INSTITUTE OF COMPUTER SCIENCES
Master in Artificial Intelligence and Data Science

Universitätsstr. 1 D–40225 Düsseldorf

Heinrich Heine Universität Düsseldorf

AI-based fluorescent labeling for cell line development

Hanna Pankova

Master thesis

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

Erklärung

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

Düsseldorf, den 29. August 2022

Hanna Pankova

Abstract

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

Contents

1 Introduction

1.1 Motivation

1.2 Notation

2 Background

2.1 Biology

cite(Tihanyi et al.): Recombinant proteins are the proteins that are spreadly used in the biomedical research, medicine production and for many different therapeutic needs like, for example, vaccines or antibodies. Therefore currently there is a great need for production of recombinant proteins in high volumetric amounts and of a good quality. Such proteins are mostly produced from the mammalian cells namely Chinese hamster ovary cells (CHO). (doi:10.1016/B978-0-08-100623-8.00007-4) The process that is used for this production is called cell line development(CLD). And the current goal in recombinant protein production is to create a an efficient expression and high-throughput systems to improve the CLD processes. (cite! Tihanyi et al.).

CLD is a process of generating single cell-derived clones that produce high and consistent levels of target therapeutic protein. (pharma.lonza.com/offerings/mammalian/cell-line-development)

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

The reason behind the use of mammalian cells hides in the fact that they can produce diverse correctly folded proteins and most importantly they have a high productivity. (cite Tihanyi). The productivity is measured in titre of the produced protein, and they can reach (0.1 - 1 g/L in batch and 1-10 g/L in fed-batch cultures). (cite Tihanyi TODO explain batch) Mostly all of the mAbs are produced using CHO cells (cite http://dx.doi.org/10.1016/j.jbiotec.2017.04.028.).

Currently companies use the same host cell line for their productions as well as in clinical trials as already checked and qualified cells simplify their road to the clinic. That is why current research can have a wide applicability in all CHO protein productions.

However there is a downside in the use of CHO cells as they are well-known by their instability. As a rapidly growing immortal cells they are also genomically unstable and extremely heterogeous which ususally leads to their main problem - production instability. The problem of choosing the stable and high producing clones that also will be able to express protein quailitatively and quantitably over time is essentially the main problem which is solved in this research.

The main challenge for manufacturing are the costs and times of productions. Right now most of the research attention is dedicated to the reduction of times and costs for CLD processes, as well as for developing techniques of high-throughput clone screening and characterization. (cite Tihanyi). With the great amounts of data acquired over time and the development of the computational modelling and statistical analysis it is possible now to do the analysis *in silico*, meaning - computationally without intervening into the cells instead of the usual *in vitro* analysis.

Differential interference contrast (DIC) microscopy is an optical microscopy technique used to enhance the contrast in unstained, transparent samples (wikipedia).

Proteins are large biomolecules and macromolecules that comprise one or more long chains of amino acid residues (https://en.wikipedia.org/wiki/Protein).

Fluorescent labelling is the process of covalently binding fluorescent dyes to biomolecules such as nucleic acids or proteins so that they can be visualized by fluorescence imaging (https://www.nature.com/subjects/fluorescent-labelling). 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 experimentalists toimage particular components of a given cell in detail. (O'reilly life sciences p113)

Cell line development (CLD) is the process by which the cellular machinery is coopted to manufacture therapeutic biologics or other proteins of interest. One can use different expression systems for cell line development: bacterial, plant-based, yeast, mammalian. (copy paste from https://www.beckman.de/resources/product-applications/lead-optimization/cell-line-development)

First step of CLD is the introduction of the gene of interest (GOI or a DNA vector or an expression vector) to CHO cells. This process is called a transfection. And it has two main downsides: first is that the transfection mostly results in a vector being inserted into a random site within a host cell genome and second, generally low efficiency of integration (cite Tihanyi). It is important to transfect a GOI into an optimal site of genome to secure a high protein expression over time during protein production, however pratically transfection transfection happens into a random location of a genome. In case the gene was transfected in the inactive site of genome (and the majority of genome is not transriptionally active) the cell will likely not express the gene. (doi:10.1016/B978-0-08-100623-8.00007-4) (doi:10.1016/j.coche.2018.08.002)

