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. Gianni Klesse
	Dr.-Ing. 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

Nowadays there is a great need for high-volume and high-quality recombinant protein production, yet cell line development (CLD) processes used for this are expensive and time-consuming. The main objective of this thesis was to provide a proof of concept in application of a novel in-silico fluorescence labeling approach to simplify the existing procedures of clone selection step in CLD. As a result, a solution to reduce phototoxicity as well as time and expenses needed for cell analysis was offered by making staining of cells completely redundant. The experiments were carried out with the use of state-of-the-art deep learning models to predict four fluorescent targets in CHO cells based on their DIC imaging only. For their better evaluation more practical biological metrics were proposed. The reliability of the models was studied and a possibility to detect drift within the data was provided via statistical two-sample hypothesis testing.This research showed that manual staining process of three cell targets at Merck KGaA can be in this case fully substituted by *in silico* fluorescence labeling models that were developed here. In case of Golgi apparatus target, it was recommended to acquire more data with a better signal-to-noise ratio. The results of this research showed successful development of data drift detection mechanisms that can alarm the end-user in case of wrong microscopy data acquisition settings. The results are of direct practical relevance to subsequent productivity predictions used for significant acceleration the CLD process.

Key words: *in-silico* fluorescence labeling, microscopy, deep learning, computer vision, CHO cells, drift detection.

Contents

1 Introduction

1.1 Motivation

Nowadays recombinant proteins are widespread in biomedical research and production of medicines like vaccines and antibodies that are used in the variety of therapeutic needs (Liu et al., 2022, Kim et al., 2011, Jayapal et al., 2007). Therefore there is currently a great need for high-volume and high-quality recombinant protein production. Optimization and improvement of cell line development (CLD) as the main process used for the production of recombinant proteins is therefore extremely important and relevant.

Clone screening is a step of CLD process in which cells are analyzed for further selection of the most stable and productive clones. This process includes differential interference contrast (DIC) microscopy of the cell and analysis of cell structure in terms of its organelles characterization. Imaging of cell organelles can be acquired via fluorescence microscopy after a staining procedure that allows to highlight an organelle of interest in fluorescence spectrum. However, it is not only expensive and time-consuming procedure, but is also toxic for the cells. Automating fluorescence microscopy via neural networks *in silico* significantly simplifies the existing procedure of clone selection, reducing phototoxicity, time and expenses needed for the analysis. Additionally, microscopy image acquisition errors can arise quite easily due to the sensitivity of the microscope settings as well as the cell phenotypes, scaling and cell fixation procedures. Therefore, further search for improvements in the process of automation of microscopy fluorescence and testing its robustness towards different input corruptions are urgent tasks.

The goal of this thesis was to provide a proof of concept on whether an *in silico* approach to fluorescent labeling can substitute manual cell staining. Successful automation of this process would prove that all information needed further clone screening and selection already exists in DIC imaging and cell staining can be escaped entirely. The research carried out in this work was aimed towards the specific needs, pipelines and data used at Merck KGaA. While the general goal of the project is to predict cell productivity and stability, this study provides not only a solution to escape manual cell staining, but also features from DIC imaging that can be used for productivity prediction directly. There are four cell targets of interest for this project that should be predicted based on DIC microscopy: nuclei, endoplasmic reticulum, Golgi apparatus and full cell surface. The aim of this study that differentiates it from the similar studies like LaChance et al., 2020 and Christiansen et al., 2018 was to not only provide deep learning models for the fluorescence predictions, but to study their reliability and to be able to detect drift during image acquisition.

This thesis is laid out as follows: Section 2 reviews the biological domain knowledge of CLD processes, machine and deep learning concepts used for image analysis; Section 3 provides an overview of the implementation and the results of *in silico* fluorescence predictions; Section 4 explores stability of the deep learning models developed in the previous section and provides valuable insights on the information from their embeddings; Section 5 shows possible future research questions that arose from the current analysis and Section 6 provides concluding remarks and succinct recommendations.

2 Background

The *in silico* fluorescence labeling approach has proven to be very promising as a substitute to the manual cell staining processes. For example, the research of Christiansen et al., 2018 did not only prove successful prediction of different cell stains with a variety of modalities and cell types, but it had also successfully determined cell viability. Nevertheless, the study is limited mainly to transmitted light (TL) z-stack imaging. This refers to the networks input being comprised of 3D images, which is not the case in this work, where only DIC imaging is used. Ounkomol et al., 2018 too shows successful predictions of several organelles in bright-field TL 3D images using 3D convolutional neural networks. However, switching to 2D data did not yield adequate results for them. More recent studies like Ugawa et al., 2021 provide an application of label-free fluorescence predicting already at the sorting stage, when a high-throughput system sorts cells individually. However, only a single-pixel detector is used by this study, meaning that it captures a wave rather than an image. Nonetheless one can recover an image from it with a ghost motion imaging technique (Bromberg et al., 2009), although this is computationally expensive (Ota et al., 2018).

There are two very interesting for this research recent studies by Cheng et al., 2021 and LaChance et al., 2020 that align very well with the processes in the project pipeline of Merck KGaA. Even though the former study manages to reach a state-of-the art performance on label-free fluorescence reconstruction, it uses reflectance images from oblique dark-field illumination as the input, which is a more specific cell imaging approach. Still, this input provides higher structural contrast in comparison to any transmission technique (Boustany et al., 2010). The latter study uses an easier imaging technique (DIC imaging) as an input, which shows great results even with low-resolution data. Both of these studies provide results based not only on training metrics, but also on performance of the models for metrics used in the practical biological evaluation. This is very important in the label-free fluorescence labeling research and was not present in research before LaChance et al., 2020.

All of the studies mentioned above, as well as this work rely on the premise that the input imaging type (here DIC) contains enough information to predict the fluorescence signal from it. This is a reasonable assumption because DIC, as well as bright-field and phase contrast imaging, are very often used for determining cell morphology (Kasprowicz et al., 2017).

This chapter provides a brief overview of the biological background needed to understand the process of cell line development and the role of fluorescent *in silico* labeling of DIC cell images within. It also covers the fundamentals of deep and machine learning techniques used here including clustering and dimensionality reduction approaches, as well as the basics behind drift detection algorithms. At the end of the chapter, a brief summary of the microscopy image acquisition process used in the research is given.

2.1 Biology

2.1.1 Cell line development process

The cell line development (CLD) is a process of generating single cell-derived clones that produce high and consistent levels of target therapeutic protein (*Cell Line Development Services* 2022). Therapeutic proteins in this case are so-called recombinant proteins and they are widely used in biomedical research, the production of medication and for various therapeutic needs such as, for example, vaccines and monoclonal antibodies (mAbs) (Ohtake et al., 2013, Jefferis, 2017, Funaro et al., 1996). A recombinant protein, as defined by Barbeau, 2018, is a modified or manipulated protein encoded by a recombinant DNA. Recombinant DNA in turn consists of a plasmid, where the genes of the target protein of interest are cloned downstream of a promoter region. As soon as this plasmid is transfected to a host cell (for example some mammalian cells that are able to produce the protein), the host will start to express this protein of interest. Today there is a great need for the production of high volumes of good quality recombinant proteins, both in industrial as well as research contexts (Tihanyi et al., 2020). For this reason the goal of many research projects in recombinant protein production is to improve expression efficiency and create high-throughput systems to improve the CLD processes (Tihanyi et al., 2020).

One of the most popular host cells used in CLD and in this thesis specifically are chinese hamster ovary (CHO) cells (Castan et al., 2018). Although different cells can be used as hosts, such as bacterial, plant-based or yeast cells, mammalian cells remain the most popular choice. The reason behind this popularity resides in the fact that they can produce a diverse range of correctly folded proteins and most importantly they have high protein production rates. The productivity rate is measured in titre of produced protein, and CHO cells can reach 0.1 - 1 g/L in batch and 1 - 10 g/L in fed-batch cultures (Tihanyi et al., 2020). Mostly all of the mAbs are produced using CHO cells (Lalonde et al., 2017). Pharmaceutical companies very often try to use the same host cell line for all their productions because already checked and qualified cells simplify the road to the clinic (Tihanyi et al., 2020). Since this research is dedicated to CHO cell line as well, it has a wide applicability.

However, there is a downside to using CHO as host cells —- these rapidly growing immortal cells are also genomically unstable and extremely heterogeneous which usually leads to the main issue: production instability. Instability here means that the cell might die or do not produce target protein. The problem of choosing stable and high-production clones that simultaneously will be able to express protein qualitatively and quantifiably over time is essentially the main goal of current research. The challenge in manufacturing here is the time and the cost of production. Currently, a lot of research attention is dedicated to the reduction of both factors, as well as the development of techniques of high-throughput clone screening and characterization (Tihanyi et al., 2020). The latter is of interest for this thesis. With great amounts of data collected over time and the development of computational modelling and statistical analysis it is now possible to carry out the analysis *in silico*, meaning computationally without interfering with the cells instead of the usual *in vitro* analysis (Christiansen et al., 2018), which will be also shown in this research.

2.1.1.1 CLD steps

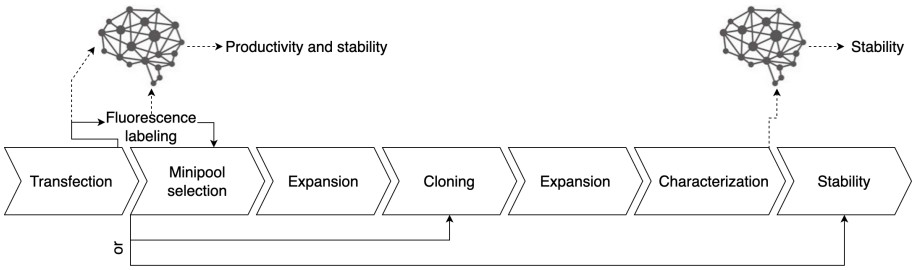


Figure 1: CLD process steps. Usual times needed for this process: from transfection to characterization — 5 months, stability — 3 months. Minipool selection step that is optimized here is taking up to 5 weeks.

The first step of CLD is called transfection — the introduction of the gene of interest (abbreviated as GOI or can be called a DNA vector or, alternatively, an expression vector) into CHO cells. There are two main challenges with this step: firstly, transfection mostly results in a vector being inserted into a random site within the host cell genome and secondly, it generally has low efficiency of integration (Tihanyi et al., 2020). It is important to transfect a GOI into the optimal site of the genome to secure high protein expression over time during protein production. Practically however, GOI is transfected into a random location of the genome. In cases where the gene was transfected into an inactive site of genome (essentially the majority of genome is transcriptionally inactive), the cell will likely be unable to express the gene (Castan et al., 2018, Hong et al., 2018).

The second step of the process is the selection of cell minipools that have successful and stable gene integrations for further expansion and cloning. The reason for not all of them being suitable is that during the transfection step, only 80% of the cells will receive a GOI vector (Castan et al., 2018). Only a small percentage of these cells actually integrate a vector into the genome and, as mentioned above, only a fraction of those are able to express the protein in a stable fashion (Shin et al., 2020). After the best minipools are selected, they will be expanded, which means that cell population is serially passaged to a larger population with the bigger number of cells.

The third step in CLD is to clone the cells. The selected stable pools of cells are phenotypically and genetically diverse. This means that they have different growth rates, metabolic profiles, and so forth. This is not ideal for industrial production - all the cells used for protein production should be derived from the same clone (Ema, 2020).

Once the cells are cloned, phenotypical and genetical heterogeneity is reduced, the next step is to characterize the cells for their expression of the GOI. One has to estimate the clones' productivity and stability. Such observations may take up to 90 days (usually stability measurements are made on the 30^{th}, 60^{th} and 90^{th} days). If the clones remain stable after this time and are able to express enough of the protein, then they are suitable for further production. However, this last step is very time-consuming and requires maintenance for feeding and analysing the cells. Predicting productivity and stability of the cells

in earlier stages would reduce this time significantly or even allow to avoid this process entirely.

2.1.2 Improving the CHOZN® platform at Merck KGaA

There are many different proteins that can be produced using CLD technologies, for example, vaccines, hormones etc. This research however is dedicated to the production of monoclonal antibodies (mAbs).

The CHOZN® platform is a widely used product of Merk KGaA. CHOZN® is a CHO mammalian cell expression system for fast and easy selection and growth of clones producing high levels of recombinant proteins (*CHOZN Platform — Technical Bulletin* 2022). The processes of developing expression systems on this platform correspond to the general CLD process described in the previous subsection 2.1.1.1. The scope of the overarching project at Merck KGaA is to simplify the labour-intensive and time-consuming process of stability measurement of the expression system by inducing predictions of productivity and stability rates during early steps in the CLD process.

After the transfection step there are several quantities that are measured in minipools in order to select the best ones. For example, cell size, its complexity, cell surface protein expression, endoplasmic reticulum (ER) mass, mitochondria mass, etc. For qualitative and quantitative characterization of cells, fluorescent labeling is used. It is a process of covalently binding fluorescent dyes to biomolecules such as nucleic acids or proteins, so that they can be visualized via fluorescence imaging (*Fluorescent labeling* 2022). A fluorophore is a chemical compound that can reemit light at a certain wavelength. These compounds are a critical tool in biology because they allow experimentators to capture particular components of a given cell in detail (Bharath Ramsundar, 2019 p.113). In order for fluorophore to enter the cell it has to be fixed. Fixation of the cells is a process of destroying their membrane and fixing them on the plate in order for the stain antibody to be able to enter the cell.

Unfortunately, fluorescence labeling is expensive in costs, time-consuming and may kill the cell due to its phototoxicity (Fried et al., 1982, Patil et al., 2018, Progatzky et al., 2013). Additionally, Jessna H. M. Yeo, 2017 found out that different selection markers affect the production stability of CHO cells. Other negative aspects of the manual staining approach include a limited number of available fluorescent channels in microscopes; some fluorophores have a spectral overlap, hence there is a limited number of detectable markers (Perfetto et al., 2004); such labels can be inconsistent (Burry, 2011, Weigert et al., 1970), and depend a lot on reagent quality and require many hours of lab work. Toxicity, for instance, is a very dangerous factor, especially for medicine production as it may even affect the final product. Because of this there is a need for an approach of *in silico* fluorescent labeling — computationally and without affecting the cell.

For *in silico* labeling, the input data is a DIC microscopy image. This is an optical microscopy technique used to enhance the contrast in unstained, transparent samples. This is a much cheaper image acquisition technique than a staining process, and it has much less variability as well (for example, no dependency on the dye or antibody quality) (Christiansen et al., 2018). The research carried out in the scope of this thesis is dedicated to

predicting fluorescence signal from the DIC imaging directly without the need of actual cell staining. The measurements needed for selecting the minipools can be calculated as usual, with the exception of using the predicted images instead.

2.2 Deep learning and machine learning basics

2.2.1 Deep learning for computer vision

Definition 2.1 (Image dataset)**.** An image dataset in the scope of this thesis constists of input DIC images X and target fluorescence images Y. Combined, couples from each form (X and Y) construct a dataset:

$$D = (X, Y) = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\} \tag{1}$$

where both $x^{(i)}$ and $y^{(i)} \in \mathbb{R}^{W \times H}$ are single images, N is the size of the dataset. Generally input data has a shape of (N, C, H, W), in this work $C = 1$.

Definition 2.2 (Model)**.** A model is a function with learnable parameters $\theta = (\theta_1, ..., \theta_K)$ where $\theta_i \in \mathbb{R}$ for $i \in 0, ..., K$ which approximates the mapping of initial data X to target data Y.

$$M(X, \theta) = Y' \approx Y \tag{2}$$

Definition 2.3 (Loss function)**.** A loss function is a function $L(y, M(x, \theta))$ of model's parameters θ, that for $(x^{(i)}, y^{(i)}) \in D$ outputs a scalar value measuring the difference between ground truth y and prediction $M(x, \theta)$. A training objective is then defined as an average over the loss of each training sample:

$$J(\theta) = \mathbb{E}_{(x,y)\sim D}\left[L(y, M(x, \theta))\right] \tag{3}$$

where p_{data} denotes an empirical distribution of the training data.

Definition 2.4 (Weight initialization)**.** Weight initialization is a process of setting the initial values of the parameters of a model to some random values.

Weight initialization plays a crucial role in model training. Even on the simplest model wrongly initialized weights (for example all constant or too large or too small) can lead to very slow convergence or prevent the model from converging at all (Kumar, 2017).

In the following samples *fan_in* denotes the maximum number of input signal units to a given layer and *fan_out* is the maximum number of output signal units from it.

Definition 2.5 (Xavier initialization)**.** Xavier initialization, which is usually a default choice in many neural networks, works well for the most part for fully connected layers with tanh as activation function. There is also a study providing some insights into why Xavier initialization may not be the optimal choice for ReLU activations (Kumar, 2017). Xavier initialization draws samples from a uniform distribution:

$$Uniform\left(-\frac{1}{\sqrt{fan\_in}}, \frac{1}{\sqrt{fan\_in}}\right) \tag{4}$$

Definition 2.6 (He initialization)**.** He initialization is another initialization method proposed by He et al., 2015 as it was noticed that Xavier initialization is not an optimal choice for the networks that include ReLU activations. The authors suggest a new robust method that enables training of even extremely deep or wide network architectures with ReLU activations. He initialization draws samples from a truncated normal distribution:

$$\mathcal{N}(0, \frac{2}{\text{fan\_in}}) \tag{5}$$

Default weight initialization of Conv2D layers in Python suggests to use the following initialization method (He et al., 2015):

$$std = \sqrt{\frac{2}{fan\_in}} \tag{6}$$

$$bound = \sqrt{3 * std} \tag{7}$$

$$Uniform\,(-bound, bound) \tag{8}$$

Although in the study of He et al., 2015 it is called Kaiming normal initialization, it is a slightly different method.

Definition 2.7 (Binary-cross entropy loss)**.** Let $y \in \mathbb{R}^{W \times H}$ be a ground truth image and $y' \in \mathbb{R}^{W \times H}$ be a prediction. Binary-cross entropy loss is defined as:

$$L(y, y') = -\frac{1}{N^2} \sum_{i=1}^{H} \sum_{j=1}^{W} y_{i,j} \cdot \log\big(y'_{i,j}\big) + (1 - y_{i,j}) \cdot \log\big(1 - y'_{i,j}\big) \tag{9}$$

Definition 2.8 (MSE (mean squared error) loss)**.** Let $y \in \mathbb{R}^{W \times H}$ be the ground truth and $y' \in \mathbb{R}^{W \times H}$ be the predicted images. The MSE loss is defined as:

$$L(y, y') = \sum_{i=1}^{H} \sum_{j=1}^{W} (y_{i,j} - y'_{i,j})^2 \tag{10}$$

Definition 2.9 (PCC (Pearson correlation coefficient) loss)**.** Let $y \in \mathbb{R}^{WH}$ be a flattened ground truth and $y' \in \mathbb{R}^{WH}$ be a flattened predicted image. The PCC loss is defined as:

$$PCC(y, y') = \frac{\sum_{i=1}^{W \times H} (y_i - \bar{y})(y'_i - \bar{y}')}{\sqrt{\sum_{i=1}^{W \times H} (y_i - \bar{y})^2 \sum_{i=1}^{W \times H} (y'_i - \bar{y}')^2}} \tag{11}$$

$$L(y, y') = \frac{1 - PCC(y, y')}{2} \tag{12}$$

where $\bar{y}$, $\bar{y}'$ are means of the ground truth and predicted images respectively.

There is an important distinction to be made here: firstly, Pearson correlation coefficient (PCC further) in a measure of similarity between two data sequences (matrices in this case), with values between -1 and 1, with 1 being a positive correlation, secondly, PCC

loss is a measure of dissimilarity between two matrices, with values between 0 and 1, with 0 meaning that matrices are the same.

This loss is widely used in cell biology for comparison of co-localization between the proteins (LaChance et al., 2020). PCC is also popular in computer vision where it is utilized for the determination of image similarity in terms of spatial-intensity (LaChance et al., 2020).

Definition 2.10 (Optimization)**.** Optimization is a process of updating the parameters θ of the model $M(X, \theta)$ to minimize the loss function $L(y, M(x, \theta))$.

With a maximum likelihood esimation, we get:

$$\theta_{MLE} = \arg\max_{\theta} \sum_{i=1}^{N} \log p_{\text{model}}(x^{(i)}, y^{(i)}, \theta) \tag{13}$$

After maximizing the sum and taking a gradient one gets:

$$\nabla_\theta J(\theta) = \mathbb{E}_{x,y\sim p_{data}} \left[\nabla_\theta \log p_{\text{model}}(x, y, \theta)\right] \tag{14}$$

The exact gradient on a discretized data-generating distribution is then:

$$g = \nabla_\theta J^*(\theta) = \sum_x \sum_y p_{\text{data}}(x, y) \nabla_\theta L(y, M(x, \theta)) \tag{15}$$

Here one can obtain an unbiased estimator of a true gradient on a mini-batch of i.i.d. samples $\{x^{(i)}, ..., x^{(m)}\}$

$$\hat{g} = \frac{1}{m} \nabla_\theta \sum_i L(y^{(i)}, M(x^{(i)}, \theta)) \tag{16}$$

Definition 2.11 (Stochastic gradient descent)**.** Stochastic gradient descent is an optimization algorithm where the parameters θ are iteratively updated every mini-batch of data by the following rule:

$$\theta_{k+1} = \theta_k - \alpha \frac{1}{m} \nabla_\theta \sum_i L(y^{(i)}, M(x^{(i)}, \theta)) \tag{17}$$

where α is a tuneable parameter called *learning rate*.

Definition 2.12 (Adadelta optimizer)**.** An Adadelta optimizer is a more sophisticated optimization technique, that follows algorithm 1 for the parameter update.

Algorithm 1 Adadelta optimization

1. Initialize: $E[g]_0^2 = 0$ and $E[\Delta\theta^2]_0 = 0$

2. Compute gradient: g_t

3. Accumulate gradient: $E[g]_t^2 = \rho E[g]_{t-1}^2 + (1-\rho)g_t^2$

4. Compute update: $\Delta\theta_t = \frac{\text{RMS}[\Delta\theta]_{t-1}}{\text{RMS}[g]_t}\hat{g}_t$

5. Accumulate updates: $E[\Delta\theta^2]_t = \rho E[\Delta\theta^2]_{t-1} + (1-\rho)\Delta\theta_t^2$

6. Apply update: $\theta_{t+1} = \theta_t + \Delta\theta_t$
RMS here is the root mean square all initial hyperparametes are take from the original study(Zeiler, 2012).

Definition 2.13 (Adam optimizer)**.** An Adam optimizer is another stohastic optimization technique, that has the following hyperparameters: α — learning rate, $\beta_1, \beta_2 \in [0, 1)$ — exponential decay rates. It follows algorithm 2 for the parameter update.

Algorithm 2 Adam optimization

1. Initialize: $m_0 = 0$ and $v_0 = 0$

2. Compute gradient: g_t

3. Update biased first moment estimate: $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$

4. Update biased second raw moment estimate: $v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2$

5. Compute bias corrected first moment estimate: $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$

6. Compute bias corrected second raw moment estimate: $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$

7. Apply update: $\theta_{t+1} = \theta_t - \alpha\frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$ Initial hyperparameters used in this work are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.

Definition 2.14 (Overfitting)**.** Overfitting is a phenomenon in which a hypothesis that fits training samples well will perform worse over the entire distribution on data rather than another hypothesis that fits the distribution of the training samples less well (Mitchell, 1997). The way to avoid overfitting that happened to the models in Section 3.3 are discussed in Section 3.1.4.2.

Definition 2.15 (Feedforward fully connected layer)**.** A feedforward fully connected layer is a trainable function with parameters $W \in \mathbb{R}^{N\times M}$ (weights) and $b \in \mathbb{R}^M$ (biases) that, in this case, maps a vector $x \in \mathbb{R}^N$ to an output $a \in \mathbb{R}^M$ via the following transformation:

$$a = W^T x + b \tag{18}$$

This is one of the simplest layers in a feedforward neural networks and input and output in it as mentioned above are vectors. However, in this study inputs and outputs are images, that are represented in memory as square matrices $x^{(i)}, y^{(i)} \in \mathbb{R}^{N\times N}$. One could simply flatten the image into a vector and use it as an input to a fully connected feedforward neural network. Nevertheless this would be a suboptimal approach.

Since essentially one of the main tasks of this research is to create a deep learning model that is able to predict a fluorescence image from a DIC image, the problem statement could be narrowed down to the following: predict an intensity high-resolution image from another intensity high-resolution image based on the features of the object morphology in it. Such problem is very common in the field of image analysis and one of the popular deep learning tools for solving such problems is convolutional neural network (CNN) or more specifically a UNet.

CNNs are able to capture nonlinear relationships over large areas of images, they greatly improve performance for image recognition tasks in comparison to classical machine learning methods (Ounkomol et al., 2018). The word "convolutional" suggests that the convolution operation should be used in at least one of the layers there.

Definition 2.16 (Convolutional layer)**.** A convolutional layer is a trainable function with parametrized kernel $K \in \mathbb{R}^{F \times G \times C}$ and bias $b \in \mathbb{R}$ that is usually denoted via the operator $(\cdot * \cdot)$. By transforming an input $x \in \mathbb{R}^{W \times H \times C}$ it produces an output S

$$S = K * x + b \tag{19}$$

that is called a *feature map* where an element on position (i, j) is defined as follows:

$$S_{i,j} = \sum_{w} \sum_{h} x_{m,n} K_{i-m,j-n} \tag{20}$$

Convolutional layer like a fully connected layer can be viewed a linear transformation as well. However, there are three main advantages that leverage convolutional layers for image processing in comparison to fully connected layers: sparse interactions, parameter sharing and equivariant representations. An image is a very redundant way of representing the semantic meaning hidden within it. Having a value of one pixel, the probability that the neighboring one will be of the same color is very high. Sparsity of interactions can be described by an example: usually a high-resolution image might have millions of pixels, however it is possible to detect smaller and very important features like contrast changes, edges, and shapes using a kernel consisting of only a few hundred pixels. By applying kernels (or filters) on the image locally, one will infer many of these features across the whole image. Such an approach reduces the memory needed for parameter storing and improves its statistical efficiency (Goodfellow et al., 2016). Parameter sharing refers to the fact that instead of learning a separate set of parameters for every location within the image, only one set of parameters will be learned and applied across all image locations. Lastly, equivariance here means that convolution operation is equivarient to the shifts in the image.

Definition 2.17 (Stride)**.** During the computation of convolution, the kernel starts sliding at the upper left corner of the input tensor, covering all locations while heading to the right and down. The step with which the window slides is called *stride*.

Definition 2.18 (Padding)**.** When convolution is applied several points on the perimeter of the input tensor will be lost and the ouput tensor will have smaller spatial dimension than the input one. One can fix this by adding a few more pixels outside the perimeter, to preserve the dimension of the output to be same as input. The amount of pixels added is called *padding*.

Visual examples of what stride and padding represent are shown in Figure 2.

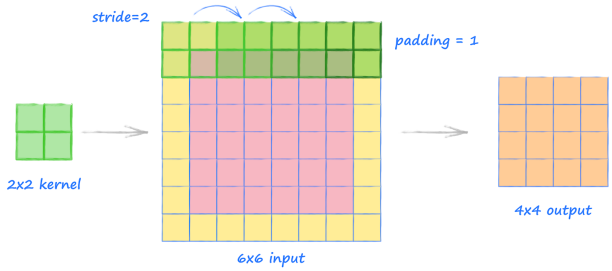


Figure 2: Stride and padding example. Taken from *Calculating the output size of convolutions and transpose convolutions* 2022.

Definition 2.19 (Max-pooling layer)**.** Maximum pooling operation reports the maximum output within a rectangular neighborhood (Goodfellow et al., 2016).

Definition 2.20 (Activation function)**.** An activation function is an element-wise non-linear function $f(\cdot)$. Some examples are:

$$f(x) = \frac{1}{1+e^{-x}} \quad \text{Sigmoid} \tag{21}$$

$$f(x) = max(0, x) \quad \text{Rectified linear unit (ReLU)} \tag{22}$$

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha * (e^x - 1), & \text{if } x \leq 0 \end{cases} \quad \text{ELU} \tag{23}$$

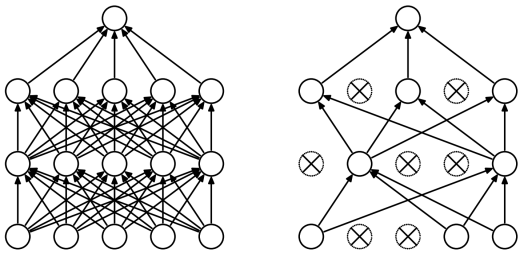
It is important to use activation functions after each convolutional or linear layer like RELU, ELU, Tahn, Sigmoid or any other non-linearities. Because any combination of linear functions can be represented with another linear function, having consecutive linear layers without non-linear function in the network is equivalent to having just one linear layer. Non-linearities In CNNs they are also often combined with max-pooling layers and dropouts to escape overfitting.

Definition 2.21 (Batch normalization layer)**.** Let's denote $B = \{x^{(i)}, ..., x^{(m)}\}$ to be a mini-batch of data. Then batch normalizing transform applied to this input data would be:

$$\begin{aligned} a^{(i)} &= \gamma \frac{x^{(i)} - \mu_B}{\sigma_B^2 + \epsilon} + \beta \\ \sigma_B^2 &= \frac{1}{m} \sum_i^m (x^{(i)} - \mu_B)^2 \\ \mu_B &= \frac{1}{m} \sum_i^m x^{(i)} \end{aligned} \tag{24}$$

where γ and β are learnable parameters, μ_B and σ_B^2 are the mean and standard deviation of the batch (Ioffe et al., 2015).

