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

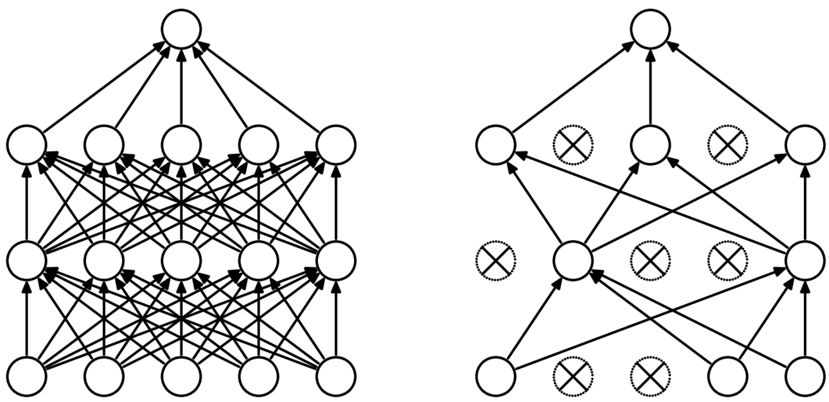
Description of my project, why is it useful, what are the processes here. How my neural network can be used for further stability predictions.

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

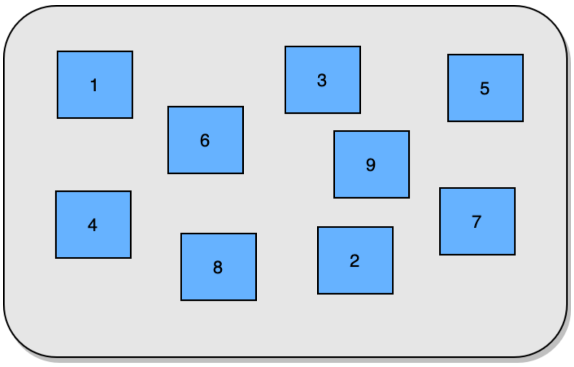


Figure 2: Way in which photos of the well-plate were taken

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

3 Model training

3.1 Neural network architecture

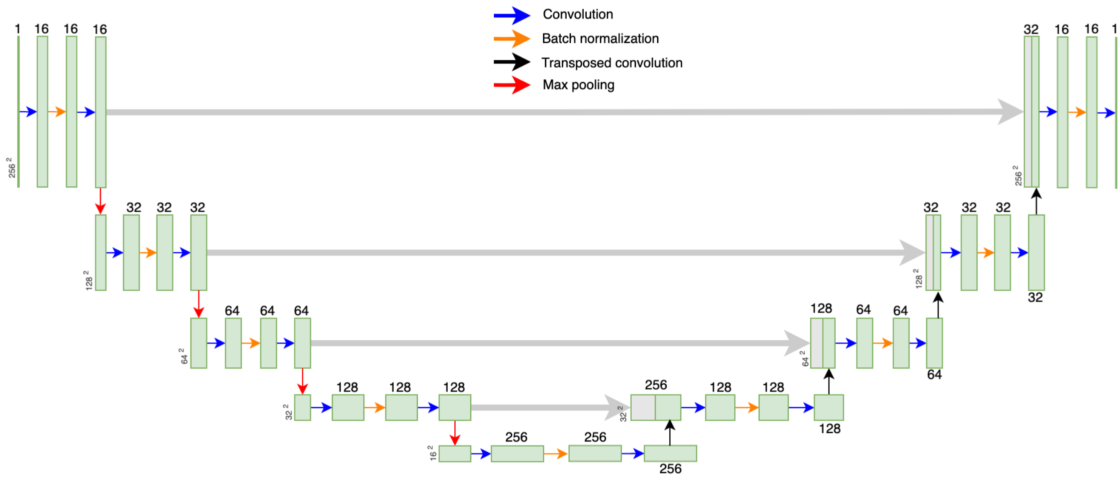


Figure 3: Unet

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.

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

Comparison on different weights Initialization. Was the wi suggested in the paper worth it?