The second step is the selection of cell pools that have successful and stable gene integrations. The reasond why not all of them are suitable for cloning is the following: during the transfection only 80% of cells recieve the vector of GOI (doi:10.1016/B978-0-08-100623-8.00007-4), only the small percent of which actually integrate a vector into the genome and, as mentioned above, only a fraction of those cells are able to stabily express a protein. (Reference needed). Such selection could be done with bulk sorting algorithm. (doi:10.1016/B978-0-08-100623-8.00007-4)

The third step in CLD is to clone the cells. The chosen stable pools of cells are phenotypically and genetically diverse - meaning they have different growth rates, metabolic profile and etc. This is not ideal for industrial production - all the cells used for protein production should be derived from the same clone ([25] here doi:10.1016/B978-0-08-100623-8.00007-4). In order to choose single best cells for further cloning one asseses several parameters like cell size, granularity, cell surface protein expression and etc. This can be done with Fluorescent Activated Cell Sorting (FACS) technology that allows to sort signle cells. (https://doi.org/10.1517/14712598.4.11.1821). Unfortunately fluorescence labeling is expensive and may ruin the cell due to its phototoxicity (https://doi.org/10.1371/journal.pone.0007497). Yeo et al. (cite Tihanyi) also found out how different selection markers can affect the production stability of CHO cells. There is a limited number of available fluorescent channels in microscopes as well as such labels

can also be inconsistent, depend a lot on reagent quality, and require many hours of lab work. Therefore there exists a need for flurescent labeling in silico - without intervening into the cell.

Assessing the productivity rates of the clones is very labour intensive and time-consuming process that can and should be optimized.

Once the cells are cloned, phenotypical and genetical heterogeneity is reduced, the next step is to charaterize clonally-derived cells based on the following criteria: cell size, growth rate, protein quality, titer, metabolities and etc. With this one can estimate clones productivity and titer. Such observations may take up to 90 days after which one can determine which cells are stable and therefore suitable for production. This is the last step of CLD process and consumes a lot of time and maintaining costs for feeding and cloning the cells. Predicting the stability of the cells directly from DIC images would reduce this time significantly allowing to escape this process completely.

However there are also some disadtantages of this approach. First, it can be less accurate than skilled cells staining perfomed manually. Extreme or unsual clones and phenotypes might be challenging if they were not used in the training set of images.

2.2 Imaging

The microscope used in the experiments takes photos of the well plate in random locations. The reason for that hides in the focusing problem, to get a reasonably good photo without blur it has to focus on a specific location, this is done there automatically, therefore the location of the focus is almost random (see Figure 1).

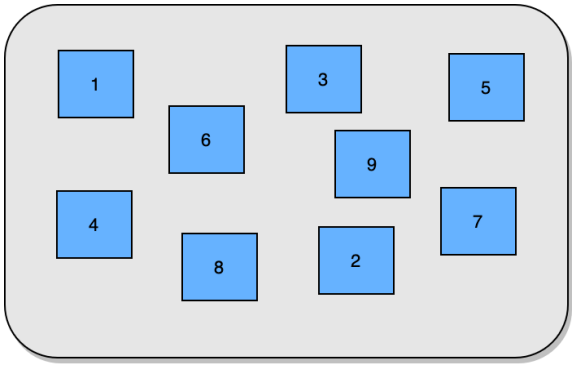


Figure 1: Way in which photos of the well-plate were taken

Although it might be problematic in the following sense: photos takes by the microscope in such manner do not gurantee that the focus will land in distinct spots all the time. This means that some cells taken during one of the photos might appear in the later ones. Since the photos have a high-resolution the crops are first performed and it might happen that same cell might appear in several crops. Afterwards, when crops are split between train, test and validation datasets it might happen that the same cell 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 loss during training.