Definition 2.22 (Dropout layer)**.** Dropout is a technique that randomly sets some weights (units) to zero (Srivastava et al., 2014). It leads to the training of several smaller networks that share the parameters. If a mask vector μ specifies which units are included in training, then dropout's objective to be minimized becomes: $\mathbb{E}_\mu [J(\theta, \mu)]$. Visually dropout is presented in the Figure 3.



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

Figure 3: Dropout. Taken from Srivastava et al., 2014

The models in this project mostly use ELU activations as ELU provides a better signal flow between the layers by not cutting off the negative values completely.

Definition 2.23 (UNet)**.** UNet is fully convolutional neural network with U-shaped encoder-decoder network architecture (Ronneberger et al., 2015). Example of the UNet architecture can be found in Figure 7.

The encoder is a common CNN, consisting of the repeated block of two 3×3 convolutions, followed by an activation function, and a 2×2 max-pooling operation with stride 2. At each encoder step the number of feature channels doubles. The decoder is also a CNN, consisting of repeated blocks of transposed convolution, that halves the number of feature channels, followed by a concatenation with a corresponding output from an encoder, and two 3×3 convolutions, followed by a ReLU. The last decoder layer is a 1×1 convolution to map the tensor to the number of output image channels needed. Skip-connections is a very important part of UNet as they allow to the flow of high-resolution features from the encoder to the decoder that in turn allows to restore a corresponding high-resolution image.

Definition 2.24 (Autoencoder)**.** Autoencoder is an unsupervised learning technique in neural networks for the representation learning purposes. Autoencoder consists of an encoder that compresses data into a lower dimensional representation and a decoder that restores the original input from the encoded representation.

2.2.1.1 Regularization techniques

Regularization is mostly used to prevent a deep learning model to overfitting on the training data and to be able to generalize well. Overfitting has occured in the models used in this research and therefore it is improtant to understand the techniques that can be

used to prevent it. There are are several approaches to regularize the model and they will be explained below.

- Early-stopping

  Overfitting can be detected via visualizing train and validation losses. Training behaviour at first will be the usual one, meaning that both train and validation losses are gradually decreasing, however at some point the train loss continues to decrease, whereas the validation loss suddenly starts to increase. Since the model has not seen any of the data from the validation set, it means that it loses its ability to generalize on unseen data, while improving its perfomance on the seen data (train set). This does not happen during earlier epochs. Assuming that the model learns a complex decision surface while training, the weights of the model will be quite small and random with the correct weight initialization and therefore the best decision surface during the early epochs would be a smooth one. But during the later ones the difference in values of the weights grows and they become dissimilar which also means that the decision surface becomes more complex and the model is now able to fit not only the training data itself, but also its noise (Mitchell, 1997 p.111). And that is why stopping before the model becomes too complex, meaning to stop before the overfitting point, mitigates this problem.

- *L1- L2*-regularization

  The complexity of the deep model grows with the number of features it uses, sometimes the model may pay attention to the features that are not important to the outcome, or even considers noise to be a feature. To prevent this one should decrease the weights associated with useless features, however one cannot know ahead of time which of them should be ignored, therefore one may limit them all (Ying, 2019). In order to do that, a penalty term is added to the loss function:

  $$\tilde{L}(Y, M(X, \theta)) = L(Y, M(X, \theta)) + \lambda R(\theta) \tag{25}$$

  for some $\lambda > 0$. This is called a *soft-constraint* optimization. When $R(\theta)$ is of the form $R(\theta) = ||\theta||_2^2 = \sqrt{\sum_i \theta_i^2}$ this is called *L2*-regularization. When it is of form $R(\theta) = ||\theta||_1 = \sum_i |\theta_i|$ this is called *L1*-regularization. *L2*-regularization used in combination with backpropagation is equivalent to weight decay. Weight decay is defined by Hanson et al., 1988 as follows:

  $$\theta_{t+1} = (1 - \lambda)\theta_t - \alpha \frac{\partial L}{\partial \theta_t} \tag{26}$$

  where α is a learning rate. Weight decay successfully has more effect on the weights along which the gradient change is smaller Goodfellow et al., 2016. *L1*-regularization induces sparsity of the weights by assigning some of them to zero, this could also be considered as a feature selection approach.

- Regularization layers

  Batch normalization and dropout layers are also considered to be a form of regularization.

- Network reduction

  Since learning a too complex and noise-fitting decision surface might be a frequent cause of an overfit, another way to mitigate this would to be reduce the space of the possible decision surfaces and therefore make the surface simpler so that it cannot fit into the noise from the data. By changing the number of adaptive parameters in the network, the complexity can be varied (Bishop, 2006 p.332).

- Expansion of the training data

  For a successful training a model needs to have a sufficient amount of quality samples. An expanded dataset can improve the quality of the predictions Ying, 2019, however only when the model has already performed well on the initial dataset. If the model was performing badly initially, adding more data will not solve the problem.

2.2.2 Dimensionality reduction methods

This research also provides a study of the embeddings of a trained UNet and an autoencoder in Chapter 4.2. In order to better understand the visualizations, all dimensionality reduction methods that were used are listed and explained in this subsection.

Definition 2.25 (Embedding)**.** An embedding in this context is an output tensor from the encoder part of the UNet or from an encoder part of an autoencoder.

The encoder output of the UNet is a tensor of size $16 \times 16 \times 256$ and after its flattening it turns into a vector of dimensionality 655536. The smallest autoencoder embedding was of size 200 which is also high-dimensional. One of the tasks of this research is to determine whether there are any interesting patterns or grouping based on various criteria hidden within the bottleneck embeddings, and whether they could be useful for further research. Yet in order for humans to comprehend the embeddings we need to map them either to 2D or 3D vectors. In this context dimensionality reduction algorithms are essential.

Dimension reduction algorithms mostly form two main categories: ones are stronger preserving the pairwise distance globally — meaning they are trying to preserve the structure among all the data samples; others prefer to save local distances. For example, PCA (Hotelling, 1933) are assigned to the first caterogy, while t-SNE (Ulyanov, 2022) and Isomap are assinged to the latter one.

2.2.2.1 PCA

Principal component analysis (PCA) is an algorithm for linear dimensionality reduction (Pearson, 1901). PCA maximizes the variance in data's low-dimensional representation in order to keep as much information as possible. Essentially, PCA gives projections $\tilde{x^{(i)}}$ for input samples $x^{(i)}$ that would be very similar to them, however have a much smaller dimensionality. Eigenvectors of the data covariance matrix are the directions of the most variance within the data, and the eigenvalues correspoding to them are the amount of

variance hidden in each dimension. That is why by projecting the data using the eigenvectors with the largest eigenvalues, one will preserve the most varience of the data possible (Deisenroth et al., 2020).

The steps of PCA algorithm are the following (Deisenroth et al., 2020):

- Subtract mean μ_d. Centering the input data is not a necessary step, but it is recommended to do so to avoid numerical problems.
- Standardize the data. Calculate the standard deviation σ_d and standardize the data to have unit variance for every dimension.
- Do an eigendecomposition of the data covariance matrix. To do so one must first compute the covariance matrix itself, since the covariance matrix is symmetric from the spectral theorem one can always find an orthonormal basis of eigenvectors.
- Project the data. First, standardize the point $x^* \in \mathbb{R}$ using μ_d and σ_d:

$$x_d^* \leftarrow \frac{x_d^* - \mu_d}{\sigma_d} \tag{27}$$

  where $d = 1, ..., D$ and x_d^* is a d-th component of vector $x^* \in \mathbb{R}^D$ Get the projection as

$$\tilde{x^*} = BB^T x^* \tag{28}$$

  with coordinates $z^* = B^T x^*$. Here B is a matrix of eigenvectors associated with the biggest eigenvalues of a covariance matrix.

2.2.2.2 Uniform Manifold Approximation and Projection (UMAP)

UMAP (cite McInnes et al., 2018) was built in a way to preserve both and it is a competitor of the t-SNE approach. However, it is much faster and provides a transformation that can be used on the new data. UMAP is a graph-based algorithm and uses a k-nearest graph as its foundation. As with any graph-based algorithm, its structure also includes two main steps:

- Graph construction procedure. During this stage a weighted k-neighbour graph will be constructed from the data. Specific transformations are applied on its edges to surround local distance. And the strong asymmetry common to k-neighbour graphs will be reduced.
- Graph layout building. In this stage one first needs to define an objective function that can preserve the desired graph characteristics and then find a low dimensional representation of the graph that will minimize the objective.

In short, UMAP optimizes a low-dimensional graph from the high-dimensional one to be structurally very similar to each other. The algorithm has two important hyperparameters, which need to be chosen carefully: number of neighbors (*n_neighbors* in code) and minimum distance (*min_dist* in code). The first one balances the local versus the global

structure of the graphs; the higher the values the more fine details will be lost. The latter one controls how densely points will be located to one another. Higher values of this parameter result in a looser structure that preserves a broader topology of the data (*Understanding UMAP* n.d.).

2.2.2.3 PaCMAP

Pairwise Controlled Manifold Approximation (PaCMAP) is another dimensionality reduction method that is able to preserve both local and global data structure in a lower dimensional space. Unlike other methods that regulate the stronger preservance of global structure by using more neighbors, PaCMAP uses mid-near pairs to first capture global structure and then refine local structure, which both preserve global and local structure. It introduces the following parameters: neighbor pairs (*n_neighbors*), ratio of mid-near pairs to nearest neighbor pairs (*MN_ratio*), and ratio of further pairs to nearest neighbor pairs (*FP_ratio*) (Wang et al., 2021). The neighbor pairs parameter is used during the building of the k-nearest neighbor graph. It is recommended to use a value of around 10 for datasets with a size smaller than 10000 (Wang, n.d.). Configuring these parameters allows the user to achieve the desired ratio between preserving local and global structure. This method also has a faster runtime than UMAP, which allows to try out more hyperparameter options. For more details on how this methods works refer to Wang et al., 2021.

2.2.3 Clustering

After visualizing the embeddings, there is an interest in checking whether they form any kind of clusters. This question is discussed in Section 4.2.1 using the DBSCAN algorithm. This part will provide the theory needed to understand how this algorithm works and how it can be set up.

2.2.3.1 DBSCAN

The DBSCAN is a density-based unsupervised algorithm for discovering clusters. It considers regions with high-density of points to be clusters and points that are located far away from any cluster or form a low density region to be ourliers. This algorithm uses two hyperparameters = {*eps*, *min_samples*} to define the clusters. The first is the distance threshold, which is used to determine whether a point is located in the neighborhood of the other point. The latter one is the minimum number of points that are needed to form one cluster. It splits the points into four categories based on these hyperparameters: *core point* (*min_samples* points are reachable from it), *directly reachable point* (is within distance *eps* from any core point), *reachable* (there is a path from a core point to it via directly reachable points) and *outliers*. For more information on how clusters are formed refer to Ester et al., 1996. DBSCAN does not require the provision of the number of clusters in advance. Although this is a nice quality of this algorithm it is not that important for current research as the number of clusters that is needed here is known in advance. For example, this algorithm is used in Section 4.2 in order to check whether different phenotypes form different

clusters or, for example, whether corrupted images would fall into a separate clusters. In all cases the number of ground truth clusters is known in advance.

2.3 Drift detection basics

Assume that during training labeled data comes from a distribution p, meaning $\{(x^{(1)}, y^{(1)}), ..., (x^{(n)}, y^{(n)})\} \sim p$ and during deployment unlabeled data comes from a distribution q, meaning $\{x^{(1)\prime}, ..., x^{(1)\prime}\} \sim q$. The goal of the drift detection is to determine if $q(x\prime)$ is the same data distribution as $p(x)$. Or, putting it more formally, determine which hypothesis holds: null-hypothesis H_0 and an alternative hypothesis H_A, where $H_0 : p(x) = q(x)$ and $H_A : p(x) \neq q(x)$. Having samples from both distributions or representation of these samples in lower dimension, one can then choose a statistical hypothesis test to compare these distributions (Muandet et al., 2017).

2.3.1 Kernel methods and two-sample testing

The test used in this work for determinig whether two distributions are the same or not is one of the multivariate kernel two-sample tests and called maximum mean discrepancy or shotly MMD. The idea behid any two-sample testing is to choose two random samples, where each was taken from one of the two different distributions and afterwards to decide whether the difference in them is statistically significant.

MMD is a kernel-based method that can distinguish between two distributions based on their kernel mean embeddings in a reproducing kernel Hilbert space (RKHS) (Rabanser et al., 2018).

The idea behind a Hilbert space embedding distribution (or a kernel mean embedding) is to map a distribution into a point in a reproducing Hilbert space. After this step, one is allowed to use all powerful kernel methods for probability measures, resulting in methods like kernel two-sample testing. One of the widely known kernel methods is a support vector machines (SVM).

To understand why kernel mean embeddings are so successful one has to first understand what a kernel function is. With the help of kernel functions an inner product of elements $x, y \in \mathcal{X}$ in some high-dimensional feature space can be calculated. If the kernel function is positive definite, then there always exists a dot product space $\mathscr{H}$ along with a function that maps a space $\mathcal{X}$ into space $\mathscr{H}$: $\phi : \mathcal{X} \to \mathscr{H}$ such that $k(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathscr{H}}$ and most importantly there is no need for explicit computation of ϕ (Schölkopf et al., 2002). Therefore if there exists an algorithm that can be expressed through dot product of $\langle x, y \rangle$ then the kernel function can be applied to this dot product and this is called a *kernel trick* (Schölkopf et al., 2002).

Now, kernel mean embedding actually extends the above mentioned feature map ϕ to the space of probability distributions. In this space each probability distribution will be mapped to a mean function defined as follows:

$$\phi(\mathbb{P}) = \mu_{\mathbb{P}} := \int_{\mathcal{X}} k(x, \cdot) d\mathbb{P}(x) \tag{29}$$

Here $k(x, \cdot)$ is a positive definite symmetric kernel function. The main goal here is to map a distribution $\mathbb{P}$ to a point in the feature space $\mathscr{H}$ and this feature space is exactly an RKHS that corresponds to a kernel k. Such a mapping might be useful because it captures all information about the initial distribution $\mathbb{P}$. This mapping $\mathbb{P} \to \mu_{\mathbb{P}}$ is injective. This means that $||\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}||_{\mathscr{H}} = 0$ if and only if $\mathbb{P} = \mathbb{Q}$. Here it means that $\mathbb{P}$ and $\mathbb{Q}$ is the same distribution. Additionally, since the mapping is injective, it is possible to use such characterization of a distribution to be used in two-sample homogeneity tests, which is exactly what is needed here.

To estimate a kernel mean embedding is much easier than to estimate a distribution itself. This approach is successfully used in data-generating processes, it also improves some statistical inference methods like two-sample testing. Such an approach is also useful when instead of data points in testing and training dataset, there are probability distributions.

Inner product $\langle x, y \rangle$ can be viewed as a similarity measure between x and y. This inner product includes a class of linear functions and this class is too restrictive for many applications, however there is a simple possible extension to add non-linearities to it with the mapping:

$$\phi : \mathcal{X} \to \mathcal{F} \tag{30}$$

where

$$\phi : x \to \phi(x) \tag{31}$$

Here $\mathcal{F}$ is high-dimensional feature space and it is possible to evaluate then:

$$k(x, y) := \langle \phi(x), \phi(y) \rangle_{\mathcal{F}} \tag{32}$$

with $\langle \cdot, \cdot \rangle_{\mathcal{F}}$ an inner product of $\mathcal{F}$.

Now $k(x, y)$ is already a non-linear similarity measure between x and y. In order to get a non-linear version of the algorithms that use dot product simply substitute $\langle x, y \rangle$ with $\langle \phi(x), \phi(y) \rangle_{\mathcal{F}}$.

Let's define the following mapping that represents in $\mathcal{X}$ any probability measure $\mathbb{P}$ and denote it as $\mu_{\mathbb{P}}$. This mapping is called a kernel mean embedding.

Definition 2.26 (Kernel mean embedding (Berlinet et al., 2004))**.** The kernel mean embedding of probability measure in $M_+^1(\mathcal{X})$ into RKHS $\mathscr{H}$ endowed with a reproducing kernel $k : \mathscr{H} \times \mathscr{H} \to \mathbb{R}$ is defined by a mapping

$$\mu : M_+^1(\mathcal{X}) \to \mathscr{H}, \mathbb{P} \to \int k(x, \cdot) d\mathbb{P}(x) \tag{33}$$

However, usually there is no access to the distribution $\mathbb{P}$ and that is why one cannot directly compute $\mu_{\mathbb{P}}$. Fortunately, there are samples that can be drawn from this distribution and with their use one can make a good approximation of $\hat{\mu}_{\mathbb{P}}$ of a true kernel mean

embedding $\mu_{\mathbb{P}}$. One such approximation could be the following unbiased estimate:

$$\hat{\mu}_{\mathbb{P}} := \frac{1}{n}\sum_{i=1}^{n} k(x_i, \cdot) \tag{34}$$

Moreover, this estimator $\hat{\mu}_{\mathbb{P}}$ will converge to $\mu_{\mathbb{P}}$ by the law of large numbers as $n \to \infty$.

Definition 2.27 (Characteristic kernel)**.** A kernel k is a characteristic kernel if the map $\mu : \mathbb{P} \to \mu_{\mathbb{P}}$ is injective. If the reproducing kernel of the RKHS $\mathscr{H}$ is characteristic, then RKHS is called characteristic as well.

In machine learning applications and statistics kernel mean embedding is as a metric for the probability distributions. And mean embeddings metric is actually just a specific case of a more general so-called integral probability metric (IPM) (Müller, 1997).

Definition 2.28 (IPM)**.** Let $\mathbb{P}$ and $\mathbb{Q}$ be two probability measures on some measurable space $\mathcal{X}$. Then IPM is defined as follows:

$$\gamma[\mathcal{F}, \mathbb{P}, \mathbb{Q}] = \sup_{f \in \mathcal{F}} \left\{ \int f(x) d\mathbb{P}(x) - \int f(y) d\mathbb{Q}(y) \right\} \tag{35}$$

with $\mathcal{F}$ being a space of real-value bounded functions.

2.3.2 Maximum mean discrepancy for drift detection

Let's assume that $\mathcal{F} := \{f \mid \|f\|_{\mathscr{H}} \leq 1\}$, this means that if supremum in Definition 35 is taken over functions in the unit ball in RKHS, then mean embedding metric is called *maximum mean discrepancy* (MMD).

Definition 2.29 (IPM)**.** Maximum mean discrepancy is defined as a distance between two mean embeddings of distributions:

$$\begin{aligned} \mathrm{MMD}[\mathscr{H}, \mathbb{P}, \mathbb{Q}] &= \sup_{\|f\| \leq 1} \left\{ \int f(x) d\mathbb{P}(x) - \int f(y) d\mathbb{Q}(y) \right\} \\ &= \sup_{\|f\| \leq 1} \{ \langle f, \mu_{\mathbb{P}} - \mu_{\mathbb{Q}} \rangle_{\mathscr{H}} \} \\ &= \|\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}\|_{\mathscr{H}} \end{aligned} \tag{36}$$

From 36: if $\mathscr{H}$ is characteristic, then $\mathrm{MMD}[\mathscr{H}, \mathbb{P}, \mathbb{Q}] = 0$ if and only if $\mathbb{P} = \mathbb{Q}$.

Similarly to kernel mean embedding estimation, through samples drawn from a distribution a biased empirical estimator of MMD can be obtained. Assume that there are two sets of samples $X = \{x^{(1)}, ..., x^{(n)}\}$ and $Y = \{y^{(1)}, ..., y^{(m)}\}$ drawn from two distributions $\mathbb{P}$ and $\mathbb{Q}$ corrspondingly, then the biased estimate of MMD can be calculated as follows:

$$\widehat{\mathrm{MMD}}_b^2[\mathscr{H}, X, Y] := \sup_{\|f\|_{\mathscr{H}} \leq 1} \left\{ \frac{1}{n}\sum_{i=1}^{n} f(x^{(i)}) - \frac{1}{m}\sum_{j=1}^{m} f(y^{(j)}) \right\} \tag{37}$$

By replacing $\frac{1}{n}\sum_{i=1}^{n} f(x^{(i)})$ with the empirical estimators of mean embeddings one would get:

$$\widehat{\mathrm{MMD}}_b^2[\mathscr{H}, X, Y] = \|\widehat{\mu}_{\mathbb{P}} - \widehat{\mu}_{\mathbb{Q}}\|^2_{\mathscr{H}} \tag{38}$$

If an unbiased estimator of MMD through the kernel function k is needed then one could use the following estimator from Corollary 2.3 in Borgwardt et al., 2006.

The most common application of MMD in statistics is two-sample testing. Particularly in testing the null hypothesis $H_0 : \|\widehat{\mu}_{\mathbb{P}} - \widehat{\mu}_{\mathbb{Q}}\|_{\mathscr{H}} = 0$ that two samples come from the same distribution against an alternative hypothesis $H_1 : \|\widehat{\mu}_{\mathbb{P}} - \widehat{\mu}_{\mathbb{Q}}\|_{\mathscr{H}} \neq 0$. But one has to be cautious here, even when both samples came from the same distribution it still might be that the MMD will not be zero due to the fact that that this is an estimate and not a precise value and we have a finite number of samples for this estimate.

$$\begin{aligned}\widehat{\mathrm{MMD}}_b^2[\mathscr{H}, X, Y] =& \frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j\neq i}^{n} k(x_i, x_j) \\ &+ \frac{1}{m(m-1)}\sum_{i=1}^{m}\sum_{j\neq i}^{n} k(y_i, y_j) \\ &- \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} k(x_i, y_j)\end{aligned} \tag{39}$$

2.4 Imaging

2.4.1 Digital imaging

Digitally an image is represented as an array of size (H, W, C) where H is the height, W is the width and C is the number of channels of the image. In this work, $C = 1$ and $W = H$. A digital image A can be represented with the matrix:

$$A = \begin{bmatrix} a_{0,0} & \cdots & a_{0,W-1} \\ \vdots & \ddots & \vdots \\ a_{H,0} & \cdots & a_{H-1,W-1} \end{bmatrix} \tag{40}$$

where $a_{i,j} \in \mathbb{R}$. Both DIC and fluorescence images were provided in tag image file format (TIFF). For the processing convenience purpose all images were normalized to be in the range of $[0, 1]$:

$$a_{i,j}^{\text{norm}} = \frac{a_{i,j} - \min(A)}{\max(A) - \min(A)} \tag{41}$$

for $\forall i \in \{0, ..., W-1\}$ and $\forall j \in \{0, ..., H-1\}$

2.4.2 Microscopy imaging

2.4.2.1 Image acquisition process details

The cells used in this research are growing in 96-well plates. A plate or a microplate in biology is a flat plate with multiple wells. The microscope used in the experiments takes several photos of each well plate in random locations within it. The reason for that lies in the focusing settings of a microscope. To get a reasonably good, and not blurry, photo, a microscope has to focus on a specific location of the plate. In the microscope used for this research the choice of this location happens automatically. It is not possible to choose which region of a well-plate will be photographed, therefore it is also not possible to do consecutive images without intersections as the regions are chosen randomly (see Figure 4).

It might be problematic in the following sense: photos taken in such a manner do not guarantee that the focus will land in distinct spots all the time. Meaning that some cells present in one of the photos might appear in the other ones as well. Since the photos are high-resolution they will first be split into crops of size 256×256 each during the preprocessing. It might happen that the same cells appear in several crops. That is why after the split of the image data between train, test and validation sets it might also happen that the same set of cells will once land in the train set and another time in the validation set, which will lead to a not completely fair and representative validaiton metrics during training.

In order to overcome this problem much more expensive equipment is needed. Since it does not cause fundamental problems in this case, except for the fact that the validation metrics might be lower than what they should have been, we neglect this effect for the remainder of the work.

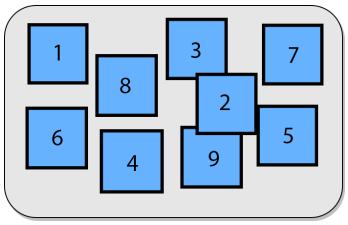


Figure 4: Radom location of the microscope focus for one well-plate

2.4.3 Local and global thresholding for image segmentation

There are in general two types of thresholding that exist to binarize an image (create its mask): global and local.

Global thresholding is a an algorithm that simply choses one threshold T for the whole histogram of the image. All pixels that are smaller than this threshold $x_{i,j} < T$ are assigned to be of class 0 (background) and all pixels that are larger than this threshold $x_{i,j} > T$ are assigned to be of class 1 (foreground). To find a good threshold automatically (at least to

some extect) Gonzalez et al., 2006 proposed the following algorithm:

Algorithm 3 Global thresholding

1. Select an initial estimate for T.
2. Segment the image using T. This will produce 2 groups of pixels G_1 (all pixels $x_i > T$) and G_2 (all pixels $x_i < T$).
3. Computer the average gray values μ_1 and μ_2 for the pixels in regions G_1 and G_2.
4. Compute a new threshold value $T' = \frac{\mu_1 + \mu_2}{2}$
5. Repeat steps 2-4 until difference in the change of value T is smaller that a predefined parameter.

Nevertheless, it is not obvious how to preselect an initial threshold in step 1. There are several options here, however keep in mind that there is also no single best solution among them. For example, when an assumption that the foreground occupies approximately the same area as the background holds, than initial threshold T should be chosen to be an average gray level and etc.

It is implemented in *skimage.filters* and according to the documentation (Prewitt et al., 1965) works in the following way: it assumes that the histogram $p = (p_0, \ldots, p_M)$ of the image is bimodal, meaning that it has two clearly defined peaks (background and foreground). Afetrwards the histogram is iteratively smoothed using a running average of size $k = 3$. The points on the histogram are updated with the value a_k from Equation 42 until only 2 local maximas (a_l and a_r) are left.

$$a_k = \frac{1}{k} \sum_{i=n-k+1}^{n} p_i \tag{42}$$

Then the threshold is taken as the minimum between the two local maximas:

$$a_l \leq T \leq a_r \tag{43}$$

One clear downside of this approach though is that images which histograms have very unequal peaks or a broad and flat valley will be unsuitable for this method (*Thresholding* 2022).

However, there is another better approach that performs better in such conditions: **local thresholding**. It is implemented in *skimage.filters* and can be used in the following way:

```
skimage.filters.local_threshold(img, block_size=7,
    method='gaussian', offset=0)
```

The main idea behind it is the following: instead of selecting one threshold for the whole image (global one), one can select several thresholds for each local region with a predefined size. The comparison between global and local thresholding is presented in Figure 24, where (a), (b) denote exteme corruption cases and (c) represents a normal illumination. "The threshold value is the weighted mean for the local neighborhood of a pixel subtracted by a constant" (Gonzalez et al., 2006). A stricter mathematical explanation of this algorithm is presented below.

Let z be a random variable that quantifies a gray-level value of the pixel, then the histogram of the image is a probability density function (PDF) $p(z)$. Since we assume that the image contains a background and a foreground, then this PDF is a mixture of two densities $p_1(z)$ and $p_2(z)$ weighted by the relative areas of these two classes (their number of pixels) P_1 and P_2. Then

$$p(z) = P_1 p_1(z) + P_2 p_2(z) \tag{44}$$

By assuming Gauassian model for both $p_1(z)$ and $p_2(z)$, one gets a Gaussian Mixture Model (GMM). Since we have assumed that each pixel can be asssigned to either a background of foreground only, $P_1 + P_2 = 1$ must hold.

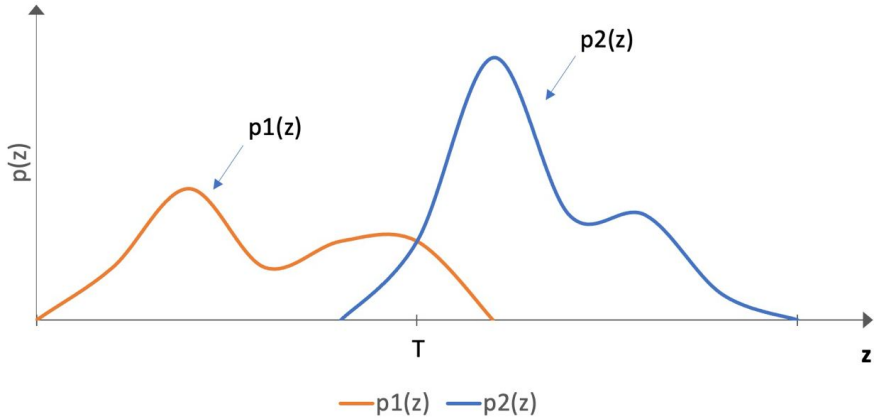


Figure 5: Histogram as a probability density function. Taken from Gonzalez et al., 2006

Probability to falsely classify an background pixel as a foreground then is:

$$E_1(T) = \int_{-\infty}^{T} p_2(z)\, dz \tag{45}$$

And probability to falsely classify a foreground pixel as a background then is:

$$E_2(T) = \int_{T}^{+\infty} p_1(z)\, dz \tag{46}$$

The overall error is:

$$E(T) = P_1 E_1(T) + P_2 E_2(T) \tag{47}$$

Under the assumptions that $p_1(T)$ and $p_2(T)$ are Gaussian distributions, Gonzalez et al., 2006 show that the optimal solution is:

$$T = \frac{\mu_1 + \mu_2}{2} + \frac{\sigma^2}{\mu_1 - \mu_2} ln\left(\frac{P_2}{P_1}\right) \tag{48}$$

Such threshold search is then applied to all of the subregions of the image with overlaps. Thresholds are calculated only for the regions that contain two clear peaks in their histograms and interpolated to the other pixels from the regions that do not contain them. If the subregions does not contain two peaks, it simply means that there is no foreground or background object on it.

2.4.4 Background removal algorithm

Presence of presence of non-specific fluorescence lighting, which is the fluorescence lightning produced by other cell parts that were not target in the staining procedures, can be troublesome for organelles like Golgi (see Figure 35). In order to mitigate this proble background removal algorithm described in this section can be used.