3.7.2 Regularization

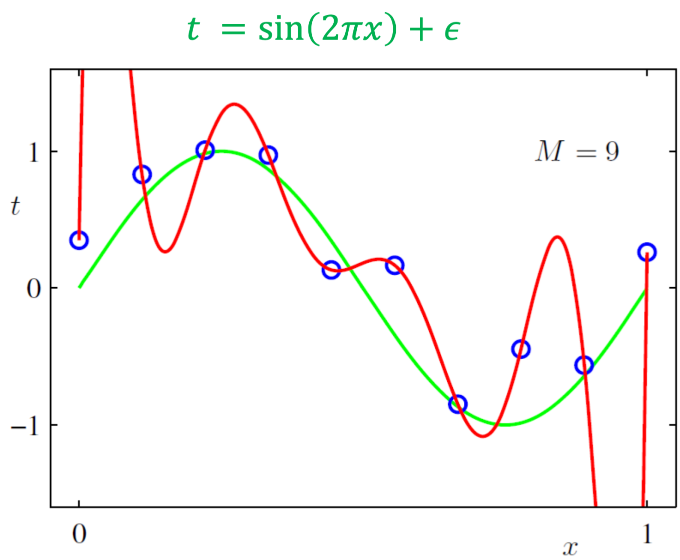


Figure 4: Overfitting

3.7.3 Optimizers

Comparison of different optimizers

4 Nuclei

4.1 Preprocessing

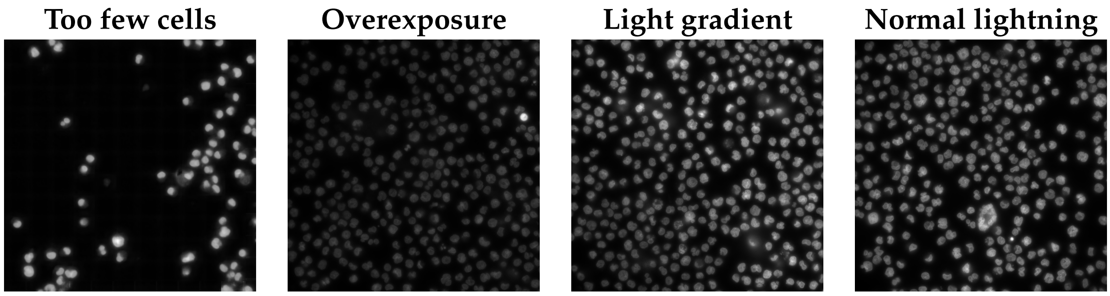


Figure 5: Different lightning conditions

4.1.1 Thresholding algorithms