2.3 ML

Background on Unet and ML in general

Convoutional neural network is a neural network that is based on convolutional layers. It is a powerful tool for image processing and is used in medical imaging. Convolution is a linear operation used in convolutional layers that can be performed by applying a kernel (a 2d matrix) across a bigger input matrix called tensor, which can be 3d. Element-wise product between them is calculated and summed, this value will be an element of the output 2d matrix. Kernel slides accross all locations of the input tensor. In case if several different kernels were used then a 3d tensor will be created.

Main advantage of convolutional neural networks if weight sharing. Kernel has learnable weights however these weights are shared across all locations of the kernel on the input tensor, thic strongly reduces the number of parameters needed.

CNNs also uses non-linearities like RELU, ELU, Tahn, Sigmoids and etc. They are also often combined with max pooling layers and dropouts to escape overfitting.

Overfitting is one of the most often problems in deep learning that prevents model to generalize well for unseen data. This can happen when the model is too big for the amount of training data given, it was not regularized well or there is just not enough data for training.

U-Net architecture is widely used for segmentation purposes. It is a convolutional neural network with the following architecture: [img] . It first performs image downsampling and upsampling afterwards. [https://arxiv.org/pdf/1505.04597.pdf] The following architecture have been used for nuclei prediction.

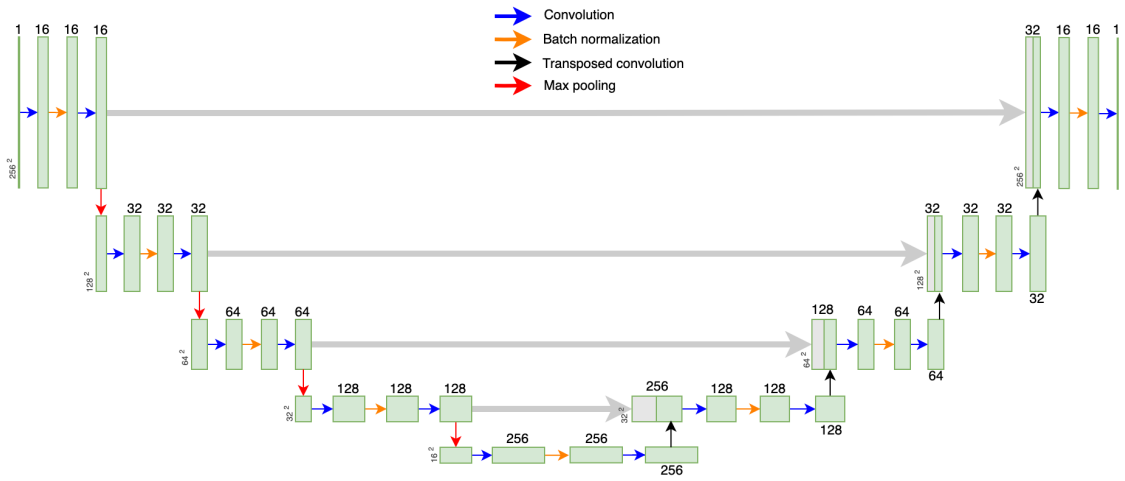


Figure 2: Unet

3 Prediction of cell organelles

3.1 Overfitting

Definition 3.1 (Overfitting)**.** "Hypothesis overfits the training samples if some other hypothesis that fits the training samples less well actually performs better over the entire distribution of instances" (p67 Mitchell Machine Learning 1997).

Overfiting prevents model to generalize well on the unseen data and in order to avoid fitting to closely to the training dataset one has several options:

(Bishop book)

3.1.1 Early-stopping

Overfitting effect occurs at later epochs. This doesn't happen during early epochs as with the correct weights initialization the weights of the model are quite small and random and therefore the best decision surface would be a smooth one. But in the later epochs the difference in values of the weights grows and they become not similar anymore which means also that the decision surface becomes more complex and will be ble to fit not only the training data itself, but also its noise. (p111 Mitchell Machine Learning 1997). And that is why stopping before the model became too complex for the given data may mitigate this problem.