The rolling ball algorithm was introduced by Sternberg, 1983 and is still widely used for processing medical and biological data. The idea of this algorithm is based on morphological opening of the image.

Definition 2.30 (Morphological opening)**.** Morphological opening is an operation in image processing, when an image is first eroded and then dilated using the same structuring element.

Morphological opening is helpful for removing noisy elements, thin lines, while preserving bigger objects in the image (*Types of morphological operations* 2022).

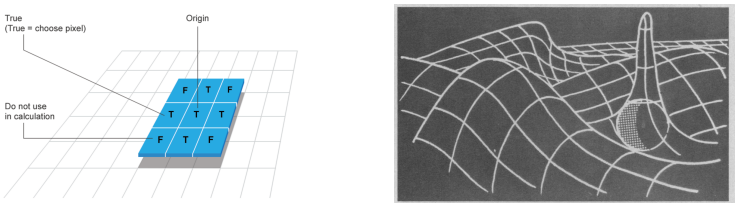
A structuring element is an analog of a kernel in image processing. It is a matrix of zeros and ones (true and false), where ones represent the elements that will be used to perform the morphological operation and others will be ignored (see an example in Figure 6 [left]). Such a structuring element is applied across the whole input image producing a new image based on the rules of a morphological operation it performs.

For example, morphological dilation takes a new value of the pixel as the maximum value of its neighbors within the structuring element. Therefore, after this operation, the lines will be thicker and in general objects will appear bigger.

Whereas morphological erosion turns the pixel value into the minimum value of its neighbors within the structuring element. After this operation the floating pixels will be removed and all objects become smaller and thinner.

Sternberg, 1983 has extrapolated the operation of morphological opening from 2D into 3D space. He defines a new interpretation of a 2D image in a 3D world called umbra. Umbra can be described as a 3D plane, where the height of each point is determined by its intesity value.

The structuring element for morphological opening of an umbra has to be then also a 3D object — in this case a ball. Morphological opening of an umbra is a union of translations of the 3D structuring element that can be entirely contained inside it (see Figure 6). One can imagine the ball freely moving inside the volume constrained by the upper surface of an umbra. The opening then consists of all the pixels that can be reached by the ball. The radius of the ball is a hyper-parameter which has to be tuned.



(a) Structuring element (b) Rolling ball

Figure 6: Visualization of a 2D and a 3D structuring element of a rolling ball algorithm.

3 Implementation and experiments

In this chapter the results of all experiments performed for predicting fluorescence signal from four cell organelles: nuclei, endoplasmic reticulum, Golgi apparatus, and full cell fluorescence are provided and discussed. This part first starts out with a description of the models and data used in the experiments, followed by four subsections dedicated to each of the organelles. Each subsection describes its own different approaches in pre- or postprocessing needed, difficulties that occurred during preparation or training steps as well as the results obtained for each organelle separately.

3.1 Setup procedures

3.1.1 Neural network architecture

As was described in section 2.2 the choice of the model architecture fell onto the UNet structure. Here a detailed description of the architecture and the different layers used will be provided. An architecture used in this research follows LaChance et al., 2020 work. The input to this network is a 256×256 pixel DIC image that should already be preprocessed based on the corresponding preprocessing pipeline to the desired organelle. Specifications about different preprocessings are described separately in the next subsections dedicated to different organelles.

The encoder part of the UNet (Figure 7) compresses the spatial dimensions of the image step by step (the spatial dimension size is denoted by a number on the left of each green block) into tensors or so-called feature maps with an increasing amount of filters (the number of filters is denoted on top of each green block). This allows to reduce the spatial information in the image and capture semantics. The decoder part on the contrary decompresses feature maps gradually increasing the amount of spatial information in tensors and reducing the number of filters. All convolutional layers use convolutions of size 3×3 with the corresponding number of filters. Downsampling in encoder reduces the spatial dimension twice during each step and is implemented using max-pooling with a size of 2×2. Upsampling in decoder increases the spatial dimension also twice during each step and is implemented using transposed convolutions with a size of 2×2.

After the first convolution that follows each max-pooling step a batch normalization layer was used as it is well-known for speeding up the training process (Ioffe et al., 2015). It should not be forgotten however that the use of batch normalization might sometimes be dangerous due to the hidden information leaks that are being induced (Abe Fetterman, 2020). Additionally, dropouts were used for example for actin predictions as the model would encounter overfit quite easily there, however in the default architecture dropouts are not present. That is another thing that differentiates this architecture from the original one in LaChance et al., 2020 work. The last layer of the UNet here is a sigmoid activation function.

There is a space for potential improvements regarding the model architecture used in

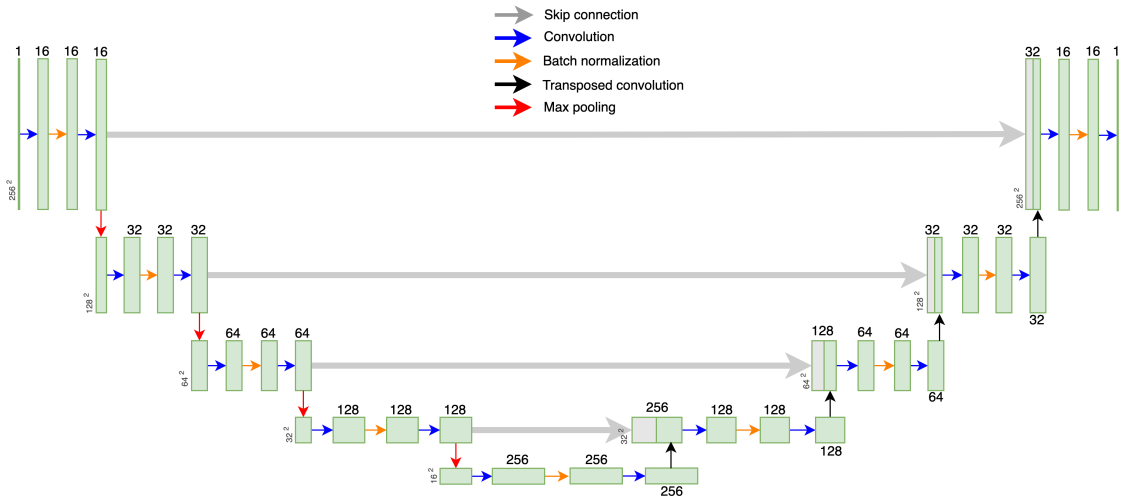


Figure 7: UNet architecture used in this research. All layers are represented with arrows of corresponding colors, tensor's spatial dimension is specified on the left and number of channels is specified on top of each tensor.

this research: for example, Cheng et al., 2021 recommend to use special dense-blocks after each convolutional layer. They consist of another 3 convolutional layers with 24 filters, batch normalization layer and ReLU activations each. This could potentially facilitate efficient training of the model, still most probably the efficience comes mostly from batch normalization layers that are already used in our architecture. Nevertheless the idea of using a bigger model such as one in Cheng et al., 2021, or more specifically a model with more filters, indeed improves the predictions (shown in Section 3.2.2.2). That that leaves room for further research and improvements regarding the size of the model and the additional use of dense-blocks.

An interesting question that arises here automatically is: what do UNet embeddings (output tensors from the encoder) represent? There is a big difference between any representation learning network such as an autoencoder and a UNet — a UNet model uses skip-connections that allow to propagate information between its encoder and a decoder. This means that embeddings do not contain purely semantic information, because essentially the network is not pushed towards compressing the information severely (as an autoencoder would do), but it should only extract information relevant for segmentation features. One of the questions solved in this work was whether or not UNet embeddings are clustering based on the following classes: cell phenotypes, any kind of corruption within the data. For example, it would be very useful to be able to not only predict the data itself, but also to say whether the prediction is reliable or not. The initial hypothesis here stats that if the predictions are not of a good enough quality this would also be reflected in the embeddings.

3.1.2 Available data

There were four targets studied in this research: nuclei, endoplasmic reticulum, Golgi apparatus and a full cell target. The initial three targets provide many insights on the state of cells based on their quantity, size and the intensity of fluorescence signal that they can produce by binding with the protein, the latter is useful in general for locating cells and calculating their area. There are of course other possible targets that could be used during the selection step of CLD, but the goal of this particular study was to provide the proof of concept on whether the network could potentially substitute manual fluorescence staining in general. The hypothesis on which we rely here is that if the model can successfully predict the fluorescence signal from some of the targets, then this research could be extended to other cell organelles as well.

The imaging data always comes in pairs (DIC + fluorescence imaging) and it splitted into several *datasets*, where each dataset corresponds to one 96-well plate. There are only several datasets for each of the organelles and each of them contains 100 images of random locations within this plate. Almost all plates contain pairs of images from one fluorescent staining target only. More details about the used folder structure can be found in Appendix A.

There were three different phenotypes present across the datasets: CHOZN, PHX and H19. The latter, however is present only for a full cell target. "Cellular phenotype is the conglomerate of multiple cellular processes involving gene and protein expression that result in the elaboration of a cell's particular morphology and function" Sul et al., 2009.

	Total images	Training crops	Validation crops	Test crops
Nuclei	595	27,264	3,008	7,616
Actin	400	18,432	2,048	5,120
Golgi	761	23,036	2,336	6,347
GFP	400	18,432	2,048	5,120

Table 1: Available data for each of the organelles

For training, all images with original size of 2136×2136 are cropped into smaller crops of size 256×256 and split between training, test and validation sets (70%, 20%, 10% accordingly) with crops from one image being present only in a single set. However, it is still possible that the cells themselves might be mixed up between the sets. This is because of the randomness of the microscopy focusing system (see subsection 3.2.3). The summary of the total number of crops in each set is presened in Table 1.

3.1.3 Augmentations

Augmentations are a powerful regularization technique that helps the network to generalize better (Perez et al., 2017) and is an effective solution for a situation with the lack of labeled data (Yang et al., 2022). The main idea behind augmentation is to increase the diversity of training data, when the acquisition of the real training data is expensive. Aug-

menting existing images creates the new synthetic samples which are hypothetically very close to the original true distribution of images. However, one should be very cautious regarding the type of augmentations that can be used. Augmentation must enrich the dataset, but it should not change the semantics hidden in each image. The peculiarity of current research is the pair-wise correspondence between the DIC input and the output fluorescence. One should not change the input in such a way that the fluorescence signal could not be inferred from it. Therefore the following augmentations have been used on cropped images:

- **Flipping**: Flip the crop horizontally (30% chance), vertically (30% chance) or both.
- **Random rotation**: Rotates the crop by a random angle.
- **Random scaling**: Reduces crop size by 100 (20% chance), 50 (20% chance) or 20 (60% chance) pixels from each side (down, top, left, right).
- **Contrast**: Halves an image's constrast (50% chance), or triples it (50% chance). Applied with a 20% chance.
- **Defocus blur**: Imitates a defocus blur of severity level 4 (see section 4.1.2) on an image with a 20% chance.

3.1.3.1 Special augmentations for rotation and scaling

Augmentations are chosen to be applied during training and the reason for that is to preserve the same conditions regarding the size of the datasets in order to have a fair comparison between approaches. For example, one decides to augment 20% of images and add them to an original dataset in order to expand it. Then the comparison between training with and without augmentations would be unfair as the sizes of the datasets are different, and it is not clear whether the performance improvement or decrease comes from the longer training (enlarged dataset) or augmentations.

Since augmentations are applied on crops directly during training there is a possibility to improve the rotation and scale of crops. The problem with these augmentations are depicted in Figure 8 — after applying them the a gray background will appear which would be filled with zeros in PyToch implementation. However, since an original image where the crop has been cut out is available, one can easily restore the background with the original values and avoid this problem entirely.

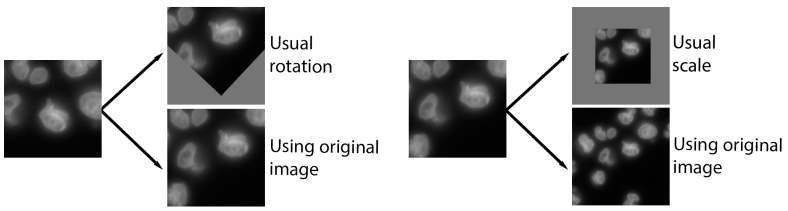


Figure 8: Improving augmentation by using original image for rotation and scaling

Applying augmentations in combination with other regularization techniques has proven to be helpful for nuclei predictions and against overfitting for ER training (see Figure 16, see Figure 28). Therefore it is recommended to use it in further research as well.

3.1.4 Model setup

In order to achieve best predictions results it is very important to pre-setup a model correctly. Since the architecture used here is very similar to the one used in LaChance paper, the setup configuration is similar as well.

3.1.4.1 Weight initialization

He initialization method was suggested by the LaChance et al., 2020 paper and has been used in this reseach as well. In order to show the importance of weight initialization choice two experiments have been conducted: in the first on the model predicting nuclei target was trained with the default weight initialization provided by PyTorch and then the initialization was switched to a true He initialization.

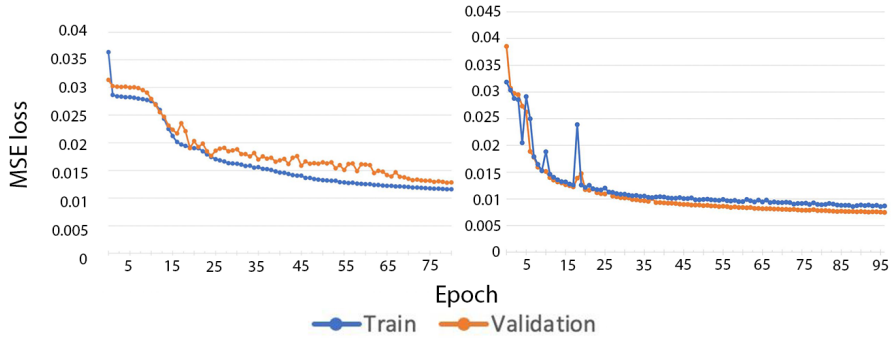


Figure 9: Nuclei training without (left) and with (right) custom weight initialization. The stagnation during first few epochs on the left signalizes about the wrong initialization of model's weights. As a result the left model also converges to a higher value.

The results of the experiments are presented in Figure 9. As can be seen, the loss in the left plots stagnates during the first few epochs and then begins to converge later. This is a symptom of a wrong initialization of the weights. Even after the convergence begins, the model still has a higher loss than the one on the right (around 0.012 in comparison to 0.007). However, the model on the right is not completely perfect as the loss still does not converge at the same speed everywhere. Although this might be not related to the weight initialization but more the to instability of the training in general as there were only few images used for these experiments.

3.1.4.2 Regularization

All regularization techniques apart from the network reduction mentioned in Section 2.2.1.1 were used in this work. For example, in ER training (see Figure 28) one can clearly observe overfitting, that was mitigated with the use of weight decay and augmentations. Batch normalization layers were added to the UNet architecture not only for regularization purposes, but for an imroved speed of learning as well. Network reduction was not used in this case as it was proven that bigger model captures finer details in a better way (see Figure 18). Finally, expansion of the data has a strong impact on traning: having $27,264$ crops of data the model was trained on $5,376$ crops only (two 96-well plates) to find the best structure and regularization first, afterwards the model was retrained using more data and the PCC loss improves from 0.77, to 0.93.

3.1.4.3 Optimizers

It is also important to choose a correct optimizer in order for a model to converge as fast as possible. Here, three different optimizers have been tried out — namely, SGD, Adam and Adadelta. One can evaluate the quiality of an optimization technique based on the speed of convergence and a value towards which the training has converged. As can be seen in Figure 10 the SGD optimizer performed the worst, while Adam and Adadelta optimizer performed similarly with Adadelta converging to slightly better values in the end. SGD optimizer converges both slower and towards a higher loss, whereas SGD converges slightly slower than Adam, however to a lower loss. One can see in Figure 10 that Adam optimizer has required some fine-tuning of the learning rate from 0.001 to 0.0001 to achieve the best result. Both Adadelta and Adam can be used for model optimization in this dataset, however adadelta optimizer was chosen as it has converged to a lower values of loss. The experiments were conducted on the truncated dataset of nuclei images using PCC loss.

3.1.5 Model evaluation: metrics for downstream tasks

As LaChance et al., 2020 notice evaluation of models predicting flureoscence signal in terms of the PCC or MSE loss is not quite an objective approach. Even when one model has a smaller loss, it will not nessecarily perform better than another model. There are other more practical every-day metrics for this task, that have more value to the end-user rather than some abstract performance measurements typically used in computer vision. Therefore the evaluation of the model in this research is additionally checked in terms of the following biologically relevant metrics:

- Number of nuclei / ER / Golgi / cells
- Area of nucleus / ER / Golgi / cell
- Total intensity of nuclei / ER / Golgi
- Mean intensity of nuclei / ER / Golgi

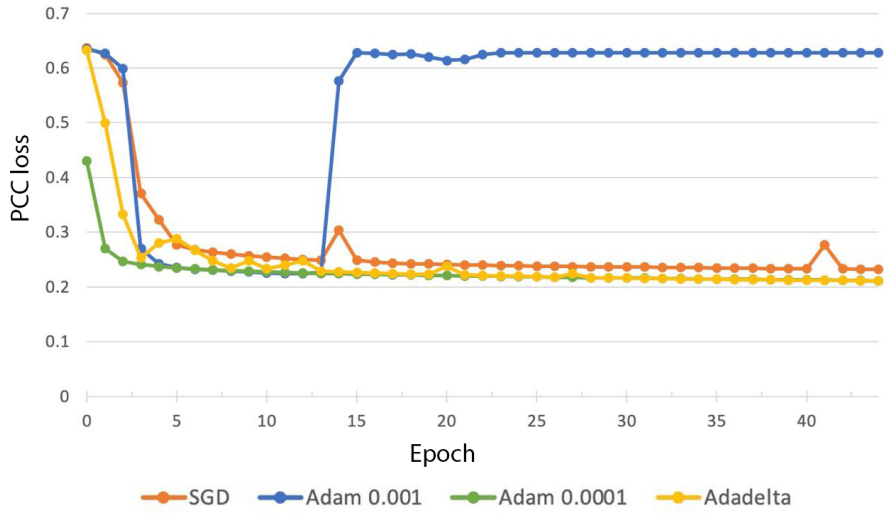


Figure 10: Comparison of convergence for different optimizers

These quantities can be compareed with each other in a much more understandable way. On the contrary, when one recieves a precision value P (let it be PCC loss in this example) from the model performance evaluation there is no way to appreciate how good this model is. Usually the value of P can easily be increased by simply training on more data or using a better resolution microscopy, but this would incur high expenses.

Highly practical metrics mentioned above were calculated for each image in the test set and for corresponding ground truth images. This produces two distributions: one is a distribution of predicted values and another is a distribution of ground truth values. Ideally, both of them should be the same distribution. In order to compare these distributions in this work three approaches are used. First, is to visually compare the violin plots by simply checking the form and range of the two. Second, to visualize a scatter plot, where two axes have values of a metric for each image from both distributions. And third, to compare them quantitevely by correlation coefficients, or more specifically, with the help of Spearman rank coefficient and Pearson correlation coefficient. Such comparisons are performed for each of the four metrics and for each organelle.

In each subsection dedicated to each organelle these metrics are presented under the according "Biological metrics" section.

3.2 Nuclei

A nucleus (plural nuclei), as related to genomics, is the membrane-enclosed organelle within a cell that contains the chromosomes. The nucleus is one of the easiest organelles to detect within the cell as it is usually located in the middle and occupies a large area of the cell (Pathak et al., 2021, see Figure 11). The nucleus contains all of the cell's chromosomes, which in turn encode the genetic material, therefore the nucleus is an essential organelle (*Nucleus* 2022). In order to stain it, DAPI was added to the cells. This is a fluorescent stain that binds strongly with some regions in DNA. DNA is strongly concentrated and near-uniformly distributed in the nucleus, hence DAPI staining identifies the nucleus (Tarnowski et al., 1991). Analysis of cell's nucleus can provide many valuable insights, for example the radius of living cells is on average bigger than in the dead ones (Christiansen et al., 2018). With fluorescence labeling one can derive some useful features used for determining whether the well plate should or should not be selected during the selection step in the CLD process.

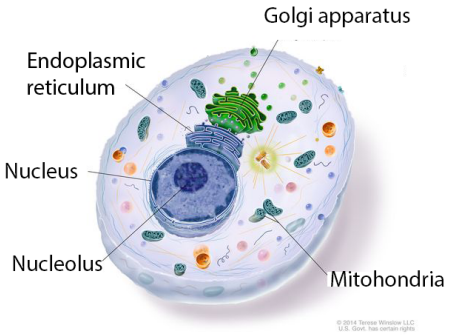


Figure 11: Cell structure. Among all organelles presented in this Figure only nucleus, endoplasmic reticulum and golgi apparatus were used and prediction targets. However, this research is easility extendable to other organelles as well. Taken from *NCI Dictionary of Cancer terms* 2014

3.2.1 Preprocessing

As not all of the dataset samples were of the same quality it was important to first manually filter the dataset to remove the bad imaging samples. Even with the normal conditions many of the images contain a lot of background. This creates a background vs. foreground class imbalance (see Figure 12a). Overexposure is also a typical problem that creates samples of a too high intensity and with details inside the nucleus missing (see Figure 12b, c). Lastly, underexposure is just as problematic as overexposure (see Figure 12d).

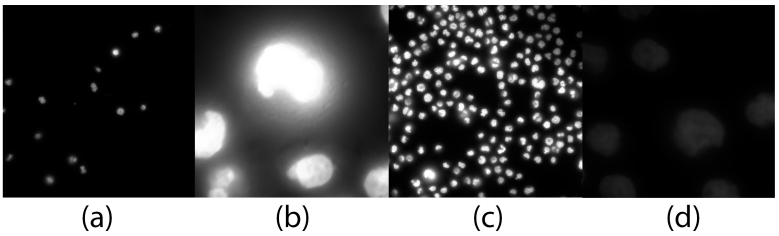


Figure 12: Nuclei fluorescence samples to be filtered out. (a) — too few cells in the image result in many fully black crops; (b) — overexposure; (c) — overexposure, lack of details; (d) — underexposure.

Once the images have been filtered out, they were normalized to have the values between 0 and 1.

3.2.2 Training and predictions

3.2.2.1 Convergence

As the nuclei dataset is one of the biggest ones (see Table 1), the experiments were first performed on the subset of the data, and only once the training pipeline was established, the training was performed using the full dataset. Figure 9 (right) represents the training of nuclei using only two 96-well plates with MSE loss. The training is very unstable in the begining, but one can clearly see that the model successfully converges afterwards. Our hypothesis behind the instability was the clear lack of training data, which was proven by further training using the full data and as a result achieving a more stable PCC loss (see Figure 13).

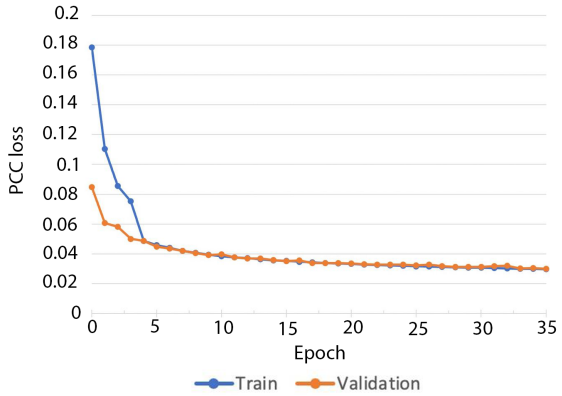


Figure 13: Having more data makes training more stable

As mentioned in Definition 2.9 for correct understanding of the following plots, one should be careful with differentiating PCC loss from PCC itself. PCC loss converts PCC to be between 0 and 1, with 0 being an optimal value.

Seeing that the model significantly stabilizes with the use of more data and that the prediction results become much more similar to the ground truth (see Figure 16 *small dataset* vs. *full dataset*), it was decided to try the use of augmentations in order to enlarge the dataset even more. In this case the augmentations were not used in order to regularize or stabilize training (by providing more difficult, for instance, blurred samples), but to simply have more data. Training and validation PCC losses from the training with the use of augmented data are presented in Figure 14. The augmentations used here are horizontal and vertical flips, rotations and crops (described in detail in section 3.1.3).

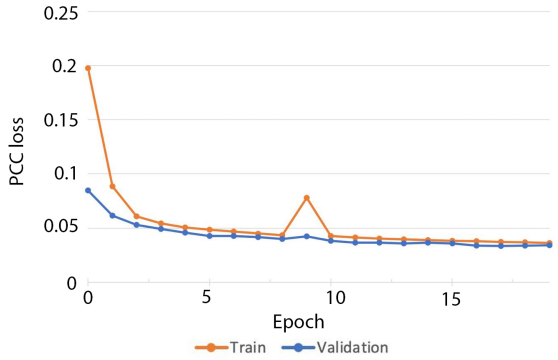


Figure 14: Adding simple augmentations in the dataset

Validation PCC loss has slightly increased to 0.0381 in comparison to the previous value of 0.0365. However, the validation set on which this loss was estimated also includes augmentations mentioned above and therefore presents a slightly more difficult task than the original validation set. True improvement is confirmed by measuring the loss of the models on the same dataset, where PCC loss has improved from 0.0365 to 0.0322 (or PCC from 0.92 to 0.93).

In the next experiment the model has additionally been regularized by adding dropout layers and using a weight decay of 0.0001 (see Figure 15). This did not bring a better result, but has only made it more difficult for the model to capture the needed feature to reproduce the fine details within the nuclei. However, this brought up a new hypothesis, namely that the model might simply have not enough of capacity to capture enough of the details. In order to confirm this a bigger model (with more filters in each layer) has to be trained.

Interestigly observations made in this section made it clear that metrics used for training (PCC and MSE losses) are indeed not representative enough to derive any conclusions regarding the model quality from them. Just by looking at Figure 16 one can see that for example, training on the full dataset of data gives much better results than training on the small dataset. Yet MSE seems to be bigger for this experiment. Also, it is not representative in comparison of a model without the correct weight initialization vs. the regularized model with augmentations. MSE loss is much higher there, because in general the image became somewhat brighter, even though the quality of the nuclei is significantly better. PCC loss seems to represent the desired quality of the model better. This trend has been noticed by LaChance et al., 2020 as well. They state that values of PCC lack the practical

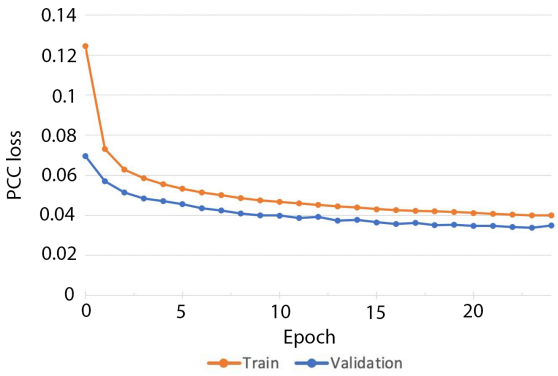


Figure 15: With regularization and augmentations

context — which value would be good enough and good enough in which context? This issue has been addressed here as well in section 3.1.5, where more practical metrics are introduced and the model evaluation on them is carried out.

3.2.2.2 Prediction quality

Resulting predictions are visually very similar to the ground truth fluorescence. The quality of the outputs are confirmed to be already good enough to enable wet lab scientists to advance the CLD process with the use of developed UNet model. Nevertheless, a more thorough visual evaluation of nuclei fluroescence predictions is provided in Figure 17. There are three main visible problems that should be addressed in further research:

- The form of the nucleus is well-captured, but the texture inside is not (the change of intensities) is not predicted very well.
- The border around the nucleus is quite blurry.
- The overall intensities of predictions seem to be higher than the ground truth ones.

All of these problems are quite similar across other organelles predictions as well and can be summarized under the statement that the model does not have high enough resolution. Which can be solved by making a model larger and providing it with more data. In order to confirm this a bigger model has been trained.

Indeed, increasing the number of filters in each convolutional layer by a factor of 4 (switching from from 16 to 64 in the first one) the model was able to capture more details. Predictions became better both visually (see Figure 18) and based on both metrics (PCC and MSE losses, see Figure 16 "bigger model" bar (right)). For more details of the increased inference, training times and their costs refer to Appendix A.

Another rather small issue is that predictions on the border of the crops seem to be worse than in the center. For example, see the cell located at the top left corner in Figure 17 selected into a green circle. Nucleus of this cell is missing in the prediction image as the

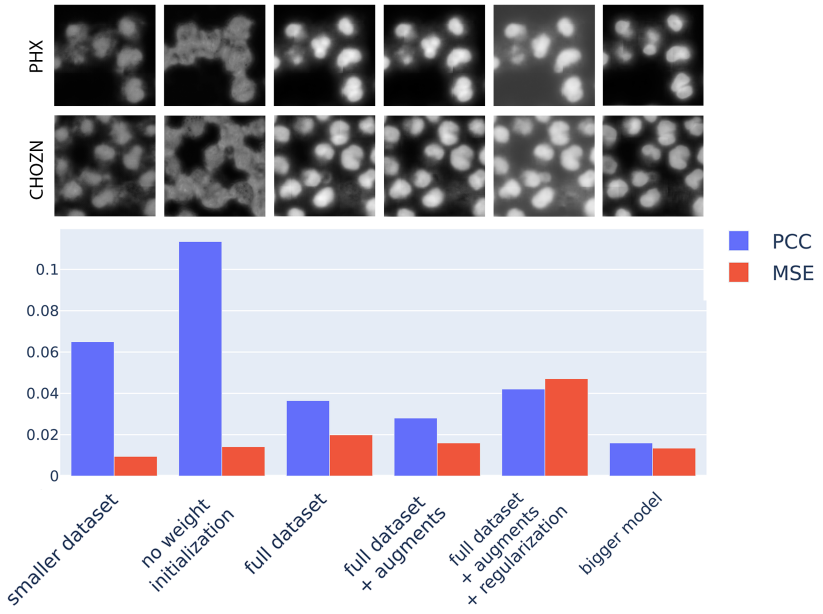


Figure 16: Comparison of different models predictions and scores. PCC loss is representative of predictions quality, whereas MSE loss is not. Increasing model's size significantly improves the results.

cell in not fully present in the DIC input. As will be mentioned in Section 3.2.3, when the cell is not fully present in the crop (split in halves, for example), the network does not have enough data to give out a good enough prediction. However, when combining crops into high-resolution images, this problem can be mitigated by the method described in the next section.

