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 |

### 3.6.2 Influence of different loss functions on metrics for downstream tasks

# 4 Stability study

## 4.1 Stability study

### 4.1.1 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 |

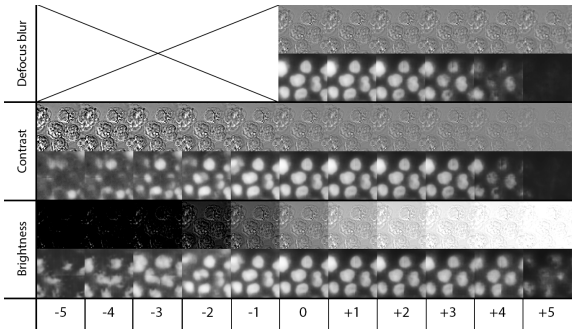



Figure 39: Influence of artificial corruptions on the predictions

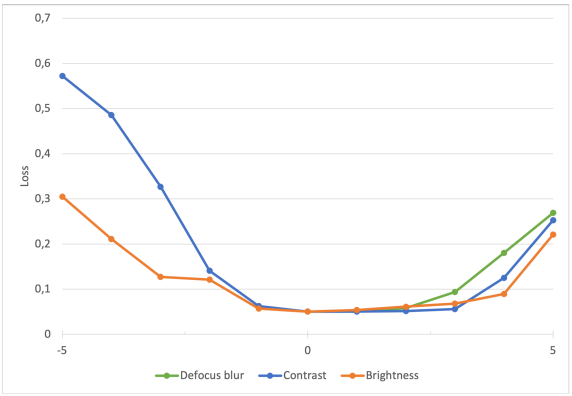



Figure 40: Change of Pearson correlation loss for artificial corruptions

### 4.1.2 Real corruptions

#### 4.1.2.1 Not fixed cells imaging as corrupted input

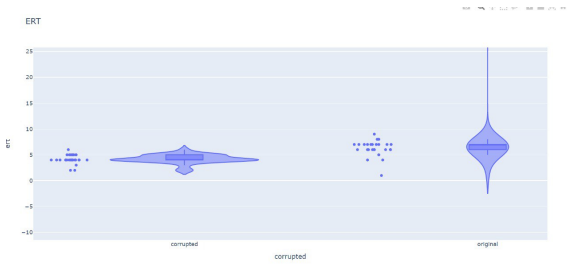



Figure 41: Online drift detection of not fixated cells

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.

#### 4.1.2.2 Real-world examples of corruptions

### 4.1.3 Influence of corruptions on metrics for downstream tasks

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

### 4.1.4 Improving predictions with additional corruption augmentations

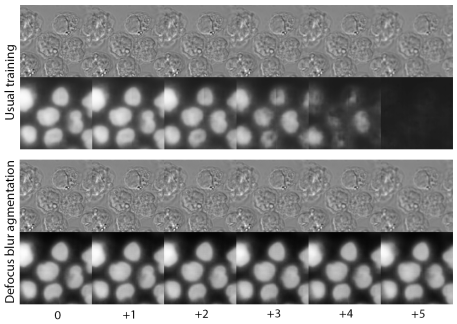



Figure 42: Using corruptions as augmentations improves predictions

## 4.2 UNET embeddings study

### 4.2.1 Application of various dimentionality reduction methods

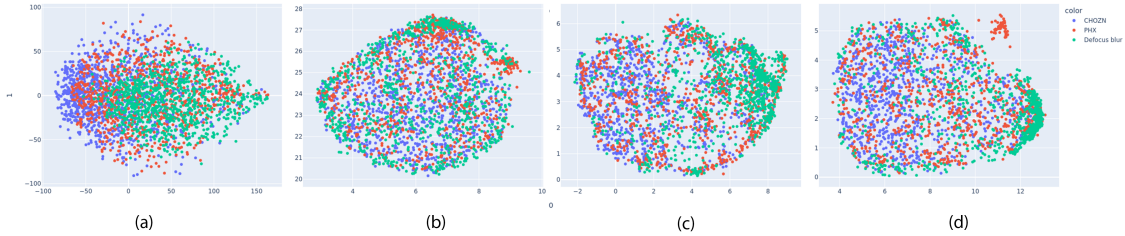



Figure 43: (a) PCA, (b) UMAP, (c) combination of PCA and UMAP with 10 and (d) 50 components