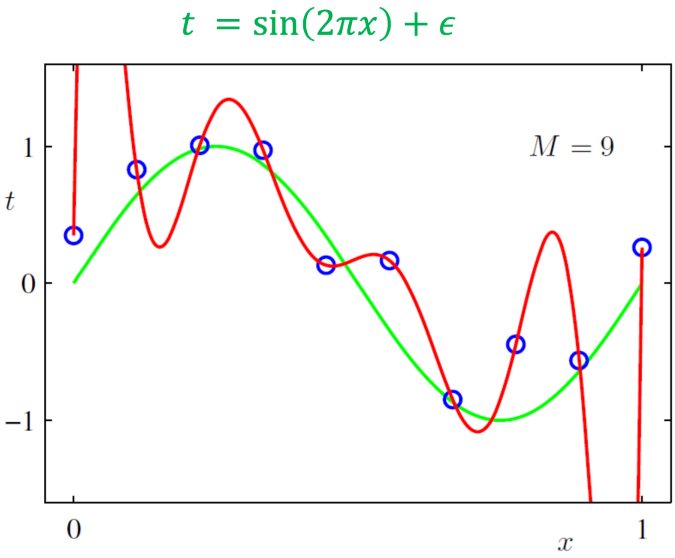


Figure 3: Overfitting

3.1.2 Regularization

The complexity of the 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 a noise to be a feature. To prevent this one should decrease the weights associated with useless features, however we cannot know ahead which of them should be ignored, therefore one may limit them all. (doi:10.1088/1742-6596/1168/2/022022) In order to do that, a penalty term in loss function is added:

$$\tilde{L}(\theta, X, y) = L(\theta, X, y) + \lambda R(\theta) \tag{1}$$

for some $\lambda > 0$. This is called a *soft-constrait* optimization. When $R(\theta)$ is of the form $R(\theta) = ||\theta||_2^2 = \sqrt{\sum_i \theta_i^2}$ this is called *L2-regularization* and 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 and Pratt (1988) as follows:

$$\theta_{t+1} = (1 - \lambda)\theta_t - \alpha \frac{\partial L}{\partial \theta_t} \tag{2}$$

where α is a learning rate. Weight decay successfully affects more those weights the gradient change along which is smaller (Goodfellow Deep learning p229). *L1-regularization* induces sparsity of the weights by assining some of them to zero, this could be also considered as feature selection approach.

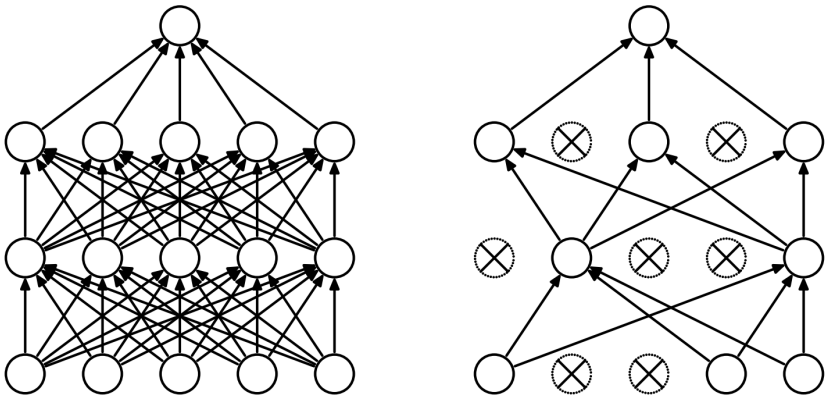
Regularization techniques like BatchNorm and Dropout could also be applied. Batch-

Norm is defined by :

$$
\begin{aligned}
y_i &= \gamma \frac{x_i - \mu_B}{\sigma_B^2 + \epsilon} + \beta \\
\sigma_B^2 &= \frac{1}{m} \sum_i^m (x_i - \mu_B)^2 \\
\mu_B &= \frac{1}{m} \sum_i^m x_i
\end{aligned}
\tag{3}
$$

where m is a batch size. Ioffe and Szegedy, 2015

Dropout is a technique that randomly sets some of the weights to zero. (Srivastava, Hinton 2014).