The problem of the higher intensities can also be addressed by a more careful dataset filtration, as several images with overexposed cells are still used for training. This might be a reason for the intensity of overprediction. Blurry borders are also a signal indicating an overexposed nucleus. Even if they are present in the dataset, they can be removed

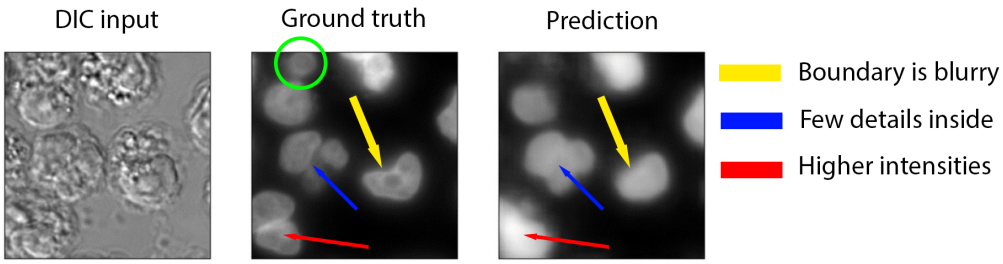


Figure 17: Typical challenges in predictions for nuclei in this study. This figure depicts three main challenges in UNet predictions with nuclei target.

using background removal techniques described in Section 2.4.4. This was not done in the scope of this research as the predictions have had good enough quality to be used in the CLD process.

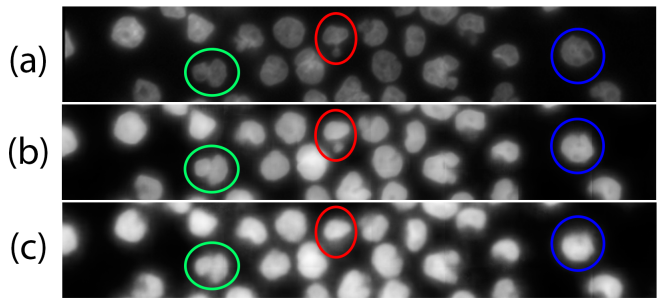


Figure 18: Predictions improvement: (a) ground truth fluorescence; (b) bigger model with four times more filters in convolutional layers; (c) standard architecture. Image resolution (amount of details captured) is clearly better for a UNet of a bigger size (b).

3.2.3 Crops combination technique

Due to the restricted amount of memory on the GPU deep learning models cannot have a high-resolution image as their input in the scope of this research. Yet this is also not obligatory: as the image contains dozens of cells within it, its processing can be limited to a crop of a smaller size. After the model has predicted fluorescence signal for each of the crops, output fluorescence images can be combined together to form a high-resolution image again. In this thesis the architecture of the model assumes an input size of $(256, 256)$ or more specifically $(None, 1, 256, 256)$, where the first dimension is responsible for the batch size and the second one states that the input is a 1-channel image.

There are several ways of how one can split the image, the easiest approach would be to use a sliding window of size w. This algorithm is depicted in Figure 19. A small window starts sliding the image from the upper left to the lower right corner with step size s feeding the selected crops into a deep learning model. From the output of the model only a center part of such a crop is accepted to form a full fluorescence image. Border size b in this case is the size of the edges of the crop that are not accepted from the predictions of the deep learning model.

Accepted areas from each of the crops have to follow each other without any space in between. In oder to achieve that if the border size has been defined in advance, one has to set the step size to in the following way:

$$s = w - 2 * b \tag{49}$$

When step size s is equal to window size w, there is no overlap between the windows.

The reason why the full prediction is not accepted to form the output lies in the following: trained models are less accurate on the borders of the crops rather than in the center. Most of the times there are cells on the borders of the crops that were sliced and therefore it

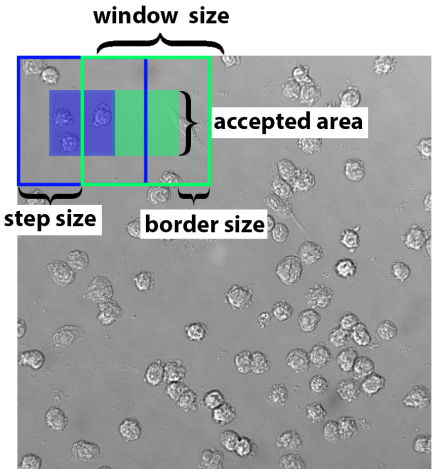


Figure 19: Sliding window approach for fluorescence prediction. Instead of predicting each crop from the image subsequently overlapping crops are used with only the center area being accepted. The size of the not accepted area is defined by a *border size* value. Having a fixed *window size* (defined by a neural network input), the *step size* is defined dynamically.

might be impossible to make a good prediction for them just due to the lack of input information. Therefore, the step size has to be smaller than the window size, so that the windows are overlapping and for each prediction we use only the image center and are allowed to ignore predictions on the border (see the comparison between different border sizes in Figure 20). Such an approach helps to reduce the effect of grid visibility on the image composed of many small crops. This can be seen in the left part of Figure 5 as opposed to the non-visible borders in the same Figure on the right. This would of course take more time to created the predictions, however, the speed is less crucial in comparison to the accuracy of the predictions.

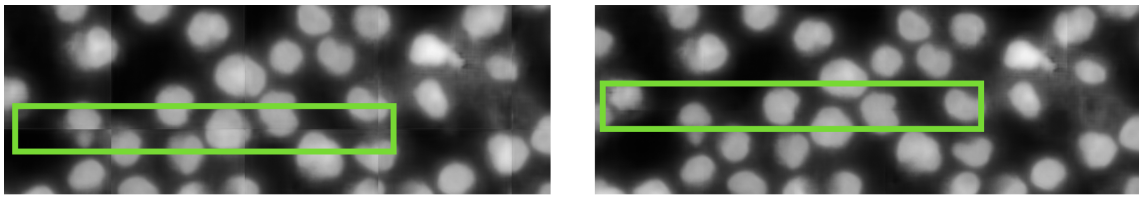


Figure 20: Difference of overlap between predictions on the resulting image. Since predictions tend to give better results in center of the crops, combination of overlapping prediction (right, with not accepted border of size 36 pixels) results in a visually better overall image rather then combination of predictions without any overlaps (left, the whole crop is accepted). Crops intersections are highlighted in green: strongly visible on the left, and almost invisible on the right.

3.2.4 Postprocessing for nuclei segmentation

To properly evaluate the practical biological metrics described in section 3.1.5 on model predictions, one must be able to segment nuclei from fluorescence (as well as from predictions) first. Segmentation in this context refers to the creation of a mask. It should consist of 0s and 1s, with a one being assigned to every pixel that is a part of the nucleus, while all the other ones are assigned with a zero. Although this might be a straightforward task for our eyes, it is not that easy to select separate nuclei via postprocessing. There are several edge cases where the nuclei are difficult to segment.

Even though the most extreme corruptions mentioned in section 3.2.1 were filtered out, some of the images that are corrupted not as severely (meaning they still have all the visible features needed for learning) are still present in the dataset, hence avoiding significant reduction of the amount of data.

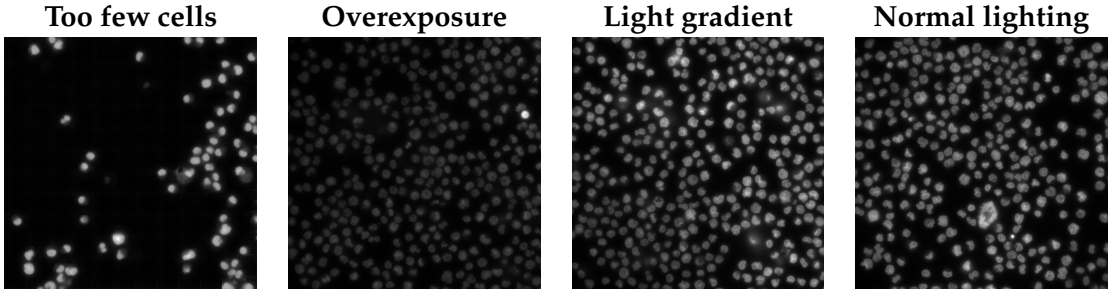


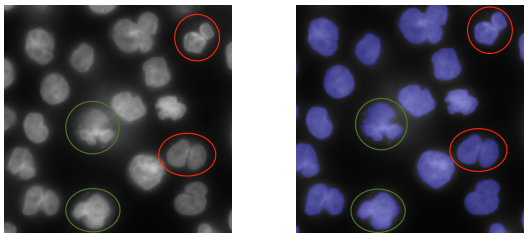
Figure 21: Different challenges in lightning and cells density that do not permit a successful use of a global thresholding for the foreground segmentation.

Examples of cases difficult for segmentation are presented in Figure 21 from left to right: the first image there are too few cells, which leads to the background being much darker than usual; the overexposure of one cell leads to difficulties segmenting the rest of the cells as they are hard to distinguish from the background; lighting gradient from darker (bottom left corner) to brighter (top right corner) region; normal lighting conditions.

Another challenge for segmentation bring nuclei that are very close to each other. This might happen sometimes because some of the cells are currently in the process of division. Also, when some have already fully divided, they might still be located close to one another. The example of such situations is presented in Figure 22.

Here, the cells that are not yet fully divided are highlighted with green circles and the ones that are fully divided, but located too close to one another, are highlighted with red circles. You can see that the segmentation algorithm (see Algorithm 4) recognises both such cases as one nucleus. This algorithm is described below and its steps are visualized in Figure 23.

A more detailed description of the reasoning why local thresholding approach was chosen is provided in the following subsection.



(a) Original fluorescence (b) Segmentation (violet mask)

Figure 22: Closely located (red) and dividing (green) cells represent a challenge for a foreground segmentation as they are recognized as one cell.

Algorithm 4 Fluorescence segmentation of nuclei

1. Normalize image.
2. Apply local thresholding and get a threshold T or a set of local thresholds $\{T_i\}$ and create an initial mask: 1 if $x_i > T$ or 0 otherwise.
3. Apply *fill_holes* transformation to the initial mask in order to get rid of unneeded details within the nuclei.
4. Run *findContours* from opencv in order to obtain separate regions and filter them based on the following criteria: filter out regions that are too big (measure the biggest possible nuclei manually), regions that are too small (measured manually as well), regions that have a shape that is not similar to convex circular type of nuclei. The last filter is done by checking the ratio of the area of the region to the area of the convex hull of the region (for more details regarding *findContours* implementation refer too Satoshi Suzuki, 1985).

3.2.4.1 Thresholding algorithms

Segmentation of the nuclei happens for both datasets — ground truth images and UNet predictions. However, there is big difference between them: ground truth images due to the difficulty of fluorescence imaging acquisition have corruptions like an over- or underexposure, light gradients and etc., while predictions do not have this issue. An inductive bias of the model lies in overcoming these challenges in lightning and as a result the predictions do not have such a variability — on the contrary, they are very stable in terms of the brightness in the images. Therefore the postprocessing for ground truth images and predictions can also vary. At first the choice of a thresholding for ground truth images is presented as it is a more challenging task.

In case of a use of a global thresholding algorithm (more details about this approach are presented in Section 2.4.3) it is very difficult to find a good threshold manually that would work out well for all ground images, especially in this case, when the brightness does differ not only between the images, but also within the image itself (light gradient example in Figure 21). After trying out several global threshold approaches, it has been determined in this study that a *global minimum thresholding* is the best one.

Although global minumum thresholding method performs better than other approaches it, as any global thresholding algorithm, still has a difficulty in dealing with non-uniform

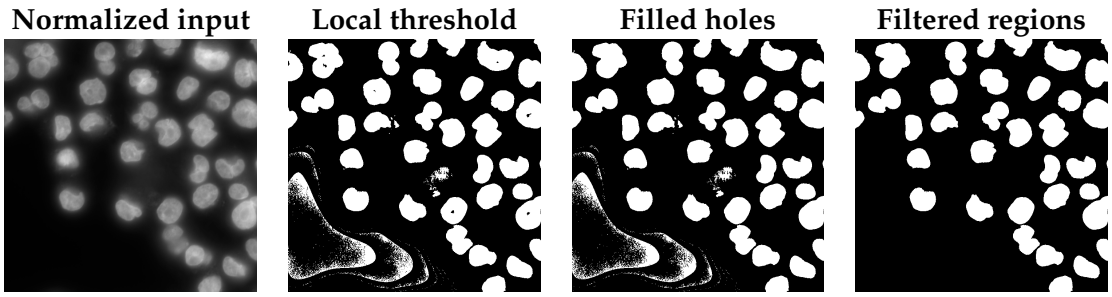


Figure 23: Fluorescence segmentation. Artifacts of local thresholding algorithm visible in the second image can be successfully filtered out based on their shape.

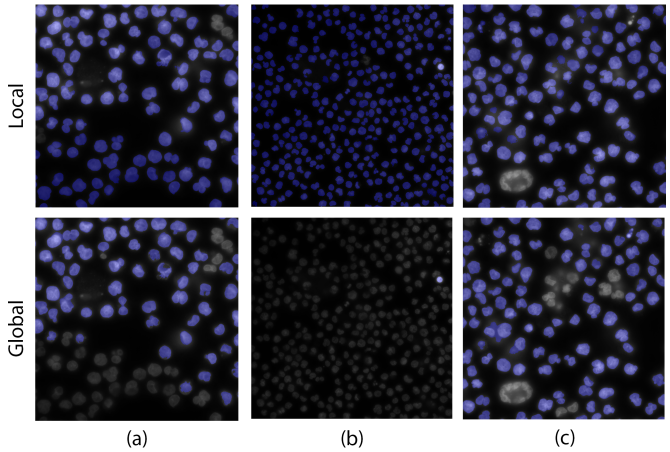


Figure 24: Local vs. global thresholding. A compasion of the results of local and global thresholding algorithms for three lightning situations in ground truth nuclei fluorescence: (a) — gradient in image illumination; (b) — overexposure of one cell resulting in underexposure of all other celss; (c) — normal conditions. Local thresholding is able to segment foreground (nuclei) much better than a global thresholding approach in all cases.

illumination within the images. For visual comparison of global thresholding applied to difficult images see Figure 24a, b. Figure 24c is an image with equal brightness level, however even there some mistakes do appear.

Therefore for nuclei segmentation a local thresholding algorithm was chosen. With the image size of 2136×2136, the local neighborhood (or a *block_ size*) by experimenting with different values was chosen equal to 111 (*Adaptive thresholding* 2022).

Of course local thresholding approach has a longer runtime time (see Table 2). Therefore when the inference speed is crucial one can still use *global minimum thresholding*. It does performs visually a bit worse that a local threshold (especially for the extreme corrupted cases), however for the normal conditions the performance is quite similar to the local thresholding (Figure 24c). As stated in in Algorithm 4 a local thresholding approach was chosen. One should also keep in mind, that after the model is trained due to its generalization ability the gradient in intensities or overexposure will not be present anymore, as it

Local Threshold	Global Threshold
0.3 sec	17 sec

Table 2: Threshold runtime for one image of size 2136×2136

is related to the image acquisition technique and does not depend on the DIC image itself. For this reason for UNet predictions global minimum thresholding can be successfully used. However, as runtime for not crucial in this case, same postprocessing approach was used for both datasets.

3.2.5 Biological metrics

In this subsection the results of the model evaluation on more practical biological metrics are presented. As it has been mentioned before, PCC and MSE losses are often not very representative of predictions quality and it is important to evaluate the model based on the real quantitative metrics that are used by biologists. The evaluation of the model trained on full nuclei dataset with full set of augmentations is presented.

Four biological metrics were measured in this case: number of nuclei, their total and mean intensities and their areas. These measurement were acquired via *skimage.regionprops* method. This method allows to recieve all metrics mentioned above based on the intensity image and its binary mask, that was obtain by segmentation postprocessing procedure described before. Having values of one biological metric for for each nuclei in one image, one can average these values and recieve one measurement corresponding to one image. By calculating such value for every image in the dataset (or four values for every image, since there are four different measurements) a distribution of the metric(s) for the whole dataset can be obtained. Having four distributions for ground truth images and another four for predicted images, allows to compare them in terms of their similarity. More details about this process can be found in Section 3.1.5.

For each metric two plots are presented here: violin plot for visual comparison and scatter plot of ground truth values vs predicted values. Every dot in these plots represents an average value of this metric across all nuclei in one image. Therefore the total amount of dots correponds to the total amount of images in the test dataset. Total and mean intensity measurements are presented in Figure 25 and area and nuclei count measurements are presented in Figure 26.

In general, based on these plots it was concluded that predictions are strongly correlated with the ground truth images in terms of practical biological metrics. Distributions of number of nuclei and their area are very similar in shape, the corresponding scatter plots form almost linear dependance. Distributions of intensities are less similar, the higher intensity of predicted images is proved here. However, high scores in Pearson and Spearman correlation coefficients (see Table 3) suggest that there is a strong link between predicted and ground truth intensity. The difference between predictions and ground truth might be caused by an absolute value shift, that can be easily adjusted for.

Clear improvement in intensity capturing capabilities of a bigger model (see Figure 18) was also captured in terms of the biological metrics. Pearson correlation coefficient for

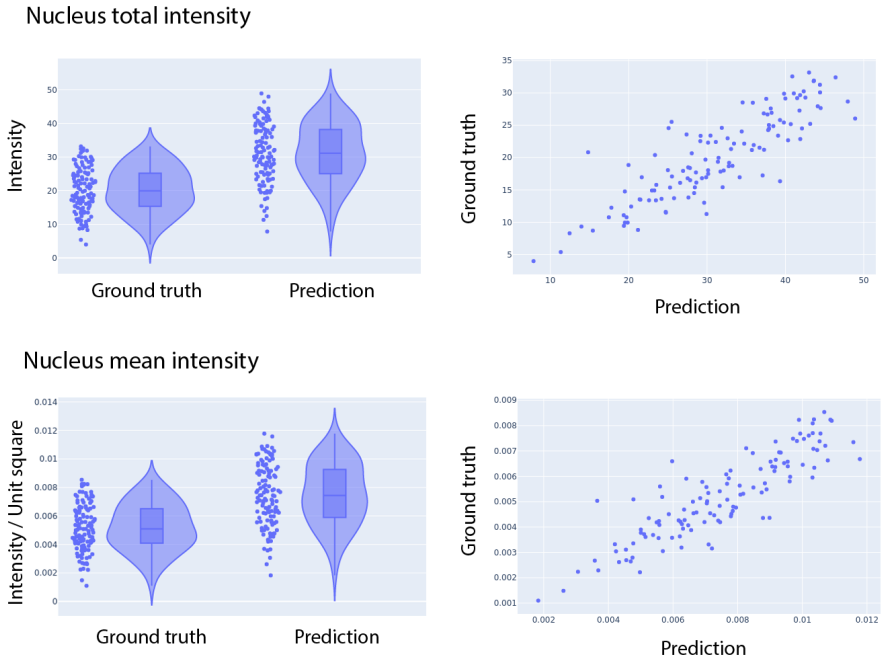


Figure 25: Metrics for practical biological evaluation on nuclei. Total and mean intensities

Table 3: Correlation coefficients for practical biological evaluation on nuclei

	Pearson	Spearman
Number of nuclei	0.982	0.984
Total intensity	0.861	0.855
Mean intensity	0.877	0.873
Area	0.971	0.976

total itensity measurements between predictions and ground truth images has improved from 0.861 to 0.882 and for mean intensity from 0.877 to 0.881. The fact that predictions became less bright was also depicted in Figure 27, where the average of the total intensity values became lower

3.2.6 Influence of scaling on predictions quality

Additionally the quality of the model has been checked for different scales of the input. It was interesting to know whether or not the quality of predictions differs based on the size of the cell within the crop. For this, two additional datasets were prepared: one, where the images were first scaled up by a factor of 1.3 and another one, where they were scaled down by a factor of 0.7. Afterwards the images were cropped as usual into 256×256 crops. Prediction quality was measured based on the biological metrics described in Section 3.1.5 and the results are presented in Table 4.

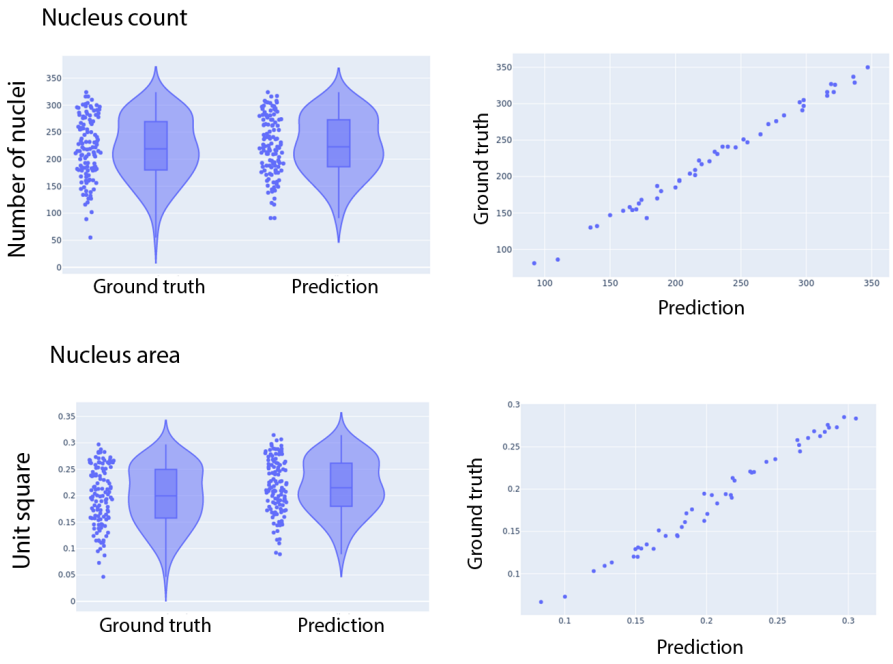


Figure 26: Metrics for practical biological evaluation on nuclei. Count and area

Table 4: Pearson correlation coefficients for practical biological evaluation 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)
Nuclei number	0.987	0.995	0.975	0.971	-0.659
Total intensity	0.902	0.88	0.861	0.856	-0.212
Mean intensity	0.922	0.906	0.88	0.872	0.077
Area	0.991	0.992	0.961	0.952	-0.69

As one can see from Table 4, the bigger the cell is within the image, the better its total and mean intensity can be predicted (see the first column). However, the number of nuclei and their area, are mostly not affected by the size of the cell. Also training on the bigger cells and predicting on the usual ones as well as vice versa, does not change the accuracy as long as the difference is not significant. Nevertheless, when the difference becomes bigger (training on the enlarged cells and predicting on the smaller ones), the quality of the model decreases quite fast. The model trained on images with 1.3 scale gives unacceptable predictions for images scaled down with coefficient 0.7. The model performance for all these values was defined as PCC.

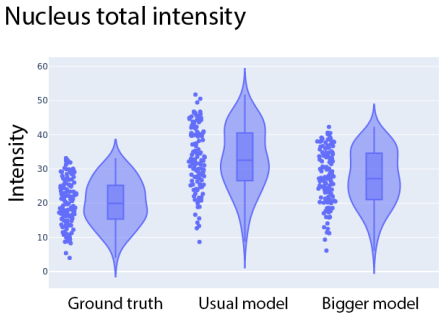


Figure 27: Total intensity: comparison of usual model and a model of a bigger size. With the increased model size nuclei intensities are captured more precisely

3.2.7 Conclusions

Fluorescence staining of nucleus in CHO cells used for recombinant protein production at Merck KGaA can be successfully replaced with *in silico* labeling using deep a neural network. The provided dataset is big enough to achieve accurate predictions, nevertheless further research could be dedicated to improving fluorescence image preprocessing in order to remove blur around the nucleus, as well as to cleaning the data from the under- or overexposed images.

The model successfully converges, and the use of augmentations improves its performance. Pearson correlation coefficient is more representative for the model's predictions evaluation than MSE, and MSE is not representative of a model's quality in this case. The performance of a model is evaluated in terms of practical biological metrics. Further research can also be conducted to advance the architecture of the model, as the use of more parameters improved the predictions significantly both visually and in metrics.

In order to perform segmentation of the nucleus it is recommended to use a local thresholding approach when the inference time is not crucial and global minimum thresholding algorithm when the inference time is critical. Training the model on cells of a larger size slightly improves the intensity predictions. Yet a slight change in cell size does not influence the model's performance significantly.

3.3 Endoplasmic Reticulum

The endoplasmic reticulum (ER) is a network of membranes inside a cell, through which proteins and other molecules move. Ribosomes are small and round organelles with the main function to produce the protein needed for the cell. They are located in a continuous membrane system that forms series of flattened sacs, this membrane is called ER (see Figure 11) (*Endoplasmic Reticulum (smooth)* 2022). ER itself is moslty located around the nucleus. In order to stain the ER, CHO cells were treated with Donkey Anti-Rabbit IgG antibody. This is a fluorescent stain that binds strongly with the ER. The analysis of this cell organelle is also important for the CLD as ER is directly related to the process of protein production within the cell: it is responsible for the synthesis, folding, modification, and transport of proteins (Kara, 2022). In contrast to other imaging datasets ER dataset does not require special preprocessing steps as the images are of a good quality and without any visible problems (see the ground truth image in Figure 29).

3.3.1 Training and predictions

During the training with PCC loss the model has successfully converged already after 35 epochs, however a clear overfit was encountered (see Figure 28a) with the best PCC loss before the overfit being 0.0713. Overfitting happens due to the lack of data, as for the case of ER there were much fewer samples than for nuclei for example (see Table 1). Even though one could use an early-stopping approach and simply choose an earlier epoch before overfit, the better approach would be to use regularization methods described in section 3.1.4.2. In this case data augmentations (flipps and scale crops) were introduced. The new learning curve is shown in Figure 28b. Overfit now happens much later (after 120 epochs) and PCC loss improves to 0.0701. Introducing a stronger regularization with the use of weight decay in adadelta optimizer and dropout layers with a dropout rate of 0.1 (see Figure 28c) reduces the overfit completely, however at the same time increases the loss to 0.099.

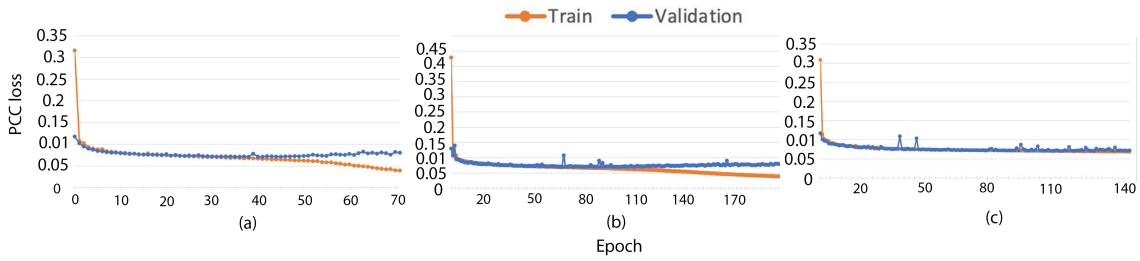


Figure 28: Default model (left), slightly regularized model (middle), strongly regularized model (right)

From the fact that the loss has increased too strong with the use of weight decay, it can be concluded that such regularization approach was too strong. In this case it is better to use early-stopping approach with the epoch that has the lowest losd and as a result the

model trained with augmentations only was chosen.

3.3.2 Combination of nuclei and actin predictions

Now having two models for the predictions of two organelles: nuclei and ER, it is interesting to visualize the predictions together (see Figure 29). It is clear from the image that ER indeed is located around the nucleus. As one can see, there is a great advantage in the used of *in silico* fluroescence labeling especially for the cases where several cell targets have to be ananlysed. Instead of a expesive and time-consuming procedure for staining several targets at the same time they can be predicted based on one DIC image only.

(a) **Ground truth** (b) **Prediction** (c) **Prediction + nuclei**

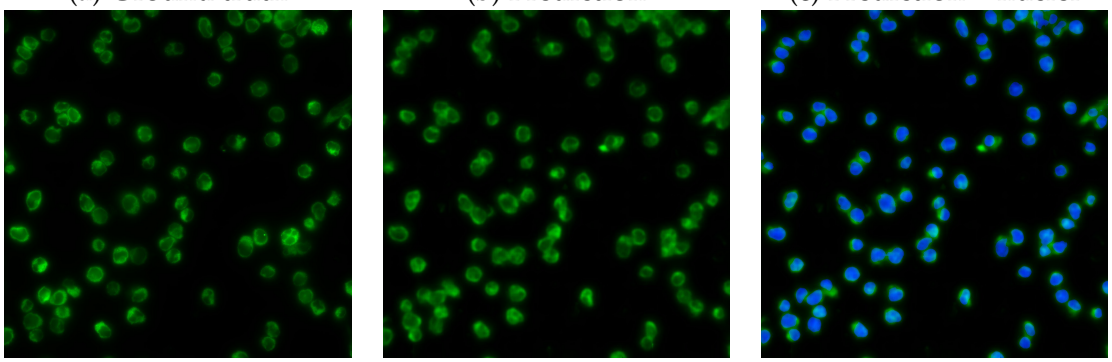


Figure 29: Combination of ER with nuclei prediction. Image (a) here is the original fluorescence image of ER, image (b) is the UNet prediction of ER and image (c) is the combination of predicted ER (green) with predicted nuclei (blue).