Global and local thresholding

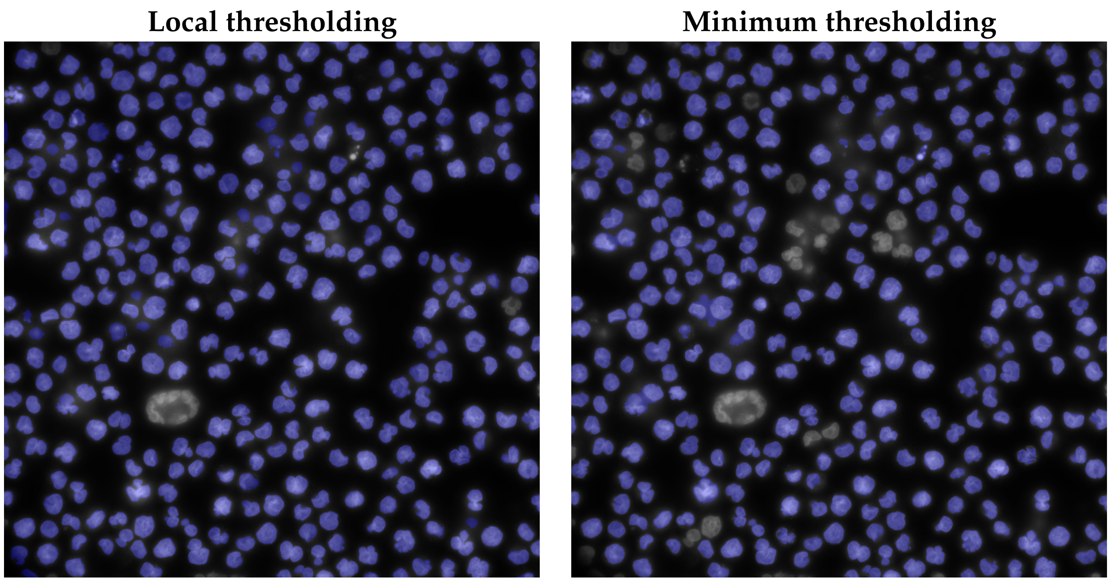
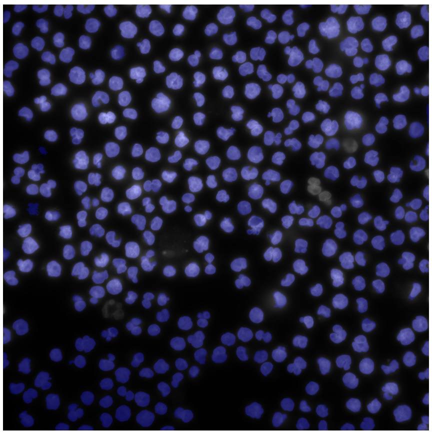
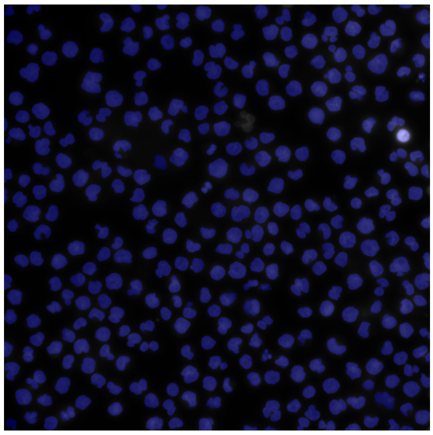


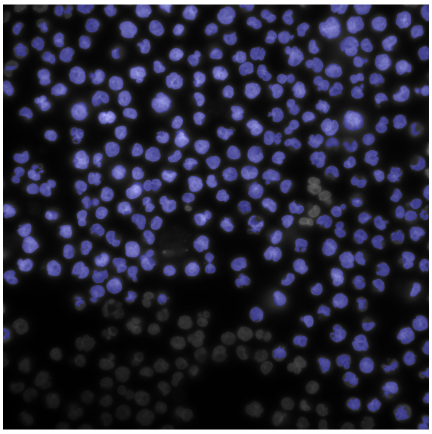
Figure 7: Local vs. Global thresholding (normal conditions)