### 4.2.2 Autoencoder embeddings as an alternative

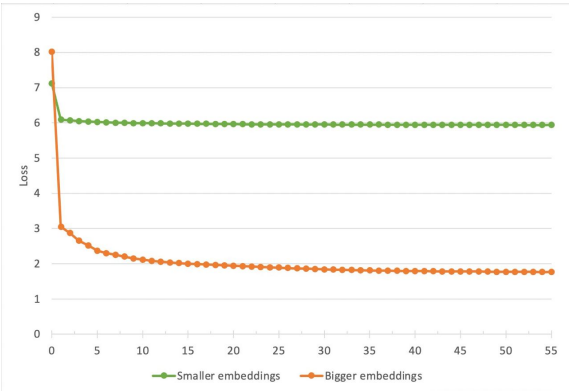



Figure 44: Autoencoders training convergence

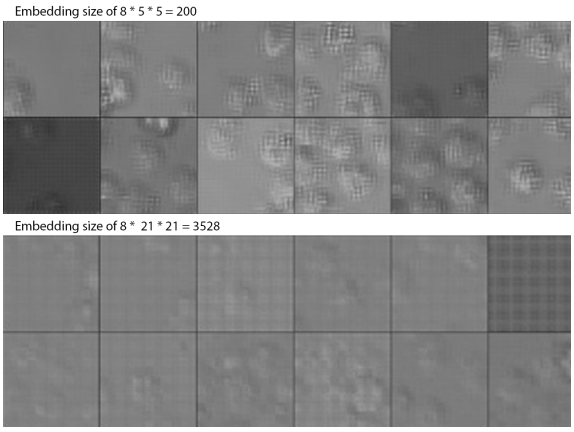



Figure 45: Samples drawn from the trained autoencoder

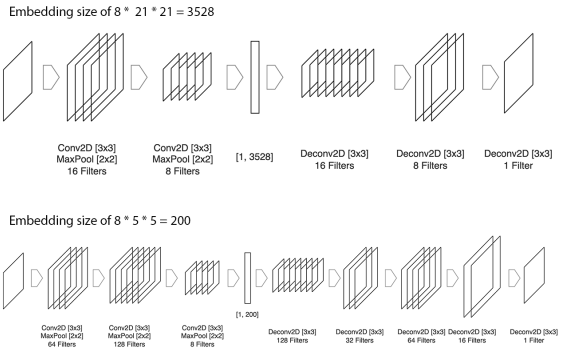



Figure 46: Architectures of two autoencoders

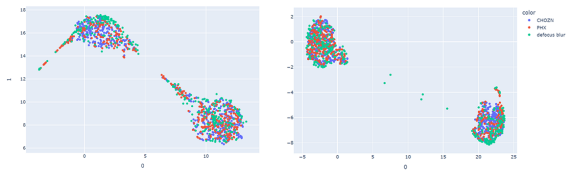


Figure 47: Autoencoder embeddings after applying PCA with 10 components and UMAP afterwards. Earlier epoch VS later epoch