3.3.3 Postprocessing for ER segmentation

The process of ER fluorescence segmentation is somewhat different in comparison to nuclei segmentation. This fluorescence staining has a stronger "shining" around the ER itself (see Figure 30) and therefore any method for background removal would be helpful to reduce it. One such method based on the rolling ball algorithm is described in more detail in Section 2.4.4. However, since the prediction results were of a good quality without any special preprocssing, rolling ball algorithm was not applied. Its application would potentially help to separate ERs for individual cells, which is not necessary for our evaluation pipeline.

Due to the non-uniform illumitation a local thresholding approach for segmentation would be necessary here as well. However, one downside of a local thresholding algorithm is the appearance of the artifacts briefly mentioned in the previous section and presented in Figure 31. Even though the background in fluorescence imaging appears to be completely black, it still contains some slight signal (non-zero values), that are boosted by a local threshold and becomes an unwanted artifact.

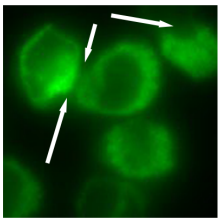


Figure 30: "Shine" surrounding the ER that makes closely located ERs not easily separable.

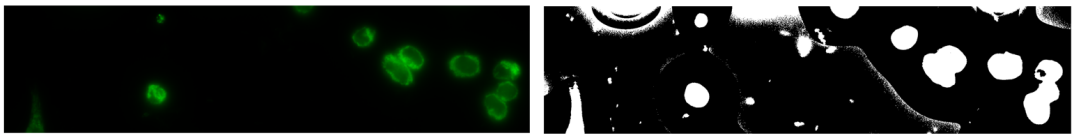


Figure 31: Artifacts from local thresholding algorithm

Such falsely recognized regions appeared in the nuclei as well. There they could be easily filtered out based on the shape criteria. All nuclei are almost round and convex objects, whereas background artifacts are prolonged, non-convex objects. Unfortunately, such filtration cannot be applied to ER imaging as very often ER from one cell is located very close to the ER from another cell, and together they may form a long non-convex object, that now cannot be filtered out this easily. This is why it is very important to remove the background noise here first before applying a local threshold.

In order to do that one can first apply a rough "over-predictive" global thresholding that will cover a true signal fully, including the "shine" around the ER, but will ignore the background noise. In the role of "over-predictive" a global mean thresholding algorithm can be used (see Figure 32.2). The mask created with the mean thresholding approach is used to zero out all the pixels that are not covered by it. And after that the local thresholding can be successfully applied with the *block_size* of 181 (Figure 32.4). The algorithm then fills in all the holes in the middle of ER that might have appeared during the thresholding. Morphological opening (see Definition 2.30) and Gaussian blur with the squared kernel 3×3 are applied. Connected components are detected afterwards and filtered based on the limit of the area they occupy. This mostly filters out very small components from the mask which might have been produced by the left out background noise. The whole algorithm overview is described below:

Segmentation steps are described in Algorithm 5 and also illustrated in Figure 32.

Algorithm 5 Fluorescence segmentation of ER

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 (see Definition 2.30)
7. Run *findContours* from OpenCV in order to obtain separate regions and filter out regions that are too small (for more details regarding *findContours* implementation refer too Satoshi Suzuki, 1985)

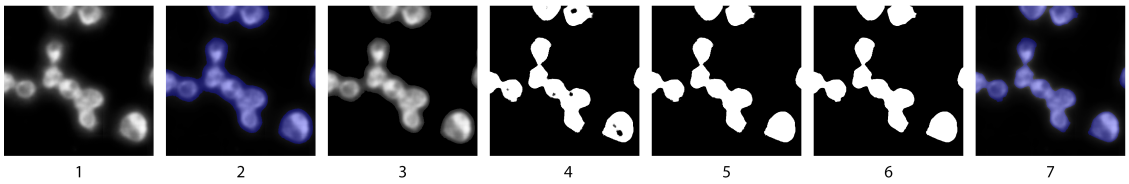


Figure 32: Segmentation steps in ER postprocessing procedure. Steps description: 1 — image normalization, 2 — global mean thresholding, 3 — zeroing out background, 4 — local thresholding, 5 — filling holes, 6 — morphological opening, 7 — filtering out too small regions. See how the "shining" effect visible in the step 2 disappears on the step 7.

3.3.4 Biological metrics

In this subsection the evaluation of the model trained on ER dataset with stronger regularization is presented. Based on the quantitative metrics of Pearson and Spearman rank correlation coefficients it was concluded that the UNet predictions performance for the ER case if even better than for nuclei. The exact values are presented in Table 5. Especially the improvement in mean and total intensities correlations is noticable (lower correlation in case of nuclei (see Table 3)).

The quality of predictions in terms of the number of ER and their area is very good. Both scatter plots confirm this as the line is almost diagonal for both os these metrics. The shapes of the distributions of count and area are very similar as well, which is depicted via the violin plots (see Figure 34). For example, the mean values are almost identical (middle line in the box plot inside the violin).

Visually predictions look slightly brigher, however this difference is less evident after a visual examination in comparison to the nuclei case. Nevertheless, the predictions do tend to have a slightly higher intensities and that was captured in the violin plots (see Figure 33), where is it clear that the ER prediction images are brighter. The mean values inside the box plots confirm this observation by being higher for the predictions.

While nuclei mostly were easily separable, this is not the case in the ER dataset, where several ER can form one bigger region together (see 32.6). In such situation extraction of the intensities metrics from *findContours* method will extract the mean and total intensities of the whole region rather than for separate ERs. Because at the end metrics are averages

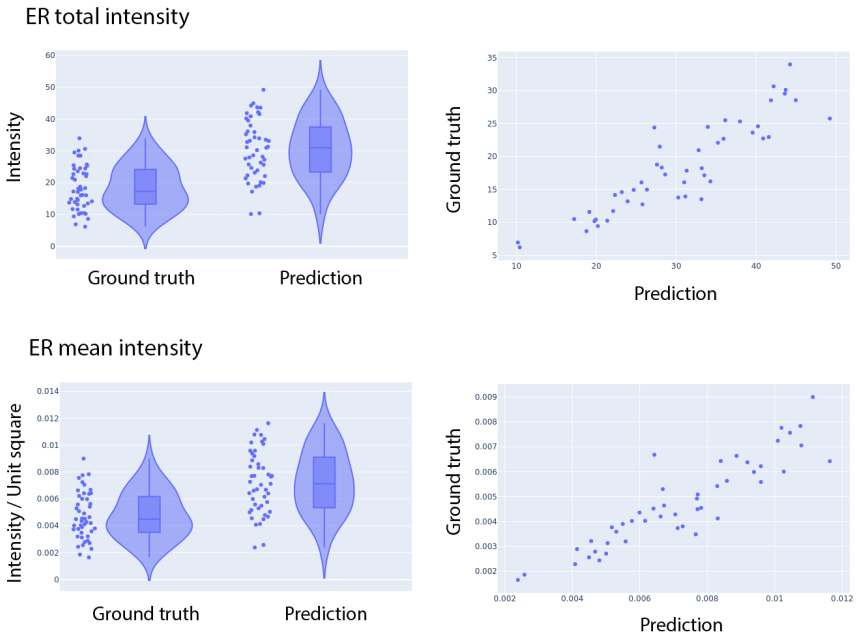


Figure 33: Metrics for practical biological evaluation on ER. Total and mean intensities

across every ER in one image, we did not encounter for this fact as there is not a huge difference in the evaluation. However for a more precise evaluation it might be helpful to apply watershed algorithm to separate the ERs first.

Table 5: Correlation coefficients for practical biological evaluation on ER

	Pearson	Spearman
Number of ER	0.988	0.984
Total intensity	0.952	0.955
Mean intensity	0.941	0.933
Area	0.991	0.990

3.3.5 Conclusions

Training the model on the ER stain images produced very successful results that can be used as a replacement for manual staining with *in silico* labeling. Overfit of the model has been encountered and several experiments using augmentations and weight decay were performed to overcome it, the best checkpoint was selected and evaluated.

The segmentation of ER for further biological metrics evaluation was proposed. The problem of overlap between cells being present was discovered, which causes the segmented regions to merge together. Subsequently, an improved preprocessing algorithm was proposed. This algorithm helps to avoid the artifacts appearing from a local thresholding.

The combination of ER with nuclei predictions was visualized and analysed, it was con-

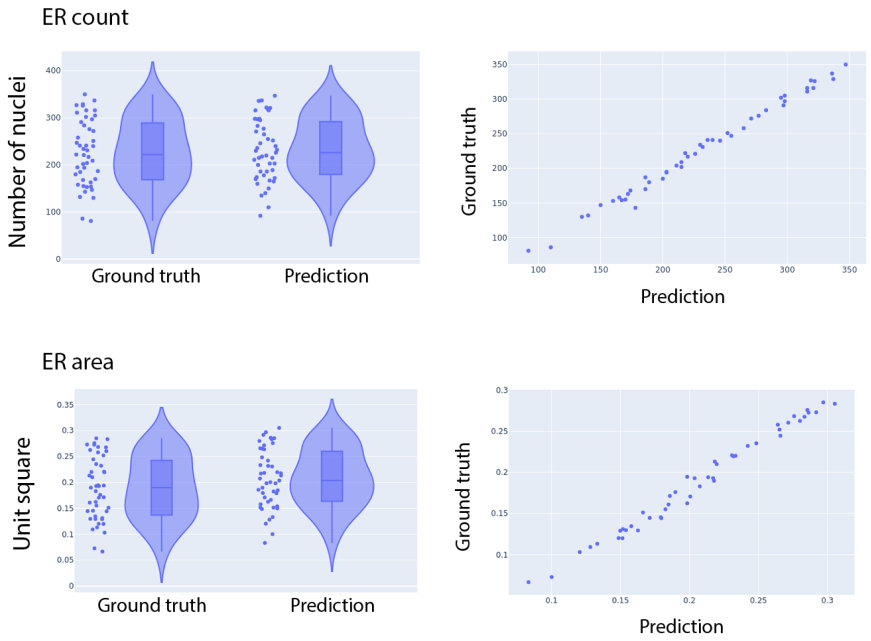


Figure 34: Metrics for practical biological evaluation on ER. Count and area

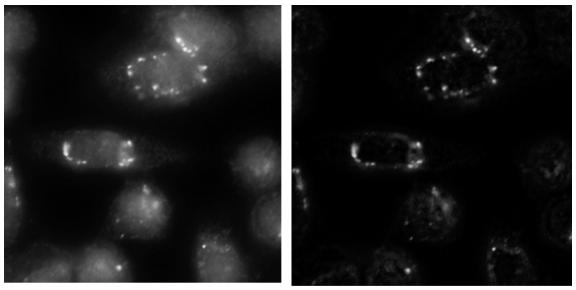
firmed that the models produce realistic results confirming biological definition of their mutual arrangement within the cell. This result shows the power of the combination of the UNet models — they allow to predict several fluorescence targets based on the same DIC input, such a procedure is very difficult to be performed manually.

3.4 Golgi apparatus

The Golgi apparatus (or Golgi, or Golgi complex) is another organelle inside the cell that packages proteins into membrane-bound vesicles which will be exported from the cell. Golgi is also located near the nucleus and ER as well, it is even called a perinuclear body. This location is explained by the biological processes: after a protein comes out of the ER, it goes into the Golgi for further processing (*Golgi body* 2022). The staining of Golgi apparatus has turned out to be the most difficult of all. Two antibodies were tried out in the beginning, however both of them resulted in a low signal-to-noise ratio. Many images were underexposed, and the density of the cells after their fixation was pretty low. The hypothesis why this was the case was that the choice of target protein in Golgi to which antibody can bind to was not the best. Golgi appartus represents an interest for this study, because it is a very difficult fluorescence target. Even having a fluorescence imaging with high signal-to-noise ratio, the lowest scores among all cell organelles in state-of-the-art paper Cheng et al., 2021 were achieved on Golgi.

3.4.1 Preprocessing

One can see an example of what Golgi fluorescence staining looks like in Figure 35. It is evident that there is a lighter foreground fluorescence and slightly darker one in the background. A true Golgi signal here is considered to be only the lighter part of fluorescence lighting. The light gray background present here is called a non-specific fluorescence lighting. It comes from the cell itself, and might occur when the antigen is impure and contains antigenic contaminants (Borek, 1984). Its brightness may vary due to longer or shorter exposure times.



(a) (b)

Figure 35: Golgi preprocessing: (a) — original signal, (b) — image with background removed (using rolling ball algorithm)

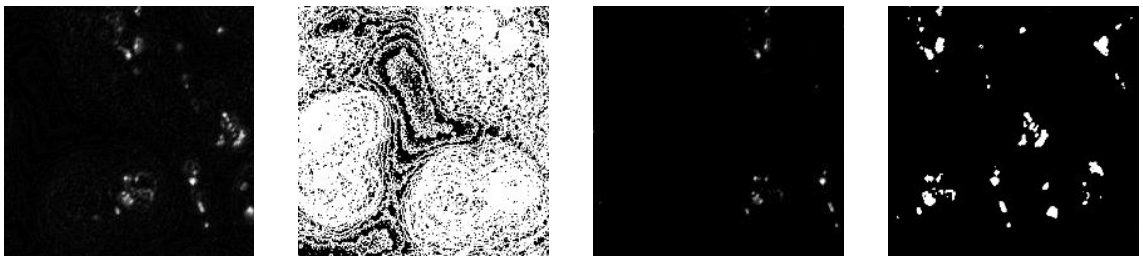
Having such a non-specific fluorescence background has two challenges for training:

- The relative area of the background fluorescence is bigger than the area of Golgi themselves. Therefore quite a big part of the loss during training will be dedicated to teaching the model to restore this background fluorescence instead of the Golgi itself.

- It introduces difficulties during the postprocessing of the predictions. As well as for nuclei, the mask of the predicted Golgi apparatus is needed for further evaluation of the biological metrics. Using the same algorithm for postprocessing segmentation that was used for nuclei, the mask of the Golgi Apparatus will consider the background noise to be relevant, although this is an unwanted behavior.

In order to get rid of this background noise, the background removal approach described in Section 2.4.4 can be used. In the left part of Figure 35 one can see the original ground truth image and on the right side we can see the image after the background was subtracted.

Additionally, all of the crops used for training were filtered base on the amount of background they contain. Without filtering the crops that contain a lot of background, a big portion of a dataset would consist of black crops only, which creates very strong class imbalance. Subtracting the background with the rolling ball algorithm unfortunately is still not enough to get a reasonably clean signal of Golgi apparatus from fluorescence imaging, as a lot of background noise will still be present there. In Figure 36a one can see a fluorescence image preprocessed with a rolling ball algorithm. After turning it into a binary image that can be seen in Figure 36b, it becomes clear that the images still contains a lot of background low-intensity noise, which was not visible for an eye. In order to get



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

Figure 36: (a) Basic preprocessing with automatic background removal algorithm only; (b) binary masked of subfigure (a); (c) Additional clipping of lower intensities after vanilla pre-processing; (d) binary mask of subfigure (c)

rid of it one could clip lower intensities of the image via image enhancement. In order to do that one can simply clip all the intensities below 90-th percentile of the image and normalize it afterwards. The result of the clipped image and its mask are illustated in Figure 36c, 36d correspondingly. It contains almost no background noise now. Such additional clipping has improved the results slightly.

3.4.2 Training and predictions

Applying the usual model architecture with PCC loss to train the model to predict fluorescence signal from DIC imaging for Golgi did not yield good results. The model seems to converge (see Figure 37), however there is no learning happening.

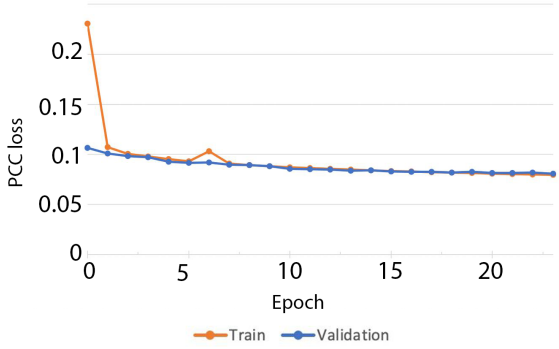


Figure 37: Straightforward training does not seem to lower validation loss significantly

This was also depicted in visual predictions where the predicted fluorescence mostly contains dark pixels only (see Figure 38 prediction). Although it seems that some pattern is hidden behind the dark pixels, visualizing it simply shows that the model picks up on the cell outline itself and does not provide any useful information on the location of Golgi. One can see this by normalizing the predicted dark image to the range $[0, 1]$ (Figure 38 normalized).

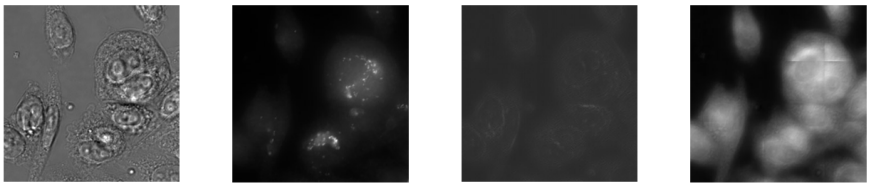


Figure 38: Training on original data

While in Figure 38 only the results of predictions for one crop are presented, it would be interesting to see what the image with combined crops would look like. This is presented in Figure 39. An interesting pattern can be seen here: essentially the model is predicting some signal across the whole cell with a brighter region in some of them. Yet these regions do not have a correct location with respect to Golgi. Also, it is important to keep in mind that the image to the right in Figure 39 is a normalized one and a true image is almost fully black.

This experiment created the hypothesis that the background removal algorithm might have reduced the signal-to-noise ratio and removes some important fluorescence signal along the way. Therefore an alternative approach of background removal had been tried, where only enhancing via clipping described above has been applied directly on fluorescence imaging (without using a rolling ball algorithm). This produces images that still have much more non-specific fluorescence background (see Figure 40b), yet this preprocessing alters the initial image to a much lower extent.

In this case predictions are not completely black anymore, however their quality is still

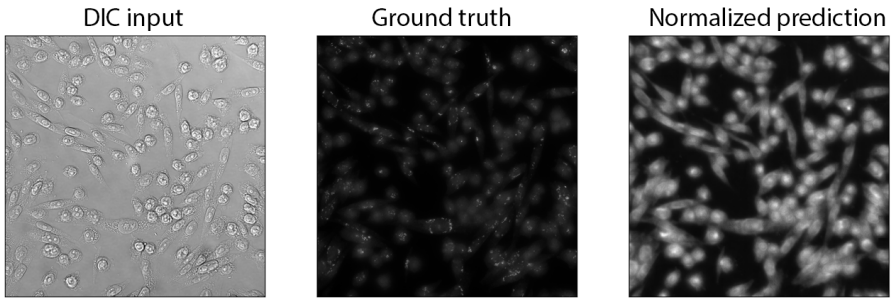


Figure 39: Full size predictions

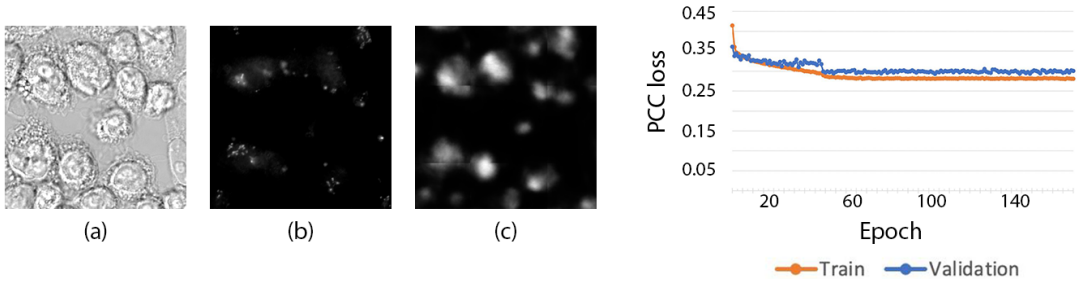


Figure 40: Training on the enhanced data only (without rolling ball algorithm)

very low. Images used in these training experiments were effectively coming from different staining procedures, using different antibodies, fixation processes and microscopy settings. This raised the idea of training the model on few selective datasets only that were created with the first staining approach in mind. The preprocessing was chosen as in the previous example — with the use of enhancement and without rolling ball algorithm as it has shown the best results.

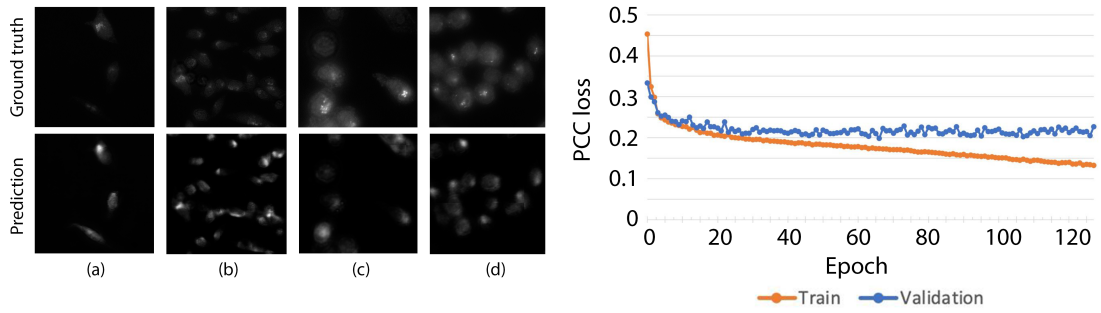


Figure 41: Small subset of the best staining

The predictions from this experiment can be found in Figure 41. Due to the stricter filtering and the use of much less data there were much smaller crops which were used for training. In this case only 4251 crops were used for training and 406 for validation

purposes, which directly led to overfitting after 70 epochs. However, the predictions did improve visually and the loss dropped to 0.19. Nevertheless the losses from these two experiments should not be compared directly as they were evaluated on two different validation sets. The improvement of the results indicated the careful choice of data samples taken from the same staining approach with the same settings, which do not require a severe preprocessing and already have a high signal-to-noise ratio. This might help the predictions significantly. There is a potential here for an improved regularization of the model trained on the small subset of data. Yet it is clear that the acquisition of a better fluorescence staining is crucial here.

3.4.3 Alternative ways to improve predictions

3.4.3.1 Asymmetrical losses

The earliest learnings from the experiments have shown that the severe class imbalance is present and the model tends to predict mostly pure background — black images. In order to overcome this an asymmetrical loss can be used during training. Similarly to a weighted loss for a classification problem with class imbalance issue present, a weighted loss for segmentation task can be introduced. In this case different pixels from the prediction will receive a different weight based on some criteria. While the use of weights cannot be easily defined for Pearson correlation coefficients, it is possible to use them with MSE loss, because weights coefficients there can be added directly in front of the squared difference between pixels.

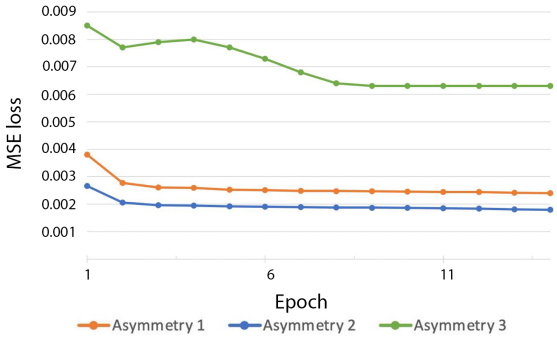


Figure 42: Punishing over and under predictions with asymmetrical MSE loss

Figure 42 presents training learning curves from three asymmetrical approaches. Let Y be a ground truth image and $\hat{Y}$ be a prediction.

Asymmetry 1 is aimed to punish an underprediction: when a model's pixel prediction is lower than a true one the loss will be higher. This resulted in slightly brighter images.

$$R = \hat{Y} - Y$$

$$L_{i,j} = \begin{cases} {R_{i,j}}^2, & R_{i,j} > 0, \\ 2{R_{i,j}}^2, & R_{i,j} < 0 \end{cases}$$

Asymmetry 2 is aimed to punish the errors on brighter pixels more: when a model's pixel prediction is higher than a true one the loss will be higher. Yet this loss also encourages the model to underpredict and results in completely black images even though it has the lowest loss.

$$R = \hat{Y} - Y$$
$$L_{i,j} = \begin{cases} {R_{i,j}}^2, & R_{i,j} < 0, \\ 2{R_{i,j}}^2, & R_{i,j} > 0 \end{cases}$$

Asymmetry 3 is a stronger version of Asymmetry 1 and it results in a fully black image.

$$R = \hat{Y} - Y$$
$$L_{i,j} = \begin{cases} {R_{i,j}}^3, & R_{i,j} > 0, \\ 10{R_{i,j}}^2, & R_{i,j} < 0 \end{cases}$$

Where in all cases resulting loss for the image is the mean of the losses of each pixel $L = \frac{1}{N^2} \sum_i \sum_j L_{i,j}$ where N^2 is the number of pixels in the image.

All of the approaches above do not bring a significant change in the performance and a mostly black image remained to be an output. Interestingly, punishing underprediction has essentially backfired, as loss then supports an overprediction. Because setting the weights of one class to be smaller amounts to the same as setting the weights of the other class to be larger, and here as a result the model is more likely to overpredict. That is why the second asymmetry has a better loss, even though the logic behind it is not that obvious at first.

There were other interesting approaches in asymmetrical losses tested that are depicted in Figure 43:

1. Adjusting overall brightness. Leads the absence of bright spots and brightness gradients in the prediction. The illumination of the cell becomes uniform across all cells.

$$L = L + \frac{\sum_i \sum_j \hat{Y}_{i,j}}{\sum_i \sum_j Y_{i,j}} \tag{50}$$

2. Adjusting overall brightness with a reversed division. Leads to fully white images as this would minimize the fraction in loss.

$$L = L + \frac{\sum_i \sum_j Y_{i,j}}{\sum_i \sum_j \hat{Y}_{i,j}} \tag{51}$$

3. Multiplying loss with prediction will result in black images again, however multiplying with the ground truth yields more interesting results. However, the model is pushed to predict average gray color across the entire image for the most part.

$$L_{i,j} = \left(Y_{i,j} - \hat{Y}_{i,j}\right)^2 * Y_{i,j} \tag{52}$$

4. Multiplying loss with 1 - ground_truth also results in completely black images as such loss puts more puts more emphasis on the correct prediction of the background.

$$L_{i,j} = \left(Y_{i,j} - \hat{Y}_{i,j}\right)^2 * (1 - Y_{i,j}) \tag{53}$$

5. This improves the asymmetry approach 3, however now the highlighted regions simply include the whole cell.

$$L_{i,j} = \left(Y_{i,j} - \hat{Y}_{i,j}\right)^2 * Y_{i,j} \tag{54}$$

$$L = \frac{\sum_{i,j} L_{i,j}}{N^2} + \frac{\sum_i \sum_j Y_{i,j}}{\sum_i \sum_j \hat{Y}_{i,j}} \tag{55}$$

6. Usual MSE.

7. Usual PCC.

From the experiments it became clear that the best approaches are PCC, pure MSE or MSE with the adjustment of overall brightness.

Figure 43: Results of advanced versions of MSE training

3.4.4 Conclusions

The Golgi apparatus is the most difficult target in this research and the models trained on Golgi data require further research. The difficulty of Golgi predictions can be attributed to the low signal-to-noise ratio of the acquired ground truth fluorescence imaging, overall low density of the cell in images and the inherent challenges of antibody staining of the Golgi apparatus.

Golgi imaging requires a strong preprocessing such as enhancement and background removal with rolling ball algorithm due to the presence of non-specific fluorescence lighting. However, even these approaches do not significantly improve the quality of training data and the predictions are still not accurate. Cheng et al., 2021 put a special attention on the significant importance of a good signal-to-noise ratio for quality Golgi fluorescence predictions. The aforementioned paper uses an advanced denoising algorithm called *noise2void* (Krull et al., 2019). This is a strong method that does not require pairs target clean signal and noisy images available, but simply assumes that that there are only noisy images available. They use the idea that if one could acquire multiple images with the same

signal but with different realizations of noise, then after averaging across these images the resulting image would approach the true signal. This might be a useful approach for denoising Golgi imaging instead of the use of enhancement and rolling ball approaches. It is highly advisable to try this method in future research. However the initial exploration of this method has shown that it is very expensive in terms of the computational costs.

One of the insights from the predictions is that there are not enough details inside the predicted fluorescence. The texture and the gradient seem to be missing and predictions are too smooth. Although this is not very crucial for other organelles, Golgi apparatus stain on the other hand has a very granular structure and consists of small dots. In order to introduce the granularity we suggest to try the introduction of a gradient of an image into the loss function. For example, by calculating the image gradient with Sobel filters for ground truth and prediction and incorporating the difference into the loss function (see Yao, 2016).

3.5 GFP

Green fluorescent protein (GFP) is protein that produces bright green fluorescence from the whole cell surface. Some cells express this protein naturally, therefore there is no need to perform cell staining or fixation in this case and the imaging can be done on living cells. The difference in DIC imaging between fixed and not fixed cells is presented in Figure 44. Clearly, the cell membrane is intact and clearly defined in not fixed cells, whereas in fixed ones there is almost no definite membrane present.

Nevertheless, in order to get training data for all other targets (nuclei, ER, Golgi apparatus) the staining procedure is unavoidable. And staining requires the cells first to be fixed. That brings up the limitation of current *in silico* labeling research — cells have to be fixated in any case. DIC imaging of living and fixed cells looks very different and models trained on fixed cells do not generalize to not fixed ones well. Luckily, fixing cells is not a cumbersome lab procedure and is far easier than staining the cells, which is avoided with the help of *in silico* fluorescence labeling. After successful training of the model on living GFP expressing cells, we found that other models cannot perform that well on living cells. Therefore, the cells were fixed in order to look like previously acquired data and the experiments were repeated. Nevertheless, we recommend to look into the possibilities of transfer learning from fixed to living cells. The results of training on the fixated cells are presented below.