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

Figure 4: Dropout

From the first training one can clearly see that overfit happens around epoch 30. Although one could just pick out one of epochs before the 30th one (before overfit has happened), applying more regularizion to the model that has been using dropout only would be a good idea. Early stopping in combination with weight decay, BatchNorm were used to regularize the model that was overfitting too quickly with dropout only. $BatchNorm$ layers have been added after the first Convolution layer in each ConvBlock and $TransposedConvBlock$. The results of training the regularized network is presented in Figure 6.

3.1.3 Network reduction

Since learning a too complex and noise-fitting decision surface might be an often cause of an overfit, another way to mitigate it 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. (cite Page 332, Neural Networks for Pattern Recognition, 1995.)

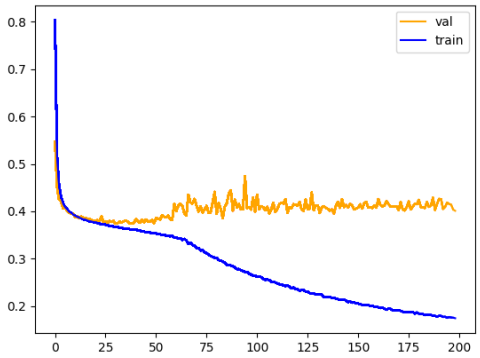


Figure 5: Not regularized

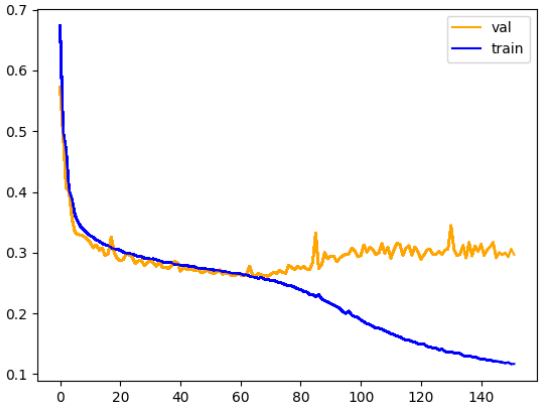


Figure 6: Regularized

3.1.4 Expansion of the training data

To well-tune the hyperparameters the model needs to have a sufficient amount of quality samples. An expanded dataset can improve the quality of the predictions, (cite doi:10.1088/1742-6596/1168/2/022022) however only when the model has already performed well on the initial dataset. If the model performing bad initially, adding more data will not solve the problem. Here having TODO n samples of data the model was trained on TODO samples only to find the best structure and regularization first, afterwards the model was retrained using more data and the quality imroved from, to TODO.

4 Crops combination

Since the neural network has been constucted in a way that the input size is 256 by 256 pixels, one cannot feed a bigger image in there. Therefore a high-resolution image has to first split into several crops which are then get recombined. There are several ways of how one can split the image, the easiest approach would be to use a sliding window. With the step size of s and the window size of w the sliding window is defined as:

When step size s is equal to window size w, there is no overlap between the windows. One very important insight from prediction results if that the model is less accurate on the borders of the image, rather than of cropping itself. Images are cropped consequently and most the times there are cells on the borders of the crops that get sliced and it might be impossible the make a good prediction for them just due to the lack of the 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 ingore predictions on the border.

Such approach would help to reduce the effect of the grid visible on the image composed of many small crops, which one can see in the Figure 7 to almost non-visible borders as in the Figure 8. This would of course take longer time in predictions, however as the speed is less crucial in comparison to the accuracy of the predictions.