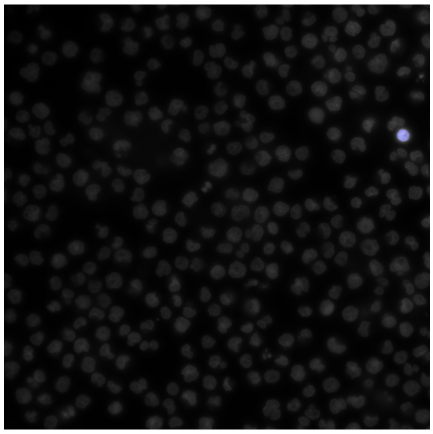
(a) Local



(b) Local



(c) Global, minimum



(d) Global, minimum

Figure 6: Local vs. Global thresholding

4.2 Training and predictions

4.2.1 Convergence

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

4.2.2 Predictions quality

Blurry, boundaries, not enough of details and possible improvements

4.3 Postprocessing for nuclei segmentation

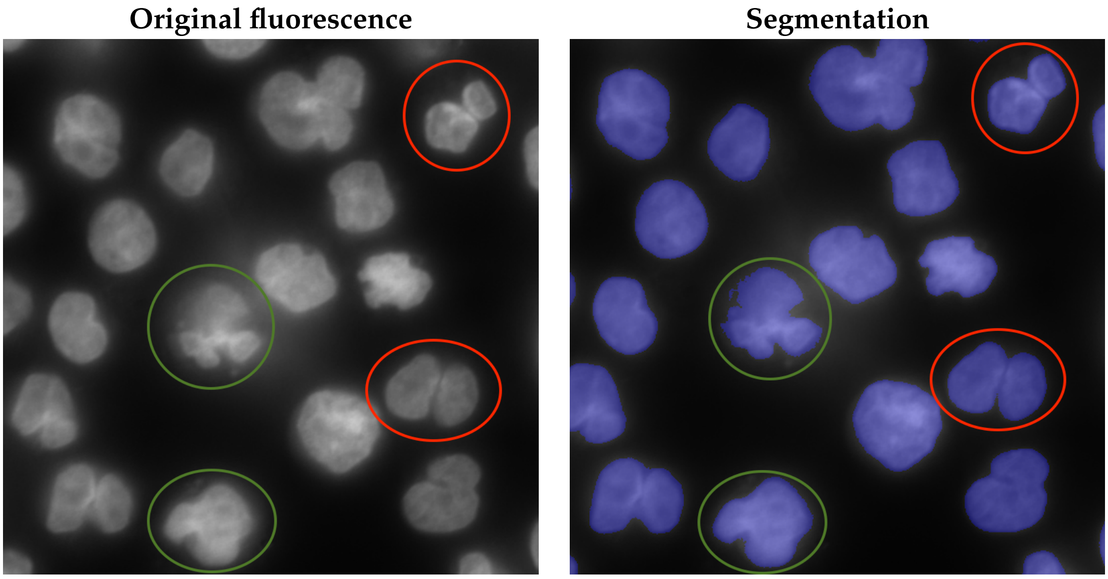


Figure 8: Closely located cells

Overall algorithm

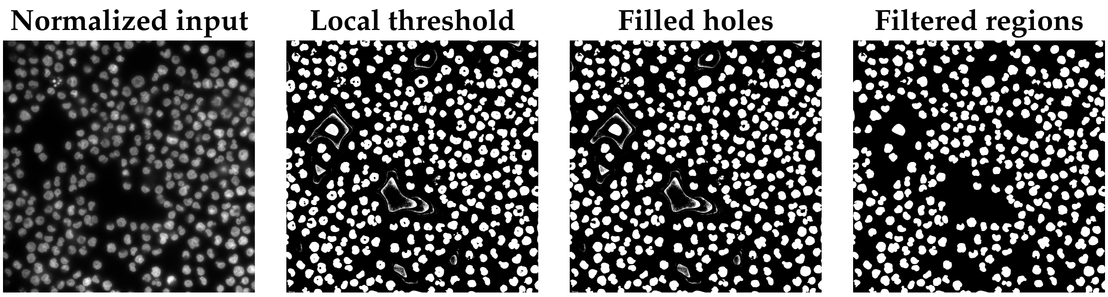


Figure 9: Fluorescence segmentation

4.4 Influence of scaling on predictions quality

Examples of predictions quality with different scales.