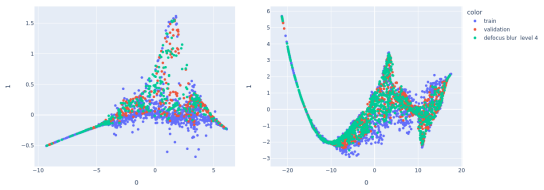



Figure 48: PacMAP does not provide information on the coruption

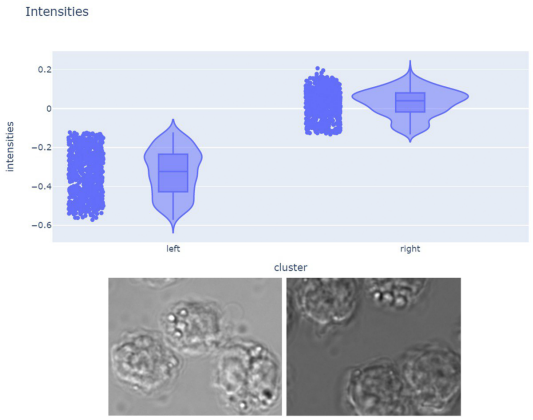



Figure 49: What do two UMAP clusters represent

### 4.2.3 Clustering of PacMAP embeddings

#### 4.2.3.1 Clustering on UNet embeddings

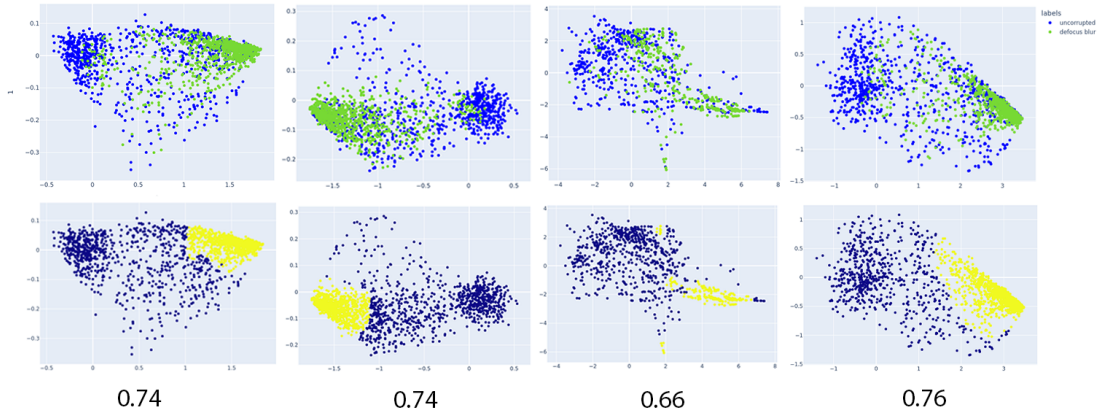



Figure 50: Clustering of UNet embeddings after PacMAP

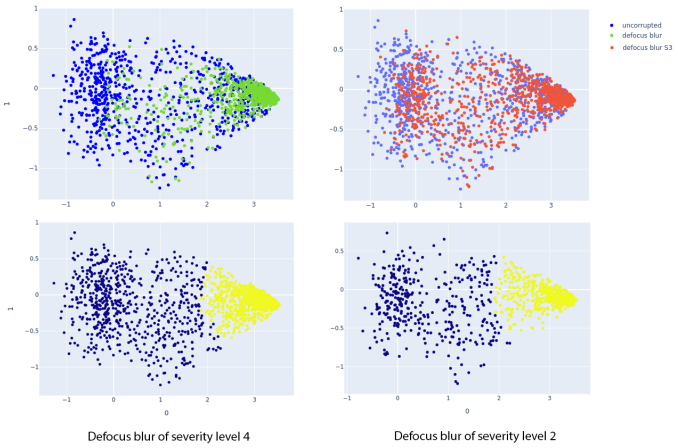



Figure 51: Clustering of UNet embeddings after PacMAP for different severities levels

TABLE with F1-score: 0.76 VS 0.64

## 4.3 Drift detection

### 4.3.1 A need to detect drift

### 4.3.2 Maximum mean discrepancy for drift detection

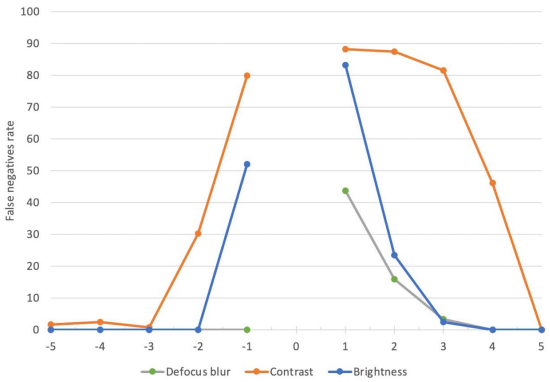



Figure 52: False negatives rate for drift detection on artificial corruptions

### 4.3.3 Online version of MMD algorithm

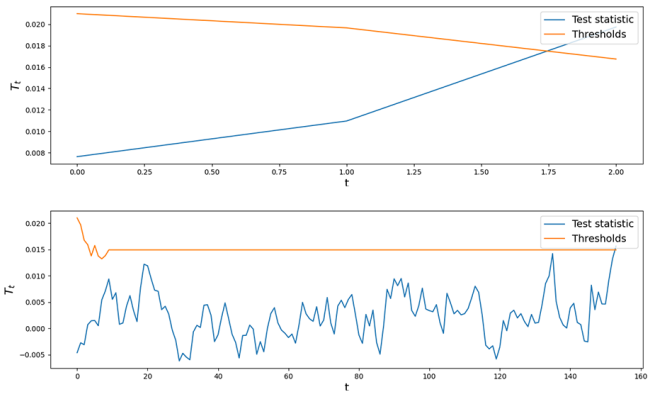



Figure 53: Expected runtime (ERT) for corrupted and in-distribution data

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 |

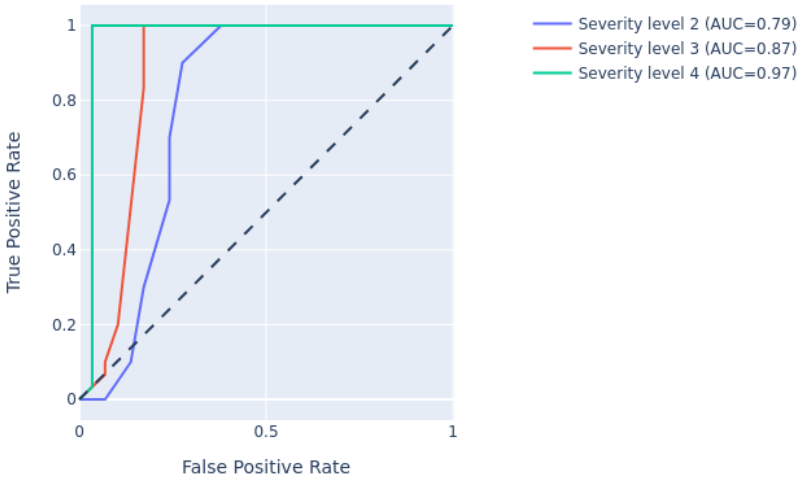



Figure 54: AUC ROC scores for various defocus corruptions severities

# 5 Software Tools

## 5.1 Foundry. Palantir

## 5.2 AWS

## 5.3 Streamlit

# 6 Future research

# 7 Summary

# List of Figures

## List of Tables