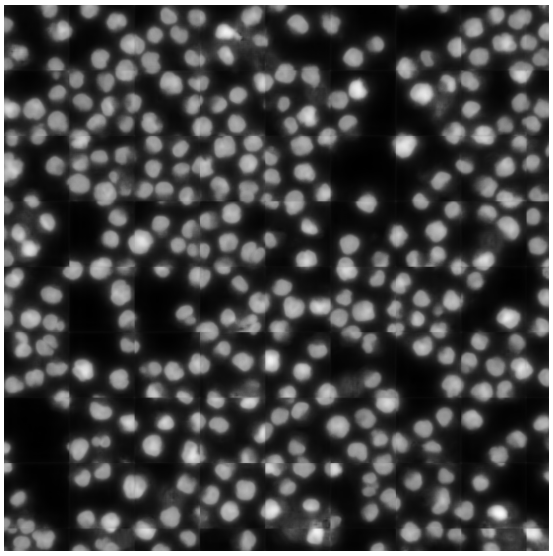


Figure 7: No overlap

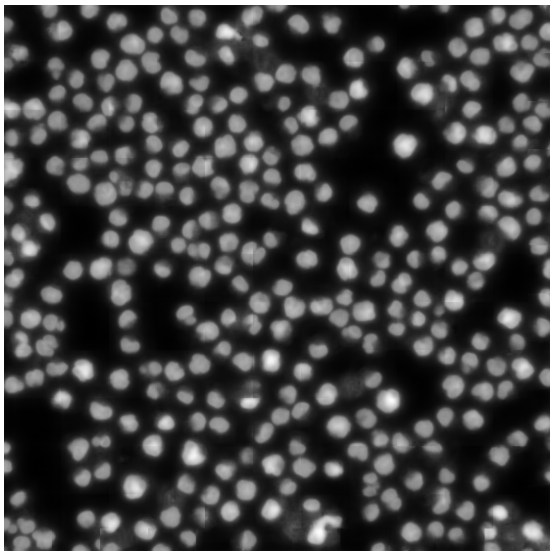


Figure 8: 30 pixels overlap

5 Predicting cell organelles

5.1 Loss

Although one can easily define a loss function like MSE or Pearson correlation coefficient between ground truth and predicted fluorescence image, there is still a problem is relating this metric to estimating quality of predicted images in practice. There is a clear need to clearly state the down-stream tasks that are going to be performed in the predictions from our data. However these downstream tasks might be different and are not determined right away. For example even though the training of the model happens with the use of Pearson score or MSE, the real practical evaluation will happen in terms of metrics like: the number of the nuclei in the image, the closeness of the intensities of the nuclei of interest, the difference in their mean intensities, the level of details in the Golgi apparatus and the strength of the non specific background fluorescence noise. These metrics are not known in advance and therefore there will always be a gap between the metrics that are used during train and the metrics that are used in practice.

During training 2 following losses have beed used. MSE loss:

$$Loss_{MSE} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{W}\sum_{k=1}^{H}(y_{j,k} - \hat{y}_{j,k})^2 \tag{4}$$

where N is the number of images in the batch and $y_{j,k}$ and $\hat{y}_{j,k}$ are the {j,k}th pixel of the ground truth and predicted images respectively.

Pearson Correlation Coefficient (PCC) is commonly used in cell biology when comparing the co-localization of two or more proteins, and also used in computer vision to assess spatial-intensity when determining image similarity (reword and cite Cohen). It is calculated as follows:

$$Loss_{PCC} = \frac{\sum_{i=1}^{N}(y_i - \bar{y_i})(\hat{y}_i - \bar{\hat{y}_i})}{\sqrt{\sum_{i=1}^{N}(y_i - \bar{y_i})^2(\hat{y}_i - \bar{\hat{y}_i})^2}} \tag{5}$$

where y_i is the flattened ground truth image, $\hat{y}_i$ is flattened predicted image and $\bar{y_i}$, $\bar{\hat{y}_i}$ are means of the ground truth and predicted images respectively.

5.2 Nuclei

In this subsection the results of the nuclei predictions will be presented. You can see examples of predictions presented in Figures TODO and TODO. There are TODO visible problems with the predictions:

- The form of the nucleus is captured, but the texture inside is not
- Blurry border aorund the nuclei
- Problems with the predictions on the borders of the crop

Predictions of the border of the crops are quite challenging for the model as there is not enough of the information due to the cropped parts of the cell. However this can be easily solved with using overlaps while cropping the image to avoid the use of the pixels predicted on the border. There more information on this in Chapter TODO (Crops combination).