5 Actin

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

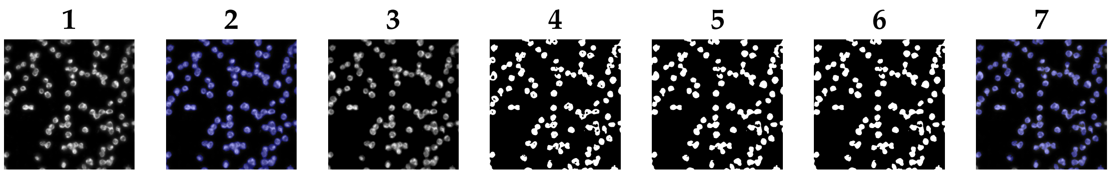


Figure 10: ER prediction

5.2 Training and predictions

5.3 Postprocessing

5.4 Combination of nuclei and actin predictions

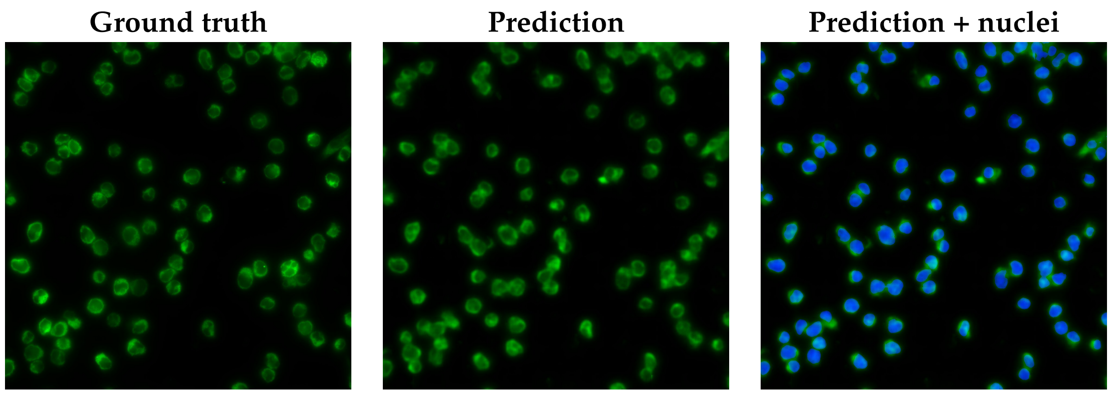


Figure 11: ER prediction

5.5 Prediction quality on the crop's border

5.6 Generalizability across phenotypes

6 Golgi

6.1 Preprocessing

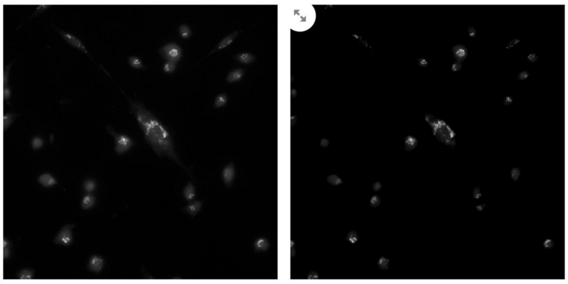


Figure 12: Golgi enhancement

6.1.1 Background removal algorithms

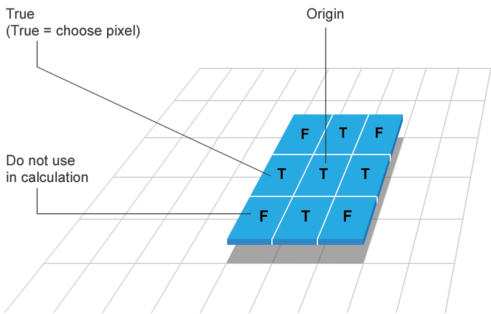


Figure 13: Structuring Element

Rolling ball algorithms

Rolling ball still leaves some noise

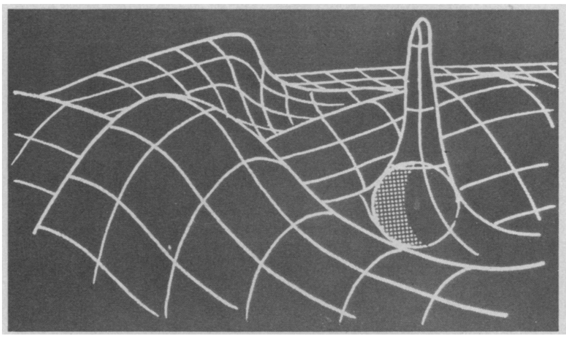
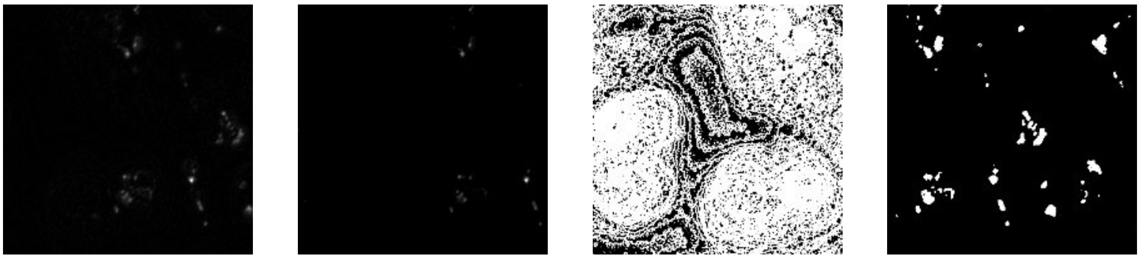