For this experiment another cell phenotype was chosen — H19. Training the model to predict GFP fluorescence essentially allows to predict the area of the whole cell, which is used in further biological analysis. There is no need to capture the intensities as they do not bring any useful features for the selection step in CLD. However, they might help for algorithms like watershed to find separate cells in order to count them. Yet the task can be simplified too to predicting a binary mask of GFP signal too. Binarized images are well-suited for cell area predictions. Although one has to find a corresponding image preprocessing pipeline in order to convert training intesity fluorescence imaging into masks first. Both training variations are provided in this chapter — with and without prediction of the intensities.

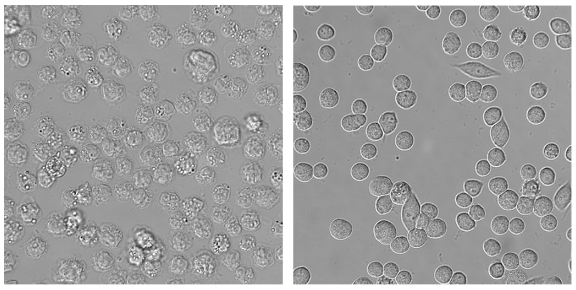


Figure 44: Examples of fixed and not fixed cells DIC imaging

3.5.1 Preprocessing

In order to convert intensity fluorescence image into a binary image, any of the previously described threshold algorithms can be applied. Here intensity of GFP in different cells may differ due to some of the cells being in focus and others being outside of focus, that is why local thresholding algorithm that was used in this case has better results than any global thresholding approach.

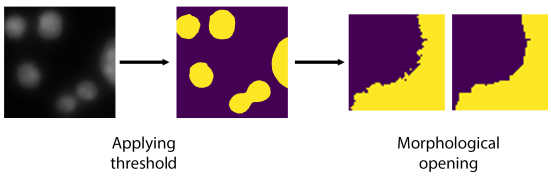


Figure 45: Converting GFP into a binary mask

Figure 45 displays the pipeline of converting an intensity image into a binary mask. After applying local threshold algorithm, additional morphological opening operation has been performed in order to smoothen the borders of the cell and remove noise.

3.5.2 Predictions

Figure 46 presents the result of training with PCC loss on intensity images. The model convergence and visual comparison between ground truth and predictions is displayed. One can see that some of the cells are present in predictions, but are missing in ground truth images. These are dead cells that do not express GFP anymore, however the cell body is still present in DIC. An additional observation is that the boundary is generally more blurry than in ground truth fluorescence, which is typical for all previous organelles as well. However, all cells are clearly visible and can be separated using additional image postprocessing pipelines.

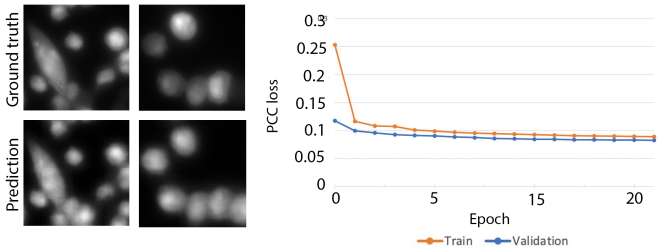


Figure 46: Training with Pearson correlation loss

In Figure 47 we can see the results of training on the binarized image dataset. The model converges and the predictions are successful as well. Without the intensities it might be

more difficult to visually separate the cells from the mask, as opposed to the image with intensity values. The predicted masks are not binarized yet contain continuous values from the interval $[0, 1]$. The following observation shows how the model generalizes: preprocessed ground truth image still has not very strict boundaries in its masking even after local thresholding binarization. After zooming in one can see that some that the boundaries of some cells were oversegmented. This happens due to the different intensity values for different cells. However, this is not a an issue as the model is still able to generalize well and predicts the middle part of the cell confidently, then smoothly reduces the confidence on the cell boundary. After thresholding such predictions one can choose such a threshold that will get just enough of the cell boundary needed. In our case the value was chosen to be 0.8.

An interesting detail here are dead cells mentioned above. They are present in DIC, but do not express GFP. One can see a clear example of such cells selected with green circles in Figure 47. The model generalizes in a way that it continues to segment all of the cells regardless of their state. This is an expected behavior as the amount of dead cells in the dataset is pretty low and occasional mistakes do not push the model strong enough to be able to learn the features that separate dead from living cells. We proceed with the existing model for futher evaluation, however it might be useful for further research to address this issue. For example, Christiansen et al., 2018 has demonstrated the ability of their model to differentiate dead cells from alive ones by staining the dead ones and then training the model to differentiate between them.

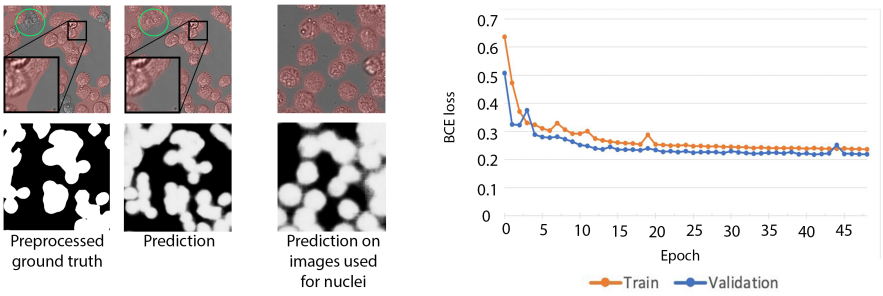


Figure 47: Training with BCE loss

3.5.3 Biological metrics

In this case only two metrics are important for evaluation — the cell area and the cell count (see Table 6). Both Pearson correlation and Spearman rank coefficients are lower than in previous experiments. Influence of additional segmentation of dead cells by the model lowers correlation scores, however they still signalize the presence of a strong correlation between prediction and ground truth. One of the key results of training this models is the ability to calculate the area of the cell. In combination with the nucleus area this gives a ratio of the nucleus size to the cell size. This ratio is very important in cell line selection as a lager fraction of cell corresponding to nucleus means more productive cell.

Table 6: Correlation coefficients for practical biological evaluation

BCE loss	Pearson	Spearman
Number of ER	0.67	0.64
Area	0.82	0.75

Violin and scatter plots depicting the two above mentioned metrics are shown in Figure 48.

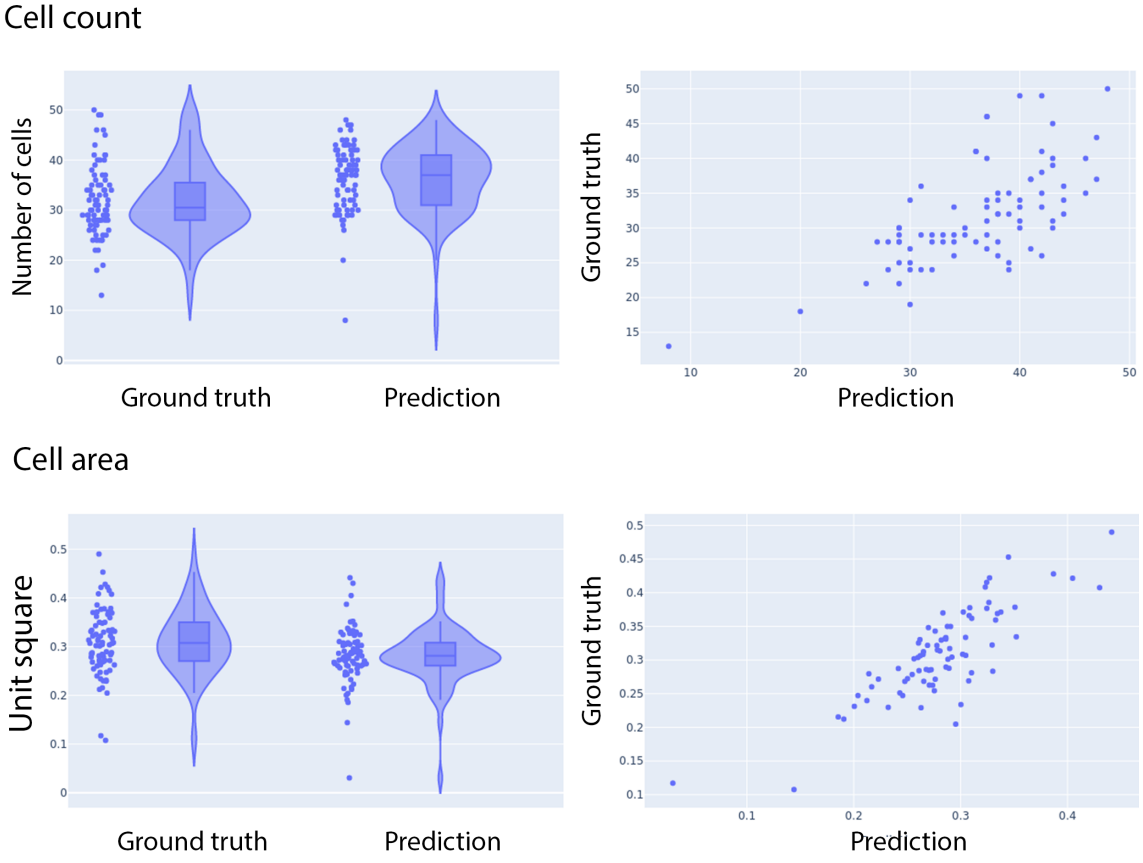


Figure 48: Biological metrics

3.5.4 Combination of GFP, nuclei and ER

Now having three successful models that are able to predict GFP fluorescence from the whole cell, nuclei and ER, one can combine their predictions for the same image. To do that an output RGB image was constructed, where each prediction takes one channel. The resulting image is shown in Figure 49. Additionally one can see that the GFP model has successfully generalized on other cell phenotypes (CHOZN, PHX). Originally this model was trained on H19 phenotype only, however predictions for two other phenotypes (see Figure 49 PHX, CHOZN) highlight the cell area successfully as well.

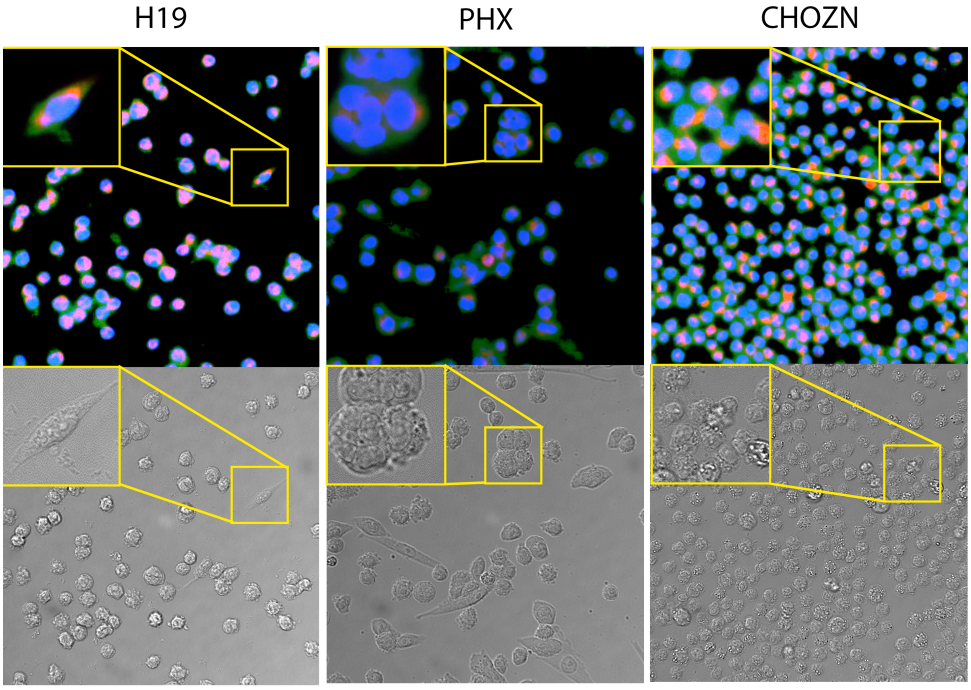


Figure 49: Combination of predictions of three UNets: GFP, nuclei and ER. Each organelle occupies one RGB channel: red — ER, green — GFP, blue — nuclei.

3.5.5 Conclusions

GFP *in silico* fluorescence labeling can successfully be used for counting the number of cells and estimating their size. The created models can both predict the intensity images as well as binarized masks. Intensity predictions might be more useful for the number of cells estimation as the cells better visually separate there. The models were evaluated on two biological metrics: cell size and cell count. The correlation coefficient suggests a strong correlation between the predictions and ground truth.

The model has limitations in its ability to differentiate between dead and alive cells. This issue should be addressed in further reasearch after acquiring data for labeling dead cells. Despite the difficulty to clearly determine cell boundaries during image preprocessing for fluorescence image binarization (its overpredicting in some cases) the model can successfully generalize and predict a correct boundary for all cells. The images of not fixed cells were provided for the first time during GFP experiments and it was clear that previously trained models do not generalize on them well. Because of this the limitation of research in terms of the cell fixation need was determined. However, it is recommended to look into possible transfer learning approaches get rid of the fixation step completely.

4 Model robustness and drift detection

In this chapter the stability and robustness of the trained models are presented. First subsection shows models' performance on the corrupted DIC input data with two sources of corrupted signal: artificial pseudocorruptions and real corruptions from the microscope settings. It presents the influence of augmentations on the robusteness, studies the influence of corruptions on practical biological metrics and shows how generalizable the models are across phenotypes. Second subsection presents a study of the image representations in the UNet embeddings including their possible clustering in the lower dimensional space based on several possible clustering hypotheses. And finally the last subsection covers the results of the drift detection algorithm built to alert end users when models predictions become unreliable. Which happens when there is a difference between the data used for training and the data used during inference.

4.1 Corruptions

For practical reasons it is important to not only evaluate the models on high-quality data exclusively, but also to know how the predictions will degrade when the input's data quality decreases. Having a model for fluorescence *in silico* labeling that can additionally alarm end users when the predictions should not be relied upon is very useful in practice. Although the DIC microscopy is a relatively easy technique, there are still setting up procedures taking place that can be prone to errors. Additionally, as the models are not easily generalizable across fixed and not fixed cells, an alarming system that is able to catch these situations would be useful to save time and cost of lab work. In order to measure the stability or robustness of the models towards data degeneration they were evaluated on the corrupted or "bad" input DIC images. There are two sources of "bad" images that can be used for such estimations. The first are actually corrupted images made in the laboratory. Such corruptions may come from different sources: for example, an oil bubble landed on the microscope lenses, low density of the cell on the image, over- or underexposure during image acquisition. Another source of image corruption would be images with artificial or pseudocorruptions created manually via image processing. They allow more systematic investigation of the impact of a corruption effect. Artificial corruptions allow to vary the severity of the corruption keeping the original fluorescence data intact.

4.1.1 Artificial corruptions

In this subsection results from evaluating models on three types of artificial image corruption are presented, namely: defocus blur that imitates the defocus of the microscope lenses, changes in brightness and changes in contrast. Every corruption has different effects on the prediction of the model based on its severity level. Therefore it is important to evaluate the error-rate (in this case a loss function) for the predictions for different

severity levels of each corruption type presented. It is also important to perform a visual evaluation of the model's predictions on corrupted data. Each corruption c has severity levels s, where $-5 \leq s \leq 5$ ($0 \leq s \leq 5$ for defocus blur corruption). 0 here corresponds to an original image without corruption. It is important to keep in mind that although severity levels were chosen to be as much comparable between one another as possible, they still might have differences in their strengths. For example, contrast has much stronger effect on predictions than brightness changes. Three types of artificial image corruption are presented below.

4.1.2 Defocus Blur

Defocus blur corruption imitates the effect of defocus on the microscope. The blur is applied to the image by convolving it with a special defocus kernel. There are two tunable parameters for this corruption type: the first one is the radius of the circle in the kernel r, and the second one is the blur strength parameter s. An example of the kernel with radius r is shown in the Figure 50.

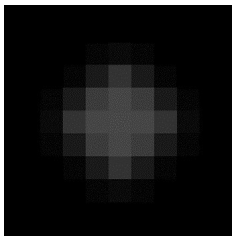


Figure 50: Defocus blur kernel

4.1.3 Brightness

Different brightness levels are also an important image corruption to perform tests on. Brightness variations appear often in the dataset during image acquisition. In order to change the brightness, an image from the RGB format was translated into HSV format, which stands for hue, saturation and value. This is also one of the popular formats to represent an image. To make an image brighter or darker, one can simply add or subtract a parameter s in a value channel for each pixel $x_{i,j}$ correspondingly. This parameter is often called bias. The bigger absolute value of this of this change, the stronger a corruption will occur.

$$\hat{x}_{i,j} = x_{i,j} + s \tag{56}$$

4.1.4 Contrast

In contrast to adding a constant value pixelwise to an image in order to change a contrast level, one can perform a multiplication of an image with another constant s. This

parameter is often called gain.

$$\hat{x}_{i,j} = s * x_{i,j} \tag{57}$$

For both contrast and brightness changes one can use $cv2.convertScaleAbs()$ from the OpenCV library. This method directly accepts gain and bias parameters and clips the image to stay within the allowed range of values.

The values of hyperparameters used in corruptions (kernel radius, gain and bias) are stretched across the range of severity levels and presented in Table 7.

Table 7: 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

Severity level of -5 for contrast represents a highly contrastive image, while 5 is a very low contrast image. For brightness corruption levels of -5 and 5 correspond to images with very low and high brightness respectively. And defocus blur corruption has only 5 levels of severity (both strength and kernel radius were varied at the same time), ranging from the original image (level 0) to the image with a stronger blur (level 5).

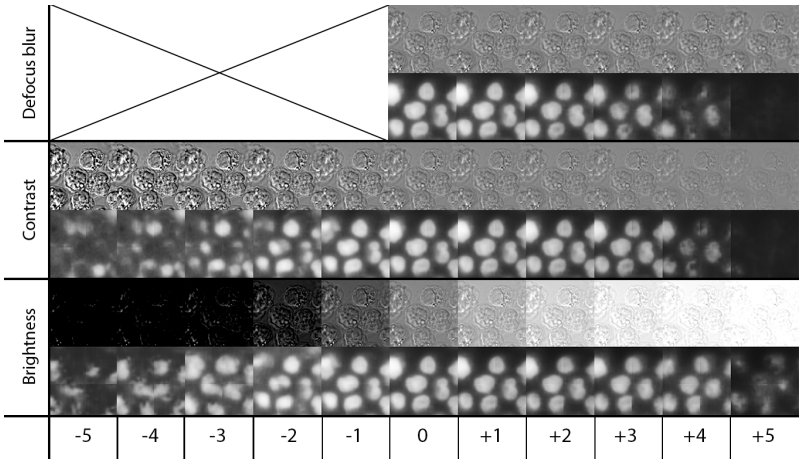


Figure 51: Influence of artificial corruptions on the predictions. Three artificial corruption types along with their influence on predictions are presented here. There are 5 different severity levels for every corruption (towards positive and negative direction), and severity level 0 represents an original image.

One can observe the input image change for each of the corruptions along with the change of prediction of nuclei model in Figure 51. It can be clearly seen that the model's predictions are quite stable towards different brightness levels and contrast. However, the predictions on the crops are very sensible towards defocus blur corruption: DIC images

with defocus blur levels $1-4$ are almost indistinguishable from the original image, yet the model's predictions degrade quite fast. This can be explained by the fact that the training dataset contains quite diverse data in terms of contrast and brightness levels and, as a result, the model is more stable towards these changes. Using defocus blur as an augmentation will help to solve this problem. This will be described in more detail in Section 4.1.6.

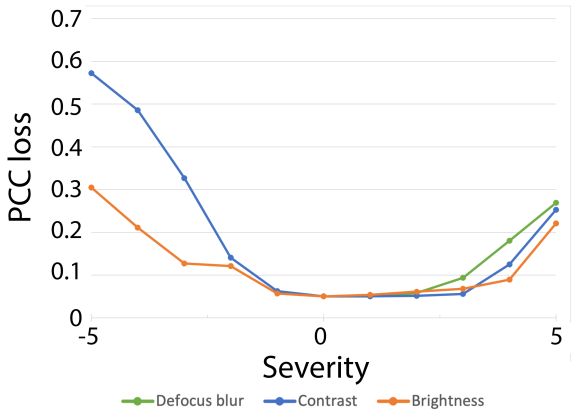


Figure 52: Change of PCC loss for artificial corruptions

Additionally, a change in PCC loss is presented in Figure 52. This is a plot of PCC loss for different artificial corruptions. Loss increases for stronger severity levels. It can be seen that in a positive direction with defocus blur corruption the model degerates more quicker and that contrast corruptions change predictions more severely in the negative one.

4.1.5 Real corruptions

Another interesting set of experiments was conducted on the image data that exhibits real corruptions that can happen in the lab due to the wrong settings of the microscopy image acquisition approach. The following data was considered:

- Figure 53 (a) Wrong fixation time of the cells in formalin. These cells are more turbulent and more strongly clumped together. This is a good example of an image that cannot be simulated artificially with image processing.
- Figure 53 (b) A real example of a defocus blur from the microscope, when microscope stage was lowered or elevated by 5 microns bringing the subject out of focus.
- Figure 53 (c) A real example of contrast / brightness differences due to wring exposure times.

The model tested here is the model trained on nucleus with simple augmentations (scale, rotation, flips). From Figure 53 it can be concluded that the time of formalin fixing does not really matter for nucleus outline, however the amount of details inside the nuclei seems

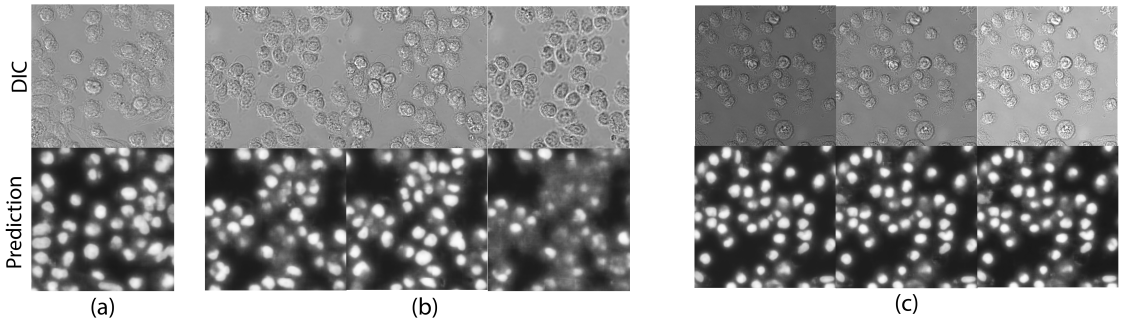


Figure 53: Real corrupted data from the lab. (a) — an example of the fixation of the cells in formalin for 15 minutes instead of the usual 10; (b) — the same image with different microscope focus adjusments. From left to right: microscope stage lowered by 5 microns bringing the subject out of focus, normal height, higher microscope stage by 5 microns bringing the subject out of focus again but in the other direction; (c) — the same image with different exposure times. From left to right: 20ms — underexposure, 30ms — normal setting, 40ms — overexposure.

to be slightly higher than with the usual fixation procedure. Focus of the microscope is clearly very important for good predictions. Although the negative direction of the focus loss seems to have better predictions than the positive one, many of the nuclei are simply gone in the fluorescence image in both cases. Over- and underexposure do not influence the predictions that drastically. Visually different exposure seems to change image brightness and the model is stable towards these changes. Nucleus outline stays mostly the same here as well, but the shining around it is dependent on the exposure times.

4.1.6 Improving predictions with additional corruption augmentations

Going from observations of the models' stability towards different image brightness which is present in the datasets a new hypothesis was drawn. Introducing corruptions that we test on into the training should improve the predictions on corrupted data. Unfortunately, it is not possible to use real lab corruptions here as the data was provided only for testing on these difficult cases and was not stained. Without staining one cannot give a quantitative measure of the quality of the predictions. However, artificial corruptions can be applied here easily. Random changes in contrast, brightness and defocus blur of severity levels -4 and 4 were added to the training augmentations. After the improved model was trained the predictions on the corrupted dataset became much better indeed (see Figure 54).

Using artificial corruptions in training augmentations positively influences the predictions on the images with real microscopy settings corruptions. See the comparison of the predictions of the model trained with (b) and without (a) artificial corruptions in augmentations on the corrupted dataset with real microscopy settings corruptions in Figure 55. For all types of the corruptions presented there it is clear that predictions the model

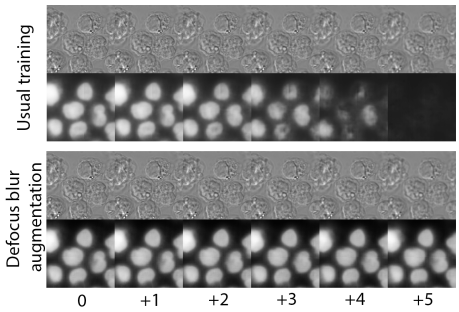


Figure 54: Using corruptions as augmentations to improve predictions: artificial defocus blur example

trained with artificial corruptions are better. The "shine" artifact around the cells is much less pronounced and for wrong exposure and fixation times corruptions there are more details inside the nuclei. In defocus blur corruption case it seems that the outlines of the nuclei are better preserved, however the intensities are still not captured for some of the cells at all. This shows that using corruptions as augmentations can strongly help the model to imrpove predictions for the corrupted images during inference. For the future research it is recommended to add real corruptions data along with the staining into the dataset.

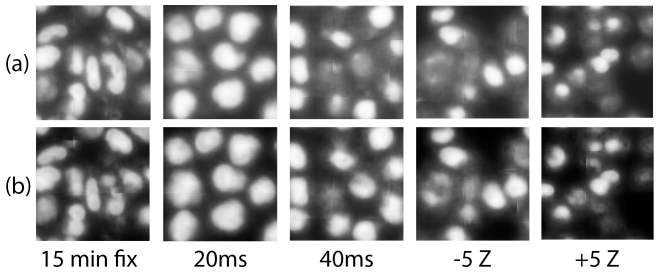


Figure 55: Using corruptions as augmentations to improve predictions: real corruptions example. (a) — model trained without artificial corruprions as augmentations; (b) — model trained with artificial corruptions as augmentations

4.1.7 Influence of corruptions on metrics for practical biological evaluation

Additionally, since for artificial corruptions the ground truth data from staining is present, the difference in biological metrics for models with and without augmentations was measured (see Table 8), more specifically Spearman rank correlarion coefficients were compared. The calculation of biological metrics remained the same except for the thresholding algorithm that was switched to a global one due to the time limit. This results in the wrong segmentation of some of the ground ground truth images due to the illumination inconsistences Therefore Spearman rank correlation coefficient is more repesentative here since it

is more stable towards the outliers. The rest of the postprocessing procedure has remained the same apart from the application of artificial corruptions on the input data.

Table 8: Correlation coefficients for practical biological evaluation on nuclei

Contrast level	Number of nuclei	Total intensity	Mean intensity	Area
+1	0.934	0.825	0.826	0.898
+2	0.932	0.820	0.819	0.899
+3	0.934	0.799	0.822	0.890
+4	0.804	0.439	0.671	0.540
+5	0.394	0.351	0.383	0.313
Defocus blur level	Number of nuclei	Total intensity	Mean intensity	Area
+1	0.934	0.832	0.827	0.905
+2	0.929	0.800	0.820	0.890
+3	0.934	0.756	0.8210	0.871
+4	0.838	0.361	0.666	0.501
+5	-0.072	-0.233	-0.231	0.07

One can observe from the table above that the metrics degrade quite fast starting from the severity level 3. Which aligns well with the visual evaluation of corruptions in Figure 51 where level 3 of contrast and defocus blur corruptions affect the intensities and level 4 affects the whole nuclei outline. Biological metrics confirm that the most affected metrics are the total and mean intensities, whereas the number of organelles seems to be the most stable metric until the very last severity level where the predictions turn almost completely black. Although these metrics are much more representative than PCC or MSE loss they also they their own downsides. For example, severity level 4 produces not reliable predictions, however correlation coefficient of mean intensity is not completely low — 0.67. Therefore it is important to assess predictions quality visually and keep in mind that biological metrics should be ideally quite high.

4.1.8 Generalizability across phenotypes

In order to evaluate how well a UNet model is able to generalize across different cell phenotypes the model was first trained on one cell phenotype only and then evaluated on the other cell phenotype. For this experiment CHOZN phenotype was chosen for model training with nuclei fluorescence target, whereas predictions evaluation was performed on PHX phenotype. The predictions were compared with the predictions of the model trained previously on both phenotypes. Comparison was performed both visually and via PCC for the biological metrics. The results in terms of the metrics clearly show the superiority of the model trained on both phenotypes, especially in terms of intensity predictions, where the PCC for the model trained on both phenotypes are almost 3% better. The postprocessing procedures for both models remained the same.

	CHOZN and PHX	CHOZN
Number of ER	0.985	0.985
Total intensity	0.716	0.680
Mean intensity	0.732	0.708
Area	0.986	0.960

Table 9: Generalizability across phenotypes for nuclei predictions. Comparison between the model trained on both phenotypes (CHOZN and PHX) and the model trained on CHOZN cells only in terms of bilogical metrics

From these results one can notice the general drop in performance on PHX phenotype even for the model that has had PHX cells in training (total intensity PCC for the whole dataset from Table 3 was 0.861 in comparsion to current 0.716). The reason for that is generally lower number of PHX samples in nuclei training dataset ($\sim 30\%$).