Explanation for the lack of details inside TODO

The nucleus is relatively a simple task for predictions as it might be relatively easier to spot the nuclei manually as well. They are relatively big with respect to the cell as well. More challenging task would be a prediction of Golgi Apparatus.

5.3 Golgi Apparatus

Golgi Apparatus (or simply Golgi) is another organelle inside the cell, that packages protein into membrane-bound vescicles. One can see an example of how the staining of the Golgi with fluorescence imaging looks like in Figure TODO ref. It is evident that there is a lighter foreground fluorescence and a bit darker one apart from the black background itself. Truly Golgi is only the lighter part of fluorescence lightning and the other part is called a non-specific fluorescence lightning. It comes from the cell itself, and might occur when the antigen is impure and contains antigenic contaminants. (cite Immunofluorescence in medical science indian) Its brightness may vary due to longer or shorter exposure time.

Having such a non-specific fluorescence background may have 2 potential challenges for training:

- The relative area of the background fluorescence is bigger than the area of Golgi themselves. Therefore quite a big part in the loss during training will be dedicated to teaching model to restore this background fluorescence instead of the Golgi itself.
- It also introduces difficulties during the post-processing of the predictions. As well as for nuclei, the mask of the predicted Golgi Apparatuses will be needed. Using the same algorithm for post-processing 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.

To escape this one has to subsctract the background first or so-called to enhance the image. The main approach was to replicate the post-processing that is used in practice in labs for segmenting fluorescence images of Golgi.

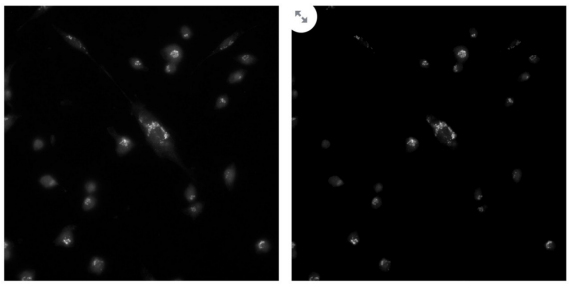


Figure 9: Golgi enhancement

On the left of Figure 9 one can see the original ground truth image and on the right - the images after the background was substracted. To do so the rolling ball algorithm has been used.

5.3.1 Rolling ball algorithm

Rolling ball algorithm has been introduced by Sternberg in 1983 (cite doi:10.1109/mc.1983.1654163) and is still widely used for processing medical and biological data. The idea of this algorithm is based on morphological opening of the image.

Morphological opening is an operation on the image involving first eroding the image and then dilating it with the same structuring element for both operations. It is helpful for extracting big image features. (cite https://cutt.ly/PGJjucI) Structuring element is a matrix smaller than the image itself that defines the area around the pixel that is going to be processed to define its new value after the morphological operation is performed. It is an analogues of a kernel in image processing. You can see an example of a structuring element on the Figure 10

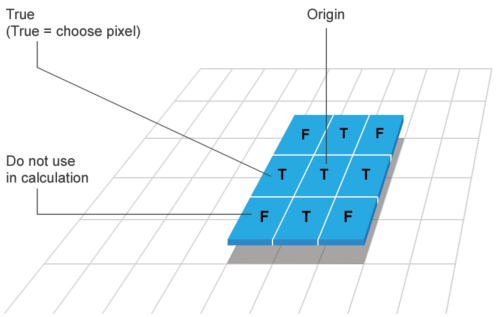


Figure 10: Structuring Element

Morphological dilation takes a new value of the pixel as the maximum value of its neighbours within the structing element. Therefore after this operation lines will be thicker and in general objects will appear bigger.

Morphological erosion on the other hand makes the pixel value as the minimum value of its neighbours within the structuring element. After this operation the floating pixels will be removed and all object become smaller and thinner.