Figure 14: Rolling Ball



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

Figure 15: (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 Predictions

6.3 Postprocessing

6.4 Alternative ways to improve predictions

6.4.1 Noise reduction methods

6.4.2 Asymmetrical losses

6.4.3 Use of gradient in loss

7 Nucleuolus and full cell prediction

7.1 Preprocessing

7.2 Predictions

7.3 Postprocessing

7.4 Combination of nucleuolus and nuclei

8 Model Evaluation

8.1 Crops combination technique

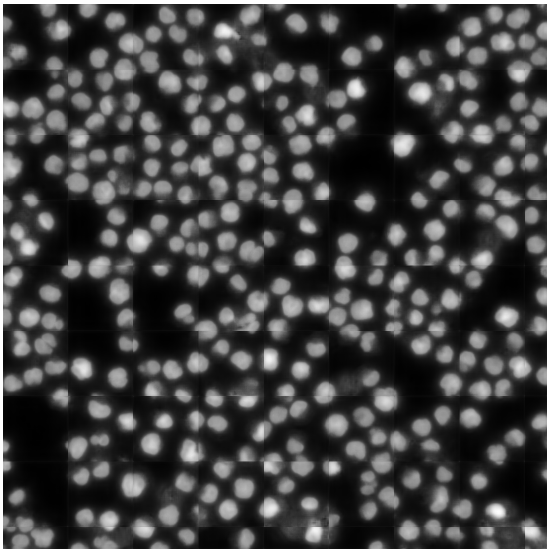


Figure 16: No overlap

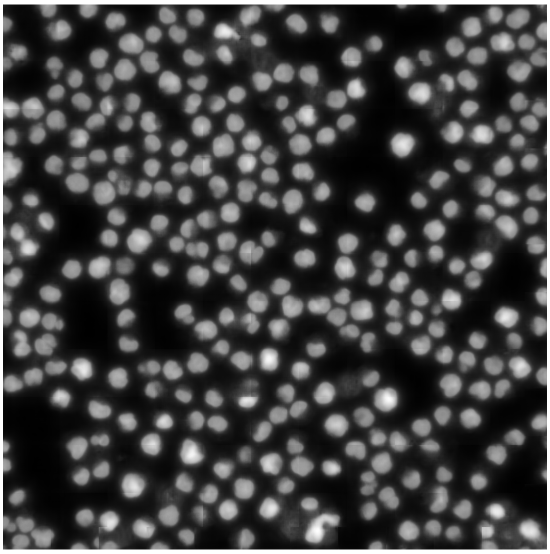


Figure 17: 30 pixels overlap

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

8.2 Metrics for downstream tasks

Which downstream tasks are there and how to evaluate them? All violin plots go here