Nevertheless, the visual evaluation of the predictions shows little to no difference between predictive models with model trained on both phenotypes giving slightly more details inside the nuclei (see Figure 56). From this comparsion it was concluded that the model is able to generalize from CHOZN to PHX phentype, however in general PHX phenotype seems to be slightly more challenging for predictions than CHOZN.

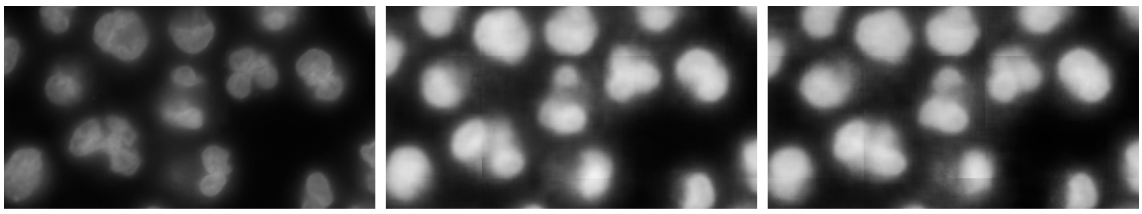


Figure 56: Visual evaluation of the UNet generalization capabilities. (a) — ground truth fluorescence of nuclei target of PHX cells, (b) — prediction of the model trained on both CHOZN and PHX cells, (c) — prediction of the model trained on PHX cells only

4.2 UNET embeddings

As well as one autoencoder embeddings that represent a high-dimensional input in lower dimensional space, UNet embeddings provide useful and interesting information on the predictions quality. However, it is important to keep in mind that the dimensionality of embeddings in the UNet case is not lower than the dimensionality of the input and is even often higher (see the UNet architecture in Figure 7). The goal of a UNet in contrast to an autoencoder is not to compress the input, but to extract useful features that are helpful for high-resolution segmentation. UNet embeddings do not contain rich image semantics in them as the embeddings of an autoencoder do. UNet compresses the spatial

dimension of the input, but at the sme time it gradually increases the number of filters that capture of information need for segmentation. As has been proven in Section 3.2.2.2 having more filters only helps to get better predictions, therefore there is no need for a UNet to have low-dimensional embeddings. Nevertheless, it is still interesting to see if the embeddings do contain any information about the input that one could use. There were two hypotheses put in question: the first one is whether embeddings of a trained UNet form any kind of clusters based on a cell's phenotype. The second hypothesis is whether embeddings of corrupted images can be clustered together further away from non-corrupted ones. If the latter hypothesis would hold true, one could alarm the end user about the outliers in the dataset based on their distance from both of the clusters.

4.2.1 Application of various dimensionality reduction methods

It is important that any image is fed into the network crop by crop, meaning that for each crop there is a separate embedding. In this section crops embeddings were not combined in any way together and were analysed separately.

The UNet embedding has a size of $16 \times 16 \times 256$ and can be flattened into a 655536-dimensional vector. In order to comprehend the embeddings for us as humans, a dimensionality reduction algorithm has to be applied. One option would be to compress a vector to 2D or 3D representation, which is easily comprehendable by humans.

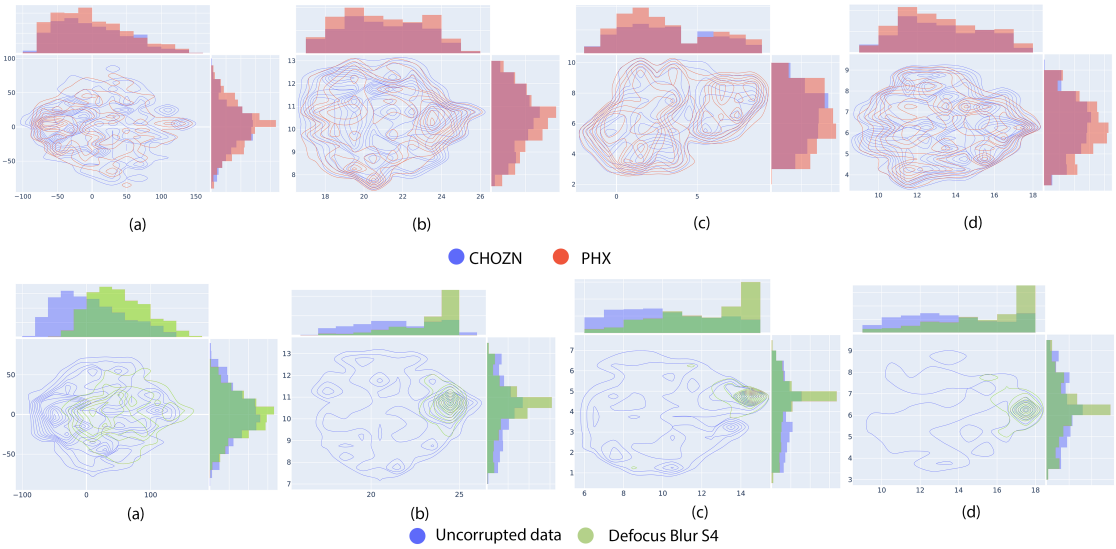


Figure 57: Visualization of UNet embeddings in 2D space. (a) PCA, (b) UMAP, (c) combination of PCA and UMAP with 10 and (d) 200 components. First row differentiates between two cell phenotypes: CHOZN and PHX, whereas the second row differentiates between uncorrupted crops and crops corrupted using artificial defocus blur of severity level 4.

In this case a two-dimensional representation was chosen. With the help of PCA, UMAP and their combination embeddings were projected into a 2D space. In Figure 57 we can see kernel density estimate (KDE) plots, that were created based on scatter plots, where

each dot represents a projected UNet embedding of a crop. Both research questions are addressed here: clustering based on phenotypes (CHOZN or PHX) and clustering based on input corruptions (defocus blur of severity level 4). It is crucial here that corrupted data was not used in training of any of dimensionality reduction method. The goal was to use only the data available in training dataset, find the transformation of high-dimensional data into a lower-dimensional space and apply it to new samples. That is also why methods like t-sne (Maaten et al., 2008) cannot be used here, because the transformation that t-sne learns cannot be applied to new samples.

From Figure 57 it becomes evident that there is no clustering based on the phenotype. On the one hand, this means that it is not possible to detect phenotype based on the UNet embedding. But on the other hand, this also implies that PHX or CHOZN phenotypes do not influence the predictions so much supports previous conclusion from Section 4.1.8 that the model generalises well across them. Regarding the clustering based on the artificial corruption it seems that embeddings of corrupted samples tend to clump more in groups, occupying one specific area of the embeddings space. It is also clear that the combination of UMAP with previously applied PCA works better with the increasing amount of components in PCA: dots in (d) seem to form a better cluster than dots in (c). However, it can still not be taken intuitively from this figure how many non-corrupted dots are hidden behind the cluster of the green dots — meaning whether non-corrupted crops cluster intersects severely with a corrupted one. In order to visualize this better, one can use a kernel density estimate (KDE) plot presented in Figure **??**. Additionally, it is clear that pure UMAP is not the best approach for the extreme number of dimensions as with the one in this case.

A cluster of corrupted images is clearly present here, however it also intersects with many non-corrupted crops. The quantitative evaluation of how this cluster is separable from the rest of the points is provided in section 4.2.1.1. Although one can already state that there is a clear opportunity to differentiate between corrupted and not corrupted images, the accuracy cannot be high due to the clusters being not well separable. For further research it is suggested to additionally check whether clusters form into a high-dimensional space before projecting them into a 2D space.

4.2.1.1 Clustering with PaCMAP

Since the UNet embeddings are the most promising for clustering based on input corruptions we will proceed with this approach. Apart from dimensionality reduction methods used in section 4.2.1, PaCMAP clustering was applied. Its detailed description can be found in section 2.2.2.3.

PaCMAP allows a more flexible hyperparameter tuning in order to preserve local and global relations from high-dimensional data and as a result a better clustering can be found. As usual PaCMAP was trained using "good" training data only, meaning no corruptions were introduced. And only when the transformation from high-dimensional into a lower-dimensional space was found, corrupted crops were projected using this transformation. Figure 58 presents results of training PaCMAP with four different hyperparametrization settings. There are three hyperparameters that were changed here: *MN_ratio*, *FP_ratio* and *n_neighbors*. A detailed explanation of their influence was also

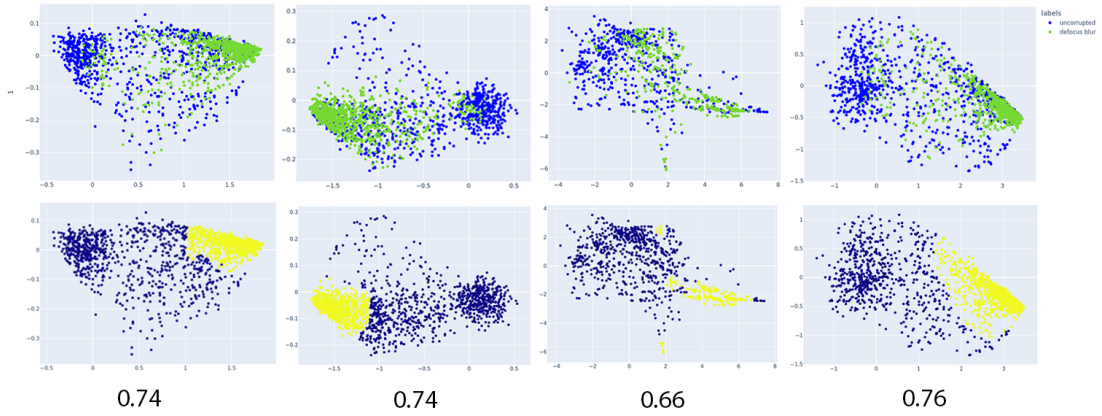


Figure 58: Clustering of UNet embeddings after PacMAP

MN_ratio	*FP_ratio*	*n_neighbors*
0.5	0.1	10
0.1	0.1	10
0.5	0.5	2
0.1	0.5	10

Table 10: PaCMAP hyperparameters

given in Section 2.2.2.3. From left to right in Figure 58 the hyperparameters were taken from Table 10:

In this case a cluster of green dots represents the projected UNet embeddings of images corrupted with defocus blur with severity level 4, which is already a strong corruption and leads to unacceptable predictions of the model, that did not have defocus blur augmentations. However, these points are still strongly mixed with non-corrupted ones. In order to check how separable they are unsupervised clustering DBSCAN algorithm was used. Results of this clustering are shown in Figure 59. The density based clustering approach utilized here was described in more details in section 2.2.3.1.

After training DBSCAN on non-corrupted crops embeddings it has recognized a class on the right (yellow dots) as a cluster and the rest of the points (blue ones) as noise, because they have quite low density in comparison to the yellow cluster. This is not a problem if we consider noisy points simply as a separate cluster. For such clustering defocus blur of level 4 splits the crops between two clusters with an F1-score of 0.74. In Figure 59 on the right red points represent projection of image embeddings after corrupting them with a defocus blur of severity level 3. In this case they are mixed with non-corrupted projections even stronger. Here prediction of already trained DBSCAN drops to F1-score of 0.64.

Overall UNet embeddings do express clustering of corrupted embeddings to a small extent, however not strongly enough to use it in practice. Here the autoencoder would be a more suitable approach, however brightness normalization has to take place first.

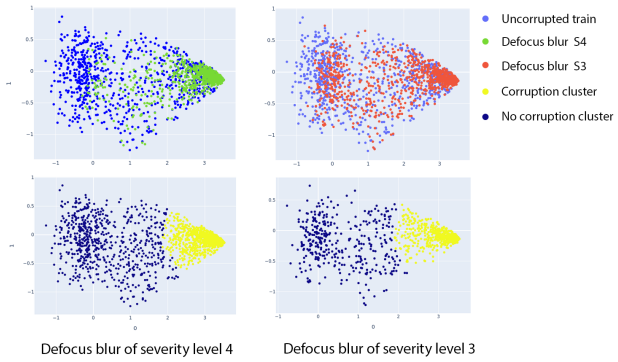


Figure 59: Clustering of UNet embeddings after PacMAP for different severities levels

4.2.2 Autoencoder embeddings as an alternative

Since UNet embeddings do not seem to exhibit any exceptional results in terms of clustering, it was decided to train an autoencoder directly on DIC image crops. Since the autoencoder's embeddings contain dense semantic information of the input they might provide more insights for clustering the previously mentioned hypotheses. Figure 60 presents the architecture of two convolutional autoencoders used for these experiments. One compresses 256×256 input crops into embeddings vector of size 3528 and another one compresses them into a vector of a smaller size of 200. Both autoencoders were trained using MSE loss. The results of their convergence are presented in Figure 60 on the right.

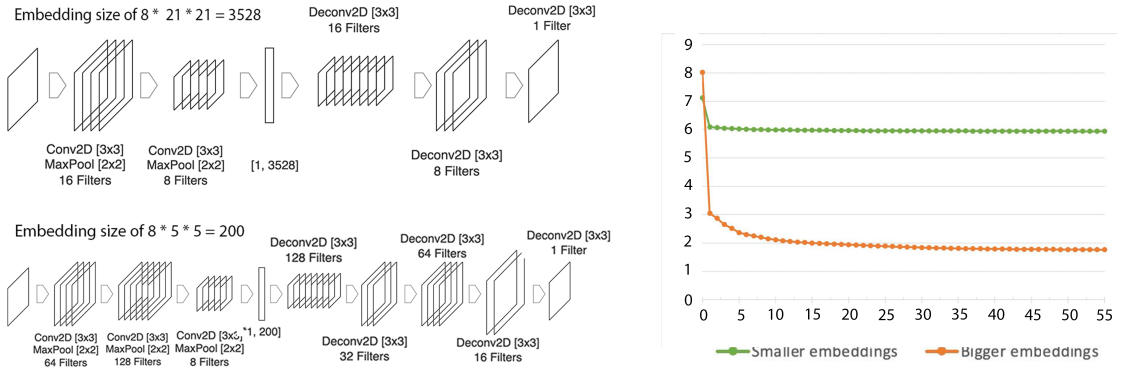


Figure 60: Architectures of two autoencoders and their training convergence

An autoencoder with embeddings of bigger size was able to achieve a lower loss as well and the samples reconstructed from it were of higher quality (see Figure 61). Clearly reconstruction of the samples will not have a high resolution as there are no skip-connections in this architecture. However, this is also not needed, the main goal here is to find out whether autoencoder embeddings provide any insights on the data.

Since an autoencoder with bigger embedding size seems to be able to reconstruct crops much better we have proceeded with its architecture. Embeddings were projected into a

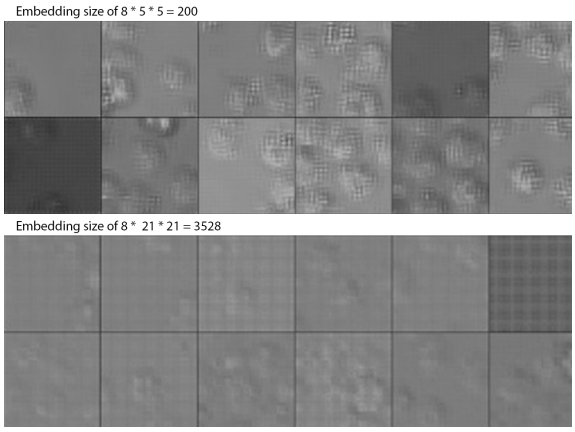


Figure 61: Samples drawn from trained autoencoders. (a) — an autoencoder with a smaller bottleneck layer, (b) — an autoencoder with a bigger bottleneck layer

two-dimensional space using first PCA with 10 components and then applying UMAP on PCA's projections. The results of such projection are presented in Figure 62. Two clearly defined clusters appear: the left plot presents projections from an earlier epoch, the right one from a later one. Embeddings separate gradually into two clusters throughout the training.

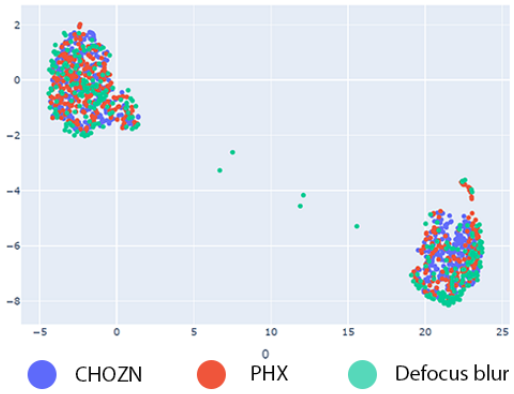


Figure 62: Autoencoder embeddings after applying PCA and UMAP afterwards

However, these two clusters are based neither on cell phenotypes nor on input corruption. All points of both phenotype as well as corruptions seem to be equally spread between two clusters. By looking at the images corresponding to each of the clusters it soon becomes apparent that the main difference between them is their brightness level. To prove this theory distributions of average image intensity of images in both clusters are presented in Figure 63. In the violin plots we can see that the distribution of the crops on the left has a much lower brightness level than the distribution of the crops on the right. However, it is hard to account for why the brightness were different, it might happed due to many different reasons, for example, the images might have been taken on different days when

the microscopy lighting conditions might have been different.

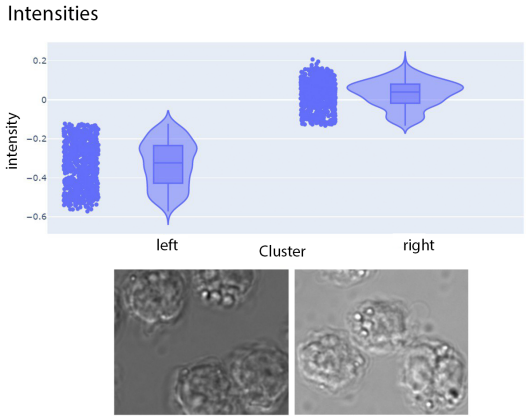


Figure 63: Different in brightness in the clusters formed by an autoencoder embeddings in a two-dimensional space

Since an autoencoder picks up on brightness difference within the crops, it is worth trying to normalize crop brightness across the entire dataset first. Nevertheless, it is not a trivial task as images have different cell density in them. This is why some images that contain primarily background pixels will always be darker than the ones that contain enough of foreground. We suggest to filter the crops based on the amount of cell criteria (which can be done using GFP model that can detect cells present in DIC) and normalize them afterwards. Retraining the autoencoder with new training data might provide more insights when the difference in brightness is gone.

It is also clear why autoencoder embeddings do not provide any clustering for corrupted crops. Corruption severities neither really change the image semantics nor are they significantly different visually speaking (see defocus blur in 51). Therefore they do not alter the ability of an autoencoder to restore input correctly. In contrast, UNet's fluorescence predictions do suffer significantly for several corruption levels, its predictions strongly change — the outline of organelles becomes more blurry, additional shine appears in fluorescence prediction. These changes happen not only during the decoding part, but they also might bring unsual values in the embedding representation. Therefore UNet embeddings have more information on the "trustworthiness" of predictions. That is why when defocus corruptions are used as training augmentations, drift detection for the model trained with these corruptions stops alarming about the drift. Even though it did for the model, which did not have these augmentations present [TODO add section reference]. This happens simply because the models' predictions degrade and start looking different, which triggers a "drift alarm". With the improved predictions, drift alarm would not be triggered even when using the same data.

4.3 Drift detection

Following the development phase, when the model training is finished, the model will be moved into deployment or production, where it is supposed to maintain an expected quality of predictions. However, input data is not always a stable source of input. One should constantly maintain quality of predictions and do regular check-ups for outliers as well as to alert the end user about a drift in input data. Drift detection happens on raw data in absence of the ground truth labels and serves as a signal that the input data differs a lot from the data used for training, meaning that predictions became unreliable.

There is a significant difference between distinguishing drift of the whole source of data in comparison to detecting single outliers. In drift detection, one looks at the whole new input data as a distribution and checks if there is a significant shift in comparison to the data used during training. To compare original training data distribution with the new one from inputs different statistical tests like Kolmogorov-Smirnov, Chi-squared and others can used.

There are two possible reactions after the drift is detected: alert the user that predictions became unreliable, and and therefore the expansion of the dataset should be considered by adding more labeled data from a newly drifted distribution in training, or applying some different logic on the model outputs. When an outlier is detected, a model might request human assistance for some particular input, because this input is too unfamiliar to the model and possibly it will not return good predictions on this one (Samuylova, 2021).

In summary, the drift detection is needed only when the meaningful shifts of the input data distribution from the training distribution need to be detected. Which is exactly the case in this project. During production there can be various shifts in image acquisition processes due to the different microscopy settings and it is beneficial to be able to detect them. The models are trained assuming the correct setup of microscopy image acquisition, however changes in exposure, illumination, cell fixation procedure might alter DIC imaging. In this case the user has to be informed about it and choose afterwards whether more data should be added to the training set or whether the mode's predictions should not be used.

It is also worth noticing that since the images are split into crops and fed into the drift detection model by parts, several crops from image can already be representative of the new image distribution. Therefore if the drift detection model is fast enough to detect drift based on the few crops only, the detection will happen after the crops collected from one image only, such a model can be used as an outlier detector.

4.3.1 Drift detection experiments

Drift detection of corrupted samples for the problem of the thesis has been performed using *alibi-detect* open source python library, that focuses specifically on outlier, adversarial and drift detection algorithms (SeldonIO, 2022). This library implements statistical hypothesis testing algorithms for detecting drifts in data.

It works the following way, before observing some data, one can specify null-hypothesis

H_0 and alternative hypothesis H_1 about generating process behind the data (its distribution for example) and also specify the test statistics $S(X)$ that are expected to be small under hypothesis H_0 and large under hypothesis H_1. Then during the observation of the new data test statistic value $S(X)$ is computed along with a probability $p = P(S(X)|H_0)$ which is called p-value. P-value is a probability that such an extreme or a more extreme value of test statistic could have been observed under the null-hypothesis. If this probability is below the established threshold t, then one can assume that data is drifted. If p-value is low, null-hypothesis will be refused. In this case a p-value threshold was chosen to be 0.009.

A drift detection algorithm was trained on UNet embeddings of not corrupted train data. For this experiment $10,000$ embeddings were used of CHZN and PHX phenotypes from the nucleus dataset. It is important that the crops were chosen in such a way that at least several cells present are present there. After splitting images into the crops many of them contain a primarily background with few cells present. By filtering out these crops one can make sure that drift detection model actually learns on the important foreground signal from the cells rather than on the predictions of the background. After the drift detection algorithm was trained, it was tested on the dataset, that was not seen by a model beforehand (test dataset). Test dataset consists of 119 images, where from each image 5 random crops were chosen. The crops for each image were chosen again in the same way as fro training by only choosing the ones that have enough cells present in them. Since images have a high resolution, one can assume that one image itself represents a new input distribution, where crops taken from this image are its samples. Therefore we can detect whether one specific image has drifted or not feeding the crops from it into a drift detection algorithm. First, the algorithm was tested on not drifted data by using a test set of nucleus dataset. Out of 119 images 8 were recognized as drifted ones. This means that the algorithm's false positive rate is approximately $\frac{8}{119} \approx 0.063$.

Below the results of the trained drift detector for two datasets are presented: same test data with artificial corruptions applied to it and on data with real microscopy corruptions.

Artificial corruptions

Figure 64 presents the results of drift detection for all artificial corruptions, more specifically the algorithm's false negative rate. One can see that the lower the severity of a corruption is, the higher the false negative rate becomes. When the corruption severity level is low the predictions remain to have a high quality (see Figure 51), therefore an end user can still rely on the UNet. However, the stronger the corruption is, the stronger fluorescence prediction degenerates and as a result a drift detector alerts a user to the presence of drift, which is a useful quiality of this algorithm. Drift detector is more sensitive towards contrast changes rather then towards defocus blur changes. It is the most sensitive towards brightness corruption.

Real corruptions

Types of real corruptions tested here are described in more details in Section 4.1.5. Two phenotypes are present there: PHX (was also present in the training dataset) and 2e3 (was not present in any of the datasets before). Since these two subsets of data look very much alike drift detection results on both of them will be combined. In Table 11 the results of the drift detection algorithm are presented. Additionally, for few samples of not drifted data

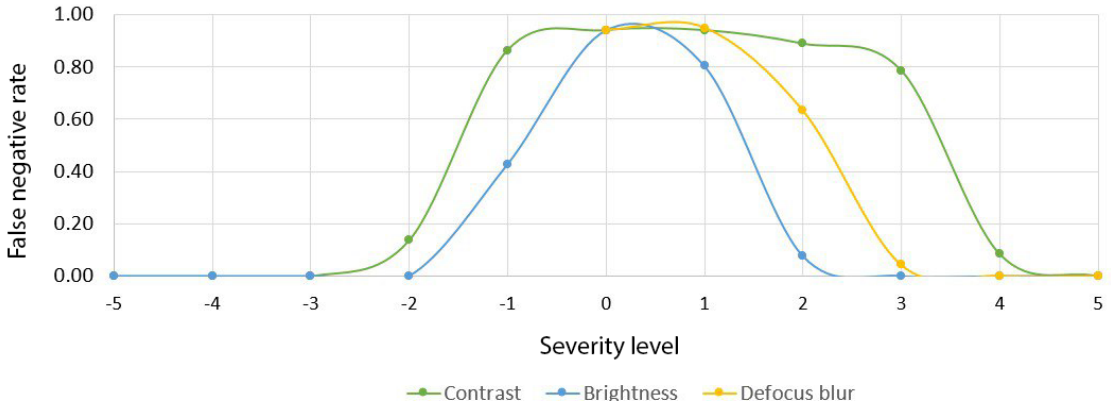


Figure 64: False negatives rate for drift detection on artificial corruptions

that were also included in this dataset, namely 2 images of the correct focus distance Z and 4 images of the correct exposure time (30ms), the detector falsely alerted 0 and 1 samples correspondingly. The results confirm that the detector is trained well enough to be able to detect drift to some extent, however not all drifted images will be noticed by it. In order to state how exactly accurate it for real corruptions is much more data would be needed. Assuming that uncorrupted test dataset is representative enough, the low false positive rate is expected (around 0.063). That is why one can presumably rely on the detector to alert the user about wrong exposures or focus corruptions. Regarding wrong fixation time of the cells, on the one hand it seems that this corruption has somewhat low detection rate, on the other one model's predictions are actually of quite a good quality. Therefore it might be the model is generally quite stable towards the prolonged fixation time.

Table 11: Drift detection for real microscopy corruptions

	15 minute fixation	-5 Z	+5 Z	20 ms exposure	40 ms exposure
Detections	3	2	3	2	2
Total images	8	4	4	4	4

4.3.2 Online drift detection experiments

MMD algorithm in *alibi-detect* library has an online version, which also presents an interest for our case. One of the main differences of an online drift detector is that it does not accept several inputs at once alerting whether they represent a drift all together, but it focuses on a single input at a time during its run. It assumes that there is a big dataset of reference data, that can be used as an example of a "correct" distribution and processes one embedding at a time. This single input would be sent into a test window where two-sample test-statistics (MMD essentially in this case) will be calculated. As soon as the test-statistic exceeds some pre-defined threshold a drift alert is send to the user. Apart from the threshold one has to define a so-called expected run-time (ERT). This time states how many inputs the detector should process on average before it makes a detection (false

positive or true positive depending on which distributions the inputs were taken from). Another hyperparameter here is a size of a test-window. Because the larger the window is, the more chance there is to detect a very slight drift, yet with a smaller window one gets a much faster response to severe drift.

It is usually recommened to reduce the dimensionality of the data before feeding it into the algorithm. But in this case the performance was exceptional even for high-dimensional embeddings of the UNET model. Nevertheless, there are also options on how to reduce the dimensionality of the data from the authors of the framework.

This algorithm was trained on the same uncorrupted training data sent through the embeddings of a nucleus UNet. Yet, test crops are fed into a detector in a consecutive way — in case of no corruptions present it is not enough to feed only several crops from one image, as the algorithm requires more data to be fed into the detector until it makes a decision. For example, ERT of not corrupted inputs can reach a value up to 180 crops until a detector alerts a false positive detection. On the contrary for corrupted inputs drift detector needs normally ≤ 6 crops (see Figure 65)

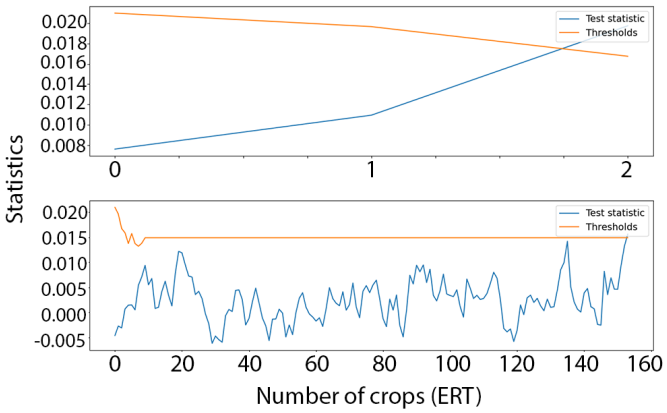


Figure 65: Expected runtime (ERT) for corrupted and in-distribution data

Having such a detector one measures ERTs for not corrupted inputs, then for corrupted ones and afterwards calculates the best threshold that can separate the two. ERTs of corrupted data should be much lower than ERTs of not corrupted one. In this experiment ERTs of both classes were separable, however both located very close to the threshold: ≈ 4.59 for corrupted ones and ≈ 7.1 for not corrupted. Yet with a threshold of 6 the accuracy scores are very high. Having a drift detector trained on not corrupted training data only, one can estimate ROC-AUC scores between two classes: not corrupted test data and the same test data but with some corruption applies to it. The results of this experiment with a defocus blur corruption of several severities are presented in Figure 66. Already on severity 3, such detector separates the two almost perfectly. The scores are also presented in Table 12.