8.3 Influence of different loss functions on metrics for downstream tasks

8.4 Influence of corruptions on metrics for downstream tasks

8.5 Improving predictions with additional corruption augmentations

9 Information in the UNET embeddings

9.1 Dimensionality reduction methods

9.2 UMAP, t-SNE, PCA

9.3 Autoencoder embeddings as an alternative

TODO add the second training loss for the second autoencoder

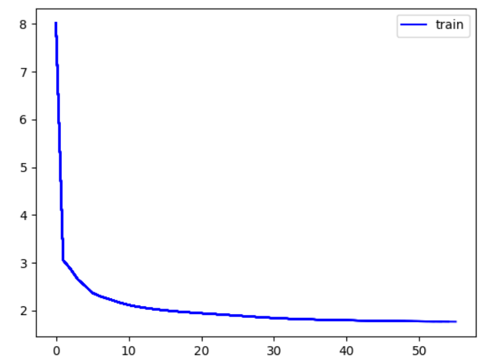


Figure 18: Autoencoder training convergence

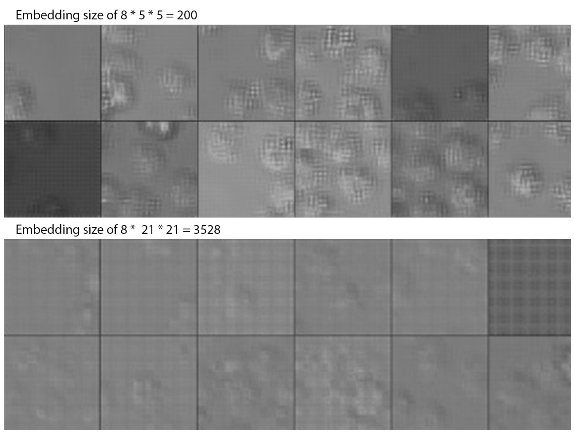


Figure 19: Samples drawn from the trained autoencoder

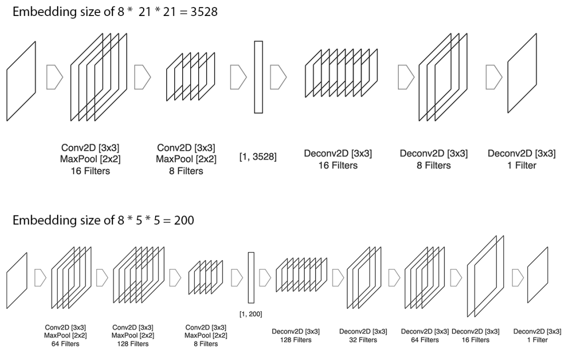


Figure 20: Architectures of two autoencoders

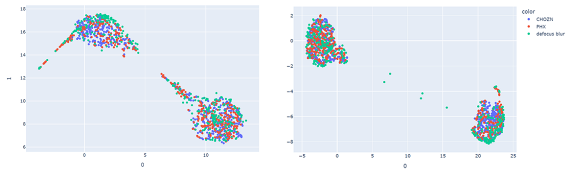


Figure 21: Autoencoder embeddings after applying PCA with 10 components and UMAP afterwards. Earlier epoch VS later epoch

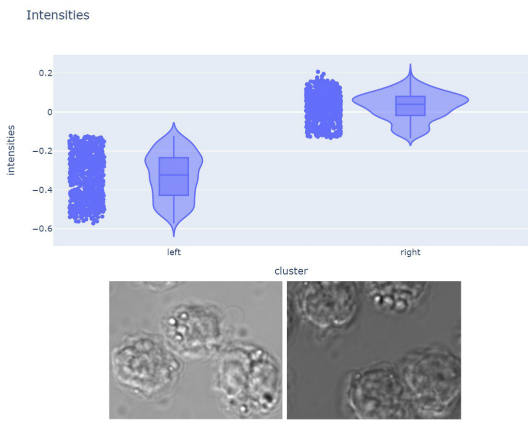


Figure 22: What do two UMAP clusters represent

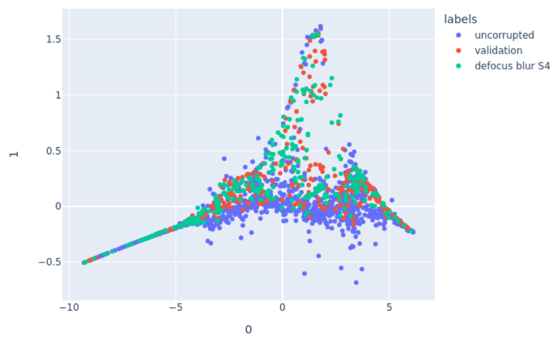


Figure 23: PacMAP does not provide information on the coruption

9.4 Clustering of corrupted samples

10 Drift detection

10.1 A need to detect drift

10.2 MMD

10.2.1 Offline drift detection

10.2.2 Online drift detection

10.3 MMD on UNet embeddings

10.4 Clustering on UNet embeddings

10.5 Clustering on autoencoder embeddings

10.6 Real corruptions

11 Software Tools

11.1 Foundry. Palantir

11.2 AWS

11.3 Streamlit

11.4 ImageJ, CellProfiler

12 Summary

List of Figures

List of Tables