Additionally, research has been performed on the influence of the hyperparameters to an online MMD drift detector, specifically: test window size, specified ERT. The results are presented in Tables 13, 14. Test window size influences how fast and how sensitive

Figure 66: AUC ROC scores for various defocus corruption severities

Table 12: Severity of artificial defocus blur on separability

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

the reaction of an algorithm should be and the ideal value in this case would be 10 crops, during this time a threshold will be established. In case of ERT as a hyperparameter as long as it is big enough the score does not change very much.

Table 13: Test window size influence on separability (artificial defocus blur)

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

Table 14: ERT influence on separability (artificial defocus blur)

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

4.3.2.1 Impact of cell fixation

In section 3.5 the difference between fixed and not fixed cells was mentioned. Visual analysis of model's predictions for not fixed cells after training it on fixed ones has shown that the model was not able to generalize well on them. This is the reason why it would be important to alarm the end user to not rely on predictions when such a situation occurs.

In this case an online drift detector trained using not corrupted data used for ER training first and tested on not fixed ER cells. The results of this test are shown in Figure 67. The

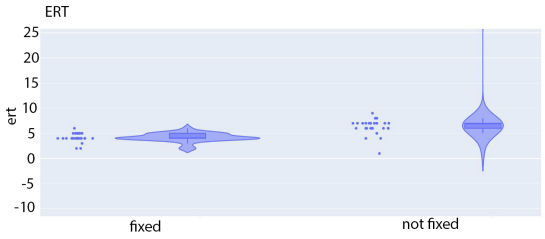


Figure 67: Online drift detection of not fixated cells

ERTs for corrupted data (left) are lower from ERT for true input. The ROC-AUC score for the separability is 0.91 and the best threshold is 6. However, not corrupted data (fixed cells) mostly have an ERT of 7, whereas corrupted data (not fixed cells) have an ERT of 4. Both classes have ERTs that are very close to the threshold, but are able to separate the classes well enough.

Application of a usual drift detection algorithm with the use of ER model the false positive rate on not corrupted (fixed) cells was 0.075. Whereas all fixed cells were recognized as drift.

4.4 Summary

Often appearing during inference DIC image corruptions were analysed with the use of artificial reproducible pseudocorruptions along with the real corruptions acquired from the wrong microscopy settings. Model's predictions degrade with the increasing severity of corruptions, however the use of artificial corruptions in traning mitigates this problem for both real and artificical DIC corruptions. Corruptions influence biological metrics as well, especially total and mean intensities are the two most sensitive metrics. This is directly related to the loss of details in predictions present and therefore it is recommended to pay more attention to intensity metrics during models evaluation. Models are well generalizable from CHOZN to PHX phenotype. Analysis of the lower representation of DIC imaging was carried out through the visualization of UNet embeddings and autoencoder embeddings trained on original DIC data. UNet embeddings reflect the influence of corruptions, however not strongly enough to be used for clustering, whereas autoencoder embeddings do not capture artificial corruptions completely. Two drift detection algorithm were built that are able to successfully differentiate between corrupted and non-corrupted input DIC data based on the UNet embeddings. Both online and offline drift detection algorithms have successfully separated drifted samples from not drifted ones. A usual drift detection algorithm is recommended for a use in the future due to the lower separability between corrupted and not corrupted classes in online version. Tests performed on real microscopic corruptions show that the algorithm is able to detect drift in the real data too, however to evalute the accuracy metrics on these cases more data

is needed. Successfull results on the use of drift detector on artificial corruptions shows a great promise for its practical use in order to alert end users when models predictions become unreliable.

5 Summary

5.1 Results

- In the scope of this master thesis three UNet models were developed that are able to predict fluorescence signal from DIC image data for the following targets: nuclei, endoplasmic reticulum and full cell surface. It was shown that developed models can successfully replace a manual fluorescence staining procedure as they are visually as well as based on practical biological metrics similar to the ground truth fluorescence images.
- Several adjustments before running any longer training experiment to model's training pipeline were made. First, the model's architecture from LaChance et al., 2020 that was used in the experiments was improved in terms of faster learning with the use of batch normalization layers. Second, additional image augmentations like rotations and scaling were added and their logic was improved through the use of the high-resolution original image from the standard one. Third, the importance of a correct weight initialization was presented. And lastly, the model was tuned by choosing the correct regularization and optimization algorithms (batch normalization and adadelta optimization in this case).
- For each of the four targets corresponding image preprocessing pipelines were developed. This step was very important to improve training data quality and address models' limitations before training, such as background-foreground class imbalance for instance. In the the Golgi apparatus preprocessing pipeline advanced methods like background removal using a rolling ball algorithm and image enhancement were introduced.
- In each training procedure, the model has successfully converged for all fluorescence targets except for the Golgi apparatus. Training experiments for the Golgi target showed that there is a lack of training data with high enough signal-to-noise ratio. Therefore further research can be carried out once the data with an improved signal-to-noise ratio has been collected, for example after a use of a better antibody for staining. However, several attems to improve model's predictions for the Golgi apparatus target were made via the use asymmentrical losses and advanced image preprocessing techniques. State-of-the-art papers like Cheng et al., 2021 also demonstrate how challenging the Golgi predictions can be and therefore an accurate model for the Golgi apparatus predictions remain an open research challenge.
- The models' limitations in terms of intensity predictions were resolved. Based on visualizations and biological metrics estimations it was derived that the models have a similar downside in overpredicting total and mean intensities. Nevertheless, a strong correlation between the values suggests that there is an absolute value shift in predictions that can be fixed with relative ease. This issue was solved with the use of a bigger model and more data respectively. An immediate improvement was

achieved when it comes to the correlation coefficients between the prediction and ground truth in the total and mean intensities.

- The models were evaluated against a set of common practical biological metrics including organelle quantity, total intensity, mean intensity and area of the organelle of interest. Every model was evaluated based on these metrics. It was shown that not only a significant correlation with ground truth values in terms of Pearson and Spearman rank correlation coefficients is present, but that the forms of distributions visualized with violin plots are very similar.

- For each organelle in question corresponding segmentation pipelines were developed, since the evaluation of the models on biological metrics require the segmentation of each predicted organelle. Postprocessing procedures for segmentation were successfully built with the OpenCV (Bradski, 2000) and *skimage* (Van der Walt et al., 2014) libraries.

- Generalizability study of the models across cell phenotypes and across not fixed cells was carried out. Regardless on which cell size was the model trained, it was able to generalize across different cell scales if the difference is under 30%. However, biological metrics show that intensities can be better reproduced when training on bigger cells. Lastly, an inductive bias of a model for full cell surface was determined: the model is able to select all the cells in the foreground regardless their state (both live and dead).

- Apart from generalizability models were tested on the stability of their predictions. For that goal artificially corrupted via image processing (like brightness, contrast changes and defocus blur imitations) data and data corrupted from faulty microscopy settings were introduced. It was shown that the models are very stable against changes in contrast and brightness, which can be caused by an over- or underexposure. Even though the models were very prone to errors with defocus blur corruption, using artificial corruptions as training augmentations improves models performance on corrupted images. Prediction on the image with corruptions created in the laboratory have shown that the models are stable against errors in cell fixation process, however they are quite sensitive towards errors in the focus of the microscope.

- UNet embeddings were studied for the possible source of additional data insights like phenotype differences or corruptions and unsupervised clustering algorithms were applied. The study has shown that image embeddings are not clustered based on cell phenotype, however corrupted images indeed form a separate cluster in the embeddings space. Nevertheless, this cluster is not well-separable from the rest of the data and further research is required. In an attempt to find a lower dimensional embedding for image representation an autoencoder embedding that was trained. The training has been carried out using the same data and the embeddings were checked for the same clustering targets. The results have shown strong clustering based on the brightness of the crops, which is not significant for this research as the brightness changes are very typical in DIC imaging due to slight differences in the exposure times and can be detecte with far easier methods.

- Finally, In order to detect corruptions and determine whether a models' prediction should or should not be reliable, two drift detection algorithms were developed. The first one is based on the maximum mean discrepancy method, and the second one is an online version of the first one. They were tested on both corruption types and have shown strong ability for detecting drift in data with high ROC-AUC scores. These algorithms performed much better than the clustering approach mentioned above and can be used in practice.

5.2 Limitations

The main limitation of this research is the need to fix the cells before taking DIC images of them. Cell fixation is a preceding step before cell staining — therefore all cells in datasets of DIC images were fixed. Since living and fixed cells look very different and models trained on fixed cells do not generalize to non-fixed ones well, predictions can be carried out on DIC images of fixed cells only. Luckily, fixing procedure is not a cumbersome lab procedure and is far easier than staining the cells, which is avoided with the help of *in-silico* fluorescence labeling. It is recommended to look into possibilities of transfer learning from fixed to living cells. Also, models developed here cannot generalize well on other cells apart from CHO and are able to produce accurate predictions for the cells used in this laboratory only. However, in future the developed product would aim to address the specific needs of each laboratory separately. Therefore the models will be trained with the goal to address the issues of one specific cell line.

5.3 Future research

This research sets the ground for a variety of further research ideas:

- As the Golgi apparatus model did not produce good enough predictions due to several reasons (such as low signal-to-noise ratio in input data, as well as an extremely strong class imbalance between the foreground and background in images), it is strongly recommended to continue research in this direction. For example, by choosing another target protein in the staining procedure, applying stronger noise reduction approaches as introduced by Krull et al., 2019, and incorporating image gradients in the loss function.
- UNet embeddings do indeed show a potential for detecting corruptions in crops, however it is not strong enough. It is highly recommended to incorporate embeddings of all crops from the same image into one point in the embedding space. For example, by averaging the embeddings. This might show a more significant clustering of corrupted data.
- Since autoencoder embeddings have shown clear clustering based on the crop brightness, it would be beneficial to normalize the brightness across all crops first.

This would allow an autoencoder to pay attention to other less distinctive features in the image. However, due to the non-uniform distribution of the cells in the image this approach not the most promising one.

- Drift detction can also be implemented based on raw DIC input without involving model's predictions at all. Therefore it is recommended to further drive research in the direction of contrastive learning algorithms for out of distribution detection, specifically following the approach proposed by Tack et al., 2020.
- Due to time constraints it was not possible to train very big models, however the improvement of prediction quality that occurs after a model enlargement is very significant. Therefore if the higher image resolution is needed, it is recommended to train models with a bigger model size and more data.
- The lack of data concerning corruptions from the microscopy settings did not allow to test online drift detection algorithm. As this version very promising for practical use it is recommended to use more data for its testing.
- Since the model for full cell surface predictions is not able to distinguish between dead and live cells, it is recommended to add more images that include dead cells into the dataset or to train an alternative model that is able to distinguish them (Ounkomol et al., 2018).
- Although staining several organelles can be challenging due to the chemical reactions between fluorophores, staining multiple organelles at the same time would provide a possibility to train the model to predict several organelles at the same time. while sharing mutual information between them.

Advances of AI and deep learning are a promising direction for the future of the field of cell line development. This work serves as evidence of the progress allowing to reduction time and cost of manual work in biological laboratories. The automation of one step of the cell line development was shown in this research, but there are many more prospects regarding the use of deep learning for the development of more automated cell line development systems. Specifically, for predicting cell stability and productivity, which is the general goal of the Merck KGaA's project, within which this study was carried out. The use of the models developed here can potentially bring useful feature representations that can be incorporated into the pipeline for productivity predictions and save reduce the time spent on manual work even more.

A Appendix

A.1 Folder structure

The imaging data always comes in pairs (DIC + fluorescence imaging) and has the following structure:

```
dataset0
├── DIC
│   ├── {CHOZN, PHX, H19}_XY001.tif
│   ├── ...
│   └── {CHOZN, PHX, H19}_XY100.tif
└── {DAPI_nuclei, FITC, FITC_er, CY5_golgi}
    ├── {CHOZN, PHX, H19}_XY001.tif
    ├── ...
    └── {CHOZN, PHX, H19}_XY100.tif
...
datasetN
└── ...
```

Here a dataset$\{0, \ldots, N\}$ is one 96-well plate. Each dataset includes two subfolders — namely, a DIC and a fluorescence subfolder. The name of the latter one corresponds to the name of the fluorescent binding molecule as they are different for each target within the cell. Each filename within the subfolder includes its index number between $\{1, \ldots, 100\}$ and the phenotype of the cell.

A.2 Training costs estimation

For training purposes in this research cloud computing services were used, or more specifically - Amazon Web Services (AWS). These remotely located servers provide a possibility to train your models on a variety of graphics cards while paying a minute rate. In order to keep costs under control it was important to estimate the time needed for training in advance to choose the most efficient graphics processing unit (GPU) possible.

Cost estimation is additionally important for the model inference as well as during production. Although for production purposes the lambda functions from AWS can used. They can be triggered only when the inference request arrives and can be turned off automatically shortly after.

For both training and inference purposes two GPU models were tested g3-4xlarge (NVIDIA Tesla M60) and p3-2xlarge (NVIDIA Tesla V100). The datasets on which the experiments were performed are the full cell target training and validation datasets. The resulting costs are presented in the Tables 15, 16.

	Model	Runtime (1 epoch)	Dataset size	Cost per minute	Cost per epoch
g3.4xlarge	usual	6.5min	18,432	0.07125$	0.3$
p3.2xlarge	usual	57sec	18,432	0.1911$	0.18$
p3.2xlarge	bigger	4.7min	18,432	0.1911$	1.47$

Table 15: Cost estimation of AWS use for training models. Prices were retrieved on 01.06.2022

	Model	Runtime	Dataset size	Cost per minute	Cost of inference
g3.4xlarge	usual	42sec	7,616	0.0713	0.050$
p3.2xlarge	usual	13sec	7,616	0.1911	0.040$
p3.2xlarge	bigger	39sec	7,616	0.1911	0.124$

Table 16: Cost estimation of AWS use for inference purposes. Prices were retrieved on 01.06.2022

In conclusion, even though a p3 instance seems to be much more expensive, it is also much more efficient and the cost estimated for training times are significantly lower. That is why all the training experiments in this research were conducted on an NVIDIA Tesla V100 GPU.

References

Josh Albrecht Abe Fetterman (2020). *Understanding self-supervised and contrastive learning with Bootstrap your own latent (BYOL)*. https://generallyintelligent.ai/blog/2020-08-24-understanding-self-supervised-contrastive-learning. Online; accessed 01-June-2022.

Adaptive thresholding (2022). http://man.hubwiz.com/docset/Scikit-image.docset/Contents/Resources/Documents/auto_examples/segmentation/plot_threshold_adaptive.html. Online; accessed 01-June-2022.

Jody Barbeau (Sept. 2018). *Introduction to recombinant proteins*. https://blog.crownbio.com/overview-recombinant-proteins. Online; accessed 01-June-2022.

Alain Berlinet and Christine Thomas-Agnan (Jan. 2004). *Reproducing Kernel Hilbert Space in Probability and Statistics*.

Peter Eastman Bharath Ramsundar (2019). *Deep Learning for the Life Sciences*. https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837. O'Reilly Media.

Christopher M. Bishop (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag.

F Borek (Oct. 1984). "Immunofluorescence in medical science". en. In: *J. Immunol. Methods* 73.1, p. 232.

Karsten M. Borgwardt, Arthur Gretton, Malte J. Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J. Smola (July 2006). "Integrating structured biological data by Kernel Maximum Mean Discrepancy". In: *Bioinformatics* 22.14, e49–e57. eprint: https://academic.oup.com/bioinformatics/article-pdf/22/14/e49/616383/btl242.pdf.

Nada N. Boustany, Stephen A. Boppart, and Vadim Backman (July 2010). "Microscopic Imaging and Spectroscopy with Scattered Light". In: *Annual Review of Biomedical Engineering* 12.1, pp. 285–314.

G. Bradski (2000). "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools*.

Yaron Bromberg, Ori Katz, and Yaron Silberberg (May 2009). "Ghost imaging with a single detector". In: *Phys. Rev. A* 79 (5), p. 053840.

Richard W. Burry (Jan. 2011). "Controls for Immunocytochemistry". In: *Journal of Histochemistry and Cytochemistry* 59.1, pp. 6–12.

Calculating the output size of convolutions and transpose convolutions (Feb. 2022).

Andreas Castan, Patrick Schulz, Till Wenger, and Simon Fischer (2018). "Cell Line Development". In: *Biopharmaceutical Processing*. Elsevier, pp. 131–146.

Cell Line Development Services (2022). https://pharma.lonza.com/offerings/mammalian/cell-line-development. Online; accessed 01-June-2022.

Shiyi Cheng, Sipei Fu, Yumi Mun Kim, Weiye Song, Yunzhe Li, Yujia Xue, Ji Yi, and Lei Tian (Jan. 2021). "Single-cell cytometry via multiplexed fluorescence prediction by label-free reflectance microscopy". In: *Science Advances* 7.3.

CHOZN Platform — Technical Bulletin (2022). https://www.sigmaaldrich.com/deepweb/assets/sigmaaldrich/product/documents/245/459/chozn-platform-technical-bulletin.pdf.

Eric M. Christiansen, Samuel J. Yang, D. Michael Ando, Ashkan Javaherian, Gaia Skibinski, Scott Lipnick, Elliot Mount, Alison O'Neil, Kevan Shah, Alicia K. Lee, Piyush Goyal, William Fedus, Ryan Poplin, Andre Esteva, Marc Berndl, Lee L. Rubin, Philip Nelson, and Steven Finkbeiner (Apr. 2018). "In Silico Labeling: Predicting Fluorescent Labels in Unlabeled Images". In: *Cell* 173.3, 792–803.e19.

Marc P. Deisenroth, A. Aldo Faisal, and Cheng Soon Ong (2020). *Mathematics for Machine Learning*. Cambridge University Press.

Ema (May 2020). *Ich Q5D derivation and characterisation of cell substrates used for production biotechnological/biological products*.

Endoplasmic Reticulum (smooth) (2022). https://www.genome.gov/genetics-glossary/Endoplasmic-Reticulum-Smooth. Online; accessed 01-June-2022.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu (1996). "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. AAAI Press, pp. 226–231.

Fluorescent labeling (2022). https://www.nature.com/subjects/fluorescent-labelling.

Jerrold Fried, Jeffrey Doblin, Shigeru Takamoto, Amaury Perez, Herbert Hansen, and Bayard Clarkson (July 1982). "Effects of hoechst 33342 on survival and growth of two tumor cell lines and on hematopoietically normal bone marrow cells". In: *Cytometry* 3.1, pp. 42–47.

Ada Funaro, AL Horenstein, and Fabio Malavasi (1996). "Monoclonal antibodies in clinical applications." In.

Golgi body (2022). https://www.genome.gov/genetics-glossary/golgi-body. Online; accessed 01-June-2022.

Rafael C. Gonzalez and Richard E. Woods (2006). *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Stephen Hanson and Lorien Pratt (Jan. 1988). "Comparing Biases for Minimal Network Construction with Back-Propagation." In: pp. 177–185.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2015). *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*.

Jong Kwang Hong, Meiyappan Lakshmanan, Chetan Goudar, and Dong-Yup Lee (Dec. 2018). "Towards next generation CHO cell line development and engineering by systems approaches". In: *Current Opinion in Chemical Engineering* 22, pp. 1–10.

H. Hotelling (1933). "Analysis of a complex of statistical variables into principal components." In: *Journal of Educational Psychology* 24.6, pp. 417–441.

Sergey Ioffe and Christian Szegedy (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*.

Karthik Jayapal, K.F. Wlaschin, W.S. Hu, and M.G.S. Yap (Oct. 2007). "Recombinant Protein Therapeutics from CHO Cells - 20 Years and Counting". In: *Chemical Engineering Progress* 103, pp. 40–47.

Roy Jefferis (2017). "Recombinant Proteins and Monoclonal Antibodies". In: *Advances in Glycobiotechnology*. Springer International Publishing, pp. 281–318.

Steven C. L. Ho Jessna H. M. Yeo (Oct. 2017). "Optimized Selection Marker and CHO Host Cell Combinations for Generating High Monoclonal Antibody Producing Cell Lines". In: *Biotechnology Journal* 12.12, p. 1700175.

Rogers Kara (2022). *Endoplasmic Reticulum*. https://www.britannica.com/science/endoplasmic-reticulum. Online; accessed 01-June-2022.

Richard Kasprowicz, Rakesh Suman, and Peter O'Toole (Mar. 2017). "Characterising live cell behaviour: Traditional label-free and quantitative phase imaging approaches". In: *The International Journal of Biochemistry and Cell Biology* 84, pp. 89–95.

Jee Kim, Yeon-Gu Kim, and Gyun Lee (Dec. 2011). "CHO cells in biotechnology for production of recombinant proteins: Current state and further potential". In: *Applied microbiology and biotechnology* 93, pp. 917–30.

Alexander Krull, Tim-Oliver Buchholz, and Florian Jug (June 2019). "Noise2Void - learning denoising from single noisy images". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Siddharth Krishna Kumar (2017). *On weight initialization in deep neural networks*.

Julienne LaChance and Daniel J. Cohen (Dec. 2020). "Practical fluorescence reconstruction microscopy for large samples and low-magnification imaging". In: *PLOS Computational Biology* 16.12, pp. 1–24.

Marie-Eve Lalonde and Yves Durocher (June 2017). "Therapeutic glycoprotein production in mammalian cells". In: *Journal of Biotechnology* 251, pp. 128–140.

Hui-Ning Liu, Wei-Hua Dong, Yan Lin, Zhao-Hui Zhang, and Tian-Yun Wang (2022). "The Effect of microRNA on the Production of Recombinant Protein in CHO Cells and its Mechanism". In: *Frontiers in Bioengineering and Biotechnology* 10.

Laurens van der Maaten and Geoffrey Hinton (2008). "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9.86, pp. 2579–2605.

Leland McInnes and John Healy (Feb. 2018). "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction". In.

Tom Michael Mitchell (1997). *Machine learning*. McGraw-Hill.

Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, and Bernhard Schölkopf (2017). "Kernel Mean Embedding of Distributions: A Review and Beyond". In: *Foundations and Trends in Machine Learning* 10.1-2, pp. 1–141.

Alfred Müller (1997). "Integral Probability Metrics and Their Generating Classes of Functions". In: *Advances in Applied Probability* 29.2, pp. 429–443. (Visited on 07/30/2022).

NCI Dictionary of Cancer terms (2014).

Nucleus (2022). https://www.genome.gov/genetics-glossary/Nucleus. Online; accessed 01-June-2022.

Satoshi Ohtake and Tsutomu Arakawa (Nov. 2013). "Recombinant Therapeutic Protein Vaccines". In: *Protein and Peptide Letters* 20.12, pp. 1324–1344.

Sadao Ota, Ryoichi Horisaki, Yoko Kawamura, Masashi Ugawa, Issei Sato, Kazuki Hashimoto, Ryosuke Kamesawa, Kotaro Setoyama, Satoko Yamaguchi, Katsuhito Fujiu, Kayo Waki, and Hiroyuki Noji (2018). "Ghost cytometry". In: *Science* 360.6394, pp. 1246–1251. eprint: https://www.science.org/doi/pdf/10.1126/science.aan0096.

Chawin Ounkomol, Sharmishtaa Seshamani, Mary M. Maleckar, Forrest Collman, and Gregory R. Johnson (Sept. 2018). "Label-free prediction of three-dimensional fluores-

cence images from transmitted-light microscopy". In: *Nature Methods* 15.11, pp. 917–920.

Rashmi Upadhyay Pathak, Mamilla Soujanya, and Rakesh Kumar Mishra (May 2021). "Deterioration of nuclear morphology and architecture: A hallmark of senescence and aging". en. In: *Ageing Res. Rev.* 67.101264.

Rakesh M. Patil, Nanasaheb D. Thorat, Prajkta B. Shete, Poonam A. Bedge, Shambala Gavde, Meghnad G. Joshi, Syed A.M. Tofail, and Raghvendra A. Bohara (Mar. 2018). "Comprehensive cytotoxicity studies of superparamagnetic iron oxide nanoparticles". In: *Biochemistry and Biophysics Reports* 13, pp. 63–72.

Karl Pearson (Nov. 1901). "On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11, pp. 559–572.

Luis Perez and Jason Wang (2017). *The Effectiveness of Data Augmentation in Image Classification using Deep Learning*.

Stephen P. Perfetto, Pratip K. Chattopadhyay, and Mario Roederer (Aug. 2004). "Seventeen-colour flow cytometry: unravelling the immune system". In: *Nature Reviews Immunology* 4.8, pp. 648–655.

Judith M. S. Prewitt and Mortimer L. Mendelsohn (Jan. 1965). "The Analysis of Cell Images". In: *Annals of the New York Academy of Sciences* 128.3, pp. 1035–1053.

Fränze Progatzky, Margaret J. Dallman, and Cristina Lo Celso (June 2013). "From seeing to believing: labelling strategies for in vivo cell-tracking experiments". In: *Interface Focus* 3.3, p. 20130001.

Stephan Rabanser, Stephan Günnemann, and Zachary C. Lipton (2018). *Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift*.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation*.

Elena Samuylova (Nov. 2021). *What is the difference between outlier detection and data drift detection?* https://towardsdatascience.com/what-is-the-difference-between-outlier-detection-and-data-drift-detection-534b903056d4. Online; accessed 01-June-2022.

Keiichi Abe Satoshi Suzuki (1985). "Topological structural analysis of digitized binary images by border following". In: *Computer Vision, Graphics, and Image Processing* 1, pp. 32–46.

Bernhard Schölkopf, Alex Smola, Alexander Smola, and A Smola (Apr. 2002). "Support Vector Machines and Kernel Algorithms". In: *Encyclopedia of Biostatistics, 5328-5335 (2005)*.

SeldonIO (2022). *SeldonIO/Alibi-detect: Algorithms for outlier, adversarial and Drift Detection*. https://github.com/SeldonIO/alibi-detect. Online; accessed 01-June-2022.

Sung Wook Shin and Jae Seong Lee (Oct. 2020). "CHO Cell Line Development and Engineering via Site-specific Integration: Challenges and Opportunities". In: *Biotechnology and Bioprocess Engineering* 25.5, pp. 633–645.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56, pp. 1929–1958.

Sternberg (Jan. 1983). "Biomedical Image Processing". In: *Computer (Long Beach Calif.)* 16.1, pp. 22–34.

Jai-Yoon Sul, Chia-wen K. Wu, Fanyi Zeng, Jeanine Jochems, Miler T. Lee, Tae Kyung Kim, Tiina Peritz, Peter Buckley, David J. Cappelleri, Margaret Maronski, Minsun Kim, Vijay Kumar, David Meaney, Junhyong Kim, and James Eberwine (May 2009). "Transcriptome transfer produces a predictable cellular phenotype". In: *Proceedings of the National Academy of Sciences* 106.18, pp. 7624–7629.

Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin (2020). *CSI: Novelty Detection via Contrastive Learning on Distributionally Shifted Instances*.

Betty I. Tarnowski, Francis G. Spinale, and James H. Nicholson (1991). "DAPI as a Useful Stain for Nuclear Quantitation". In: *Biotechnic and Histochemistry* 66.6, pp. 296–302. eprint: https://doi.org/10.3109/10520299109109990.

Thresholding (2022). https://scikit-image.org/docs/stable/auto_examples/applications/plot_thresholding.html. Online; accessed 01-June-2022.

Borbála Tihanyi and László Nyitray (Dec. 2020). "Recent advances in CHO cell line development for recombinant protein production". In: *Drug Discovery Today: Technologies* 38, pp. 25–34.

Types of morphological operations (2022). https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html. Online; accessed 01-June-2022.

Masashi Ugawa, Yoko Kawamura, Keisuke Toda, Kazuki Teranishi, Hikari Morita, Hiroaki Adachi, Ryo Tamoto, Hiroko Nomaru, Keiji Nakagawa, Keiki Sugimoto, Evgeniia Borisova, Yuri An, Yusuke Konishi, Seiichiro Tabata, Soji Morishita, Misa Imai, Tomoiku Takaku, Marito Araki, Norio Komatsu, Yohei Hayashi, Issei Sato, Ryoichi Horisaki, Hiroyuki Noji, and Sadao Ota (Dec. 2021). "In silico-labeled ghost cytometry". In: *eLife* 10.

Dmitry Ulyanov (2022). *Dmitryulyanov/multicore-TSNE: Parallel T-SNE implementation with python and Torch wrappers*. https://github.com/DmitryUlyanov/Multicore-TSNE. Online; accessed 01-June-2022.

Understanding UMAP (n.d.).

Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu (2014). "scikit-image: image processing in Python". In: *PeerJ* 2, e453.

Yingfan Wang (n.d.). *Yingfanwang/PACMAP: PaCMAP: Large-scale dimension reduction technique preserving both global and local structure*.

Yingfan Wang, Haiyang Huang, Cynthia Rudin, and Yaron Shaposhnik (2021). "Understanding How Dimension Reduction Tools Work: An Empirical Approach to Deciphering t-SNE, UMAP, TriMap, and PaCMAP for Data Visualization". In: *Journal of Machine Learning Research* 22.201, pp. 1–73.

Martin G. Weigert, Italo M. Cesari, Shirlee J. Yonkovich, and Melvin Cohn (Dec. 1970). "Variability in the Lambda Light Chain Sequences of Mouse Antibody". In: *Nature* 228.5276, pp. 1045–1047.

Suorong Yang, Weikang Xiao, Mengcheng Zhang, Suhan Guo, Jian Zhao, and Furao Shen (2022). *Image Data Augmentation for Deep Learning: A Survey*.

Yuqin Yao (Jan. 2016). "Image Segmentation Based on Sobel Edge Detection". In.

Xue Ying (Feb. 2019). "An Overview of Overfitting and its Solutions". In: *Journal of Physics: Conference Series* 1168, p. 022022.

Matthew D. Zeiler (2012). *ADADELTA: An Adaptive Learning Rate Method*.

List of Figures

List of Tables