Sternberg has extrapolated the operation of morphological opening from 2D into 3D space. If one can imagine the image to be a 3D plane, with the height of ech pixel being determined by its intesity, such an interpretaion of the image is called umbra. The structuring element for morphological opening of an umbra has to be then also a 3D object - a ball for example. The opening of an umbra then will be a union of translations of the 3-D structuring element that can be rntirely contained inside umbra (see Figure 11). One can image the ball freely moving inside the volume constrained by the upper surface of an umbra. The opening then consists of all the pixels then can be reaches by the ball. The radius of the ball then is the hyper-parameter than has to be tuned.

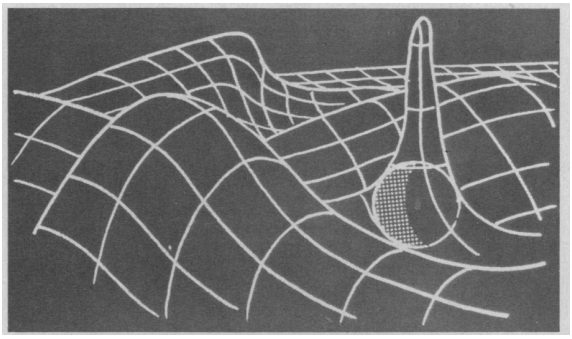


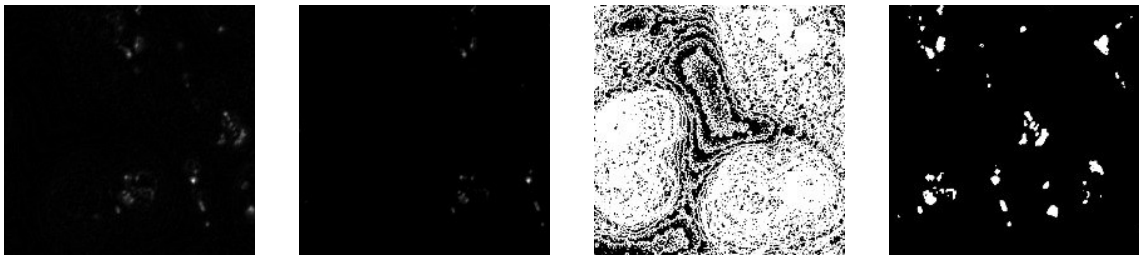
Figure 11: Rolling Ball

However, subtracting the background with rolling ball algorithm is not enough to get a reasonably clean signal of Golgi Apparatus from fluorescence imaging. To check this one can convert the original image from the vanilla pre-processing with rolling ball algorithm into a mask. To do so, let's just assing all non-zero pixels to the maximum value of image intensity. The mask that have been created like this would look like in Figure 12c. It clearly still contains a lot of background noise of low intensity and that is why this background noise was not visible for an eye. In order to remove it one could clip lower intensities of the image via the following approach:

```
import numpy as np

def clip(image):
  minval = np.percentile(image, 90)
  image = np.clip(image, minval, image.max())
  image = (image - image.min()) / (image.max() - image.min())
  return image
```

The result of the clipped image and its mask are illustated in the Figure 12b, 12d correspondingly. It contains almost no background noise now. Such additional clipping might improve the results slightly.



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

Figure 12: (a) Vanilla pre-processing with automatic background removal algorithm only; (b) Additional clipping of lower intensities after vanilla pre-processing; (c) masked or subfigure (a); (d) mask of subfigure (b)

6 Downstream metrics

The evaluation of the model in terms of the loss is not quite objective for the current problem. Even when one model might have a smaller loss, it doesn't nessecarily will perform better than another model. (cite Cohen) Therefore the evaluation of the model is done in terms of the metrics that are defined as follows. The most used ones by the clients were

- Number of nuclei
- The relative area of the nuclei
- Total intensity
- Mean intensity

They would be calculated for each image in the test set and then the distributions of these metrics for ground truth and predictions would be compared visually with violin plots and Spearman rank and Pearson correlation coefficients

6.1 Nuclei segmentation

6.1.1 Challenge

To properly evaluate these metrics on model predictions, post-processing first for segmenting the nuclei is needed. This is not a very straight-forward task to do as there are different edge cases where the nuclei are difficult to segment due to the variety of different factors.

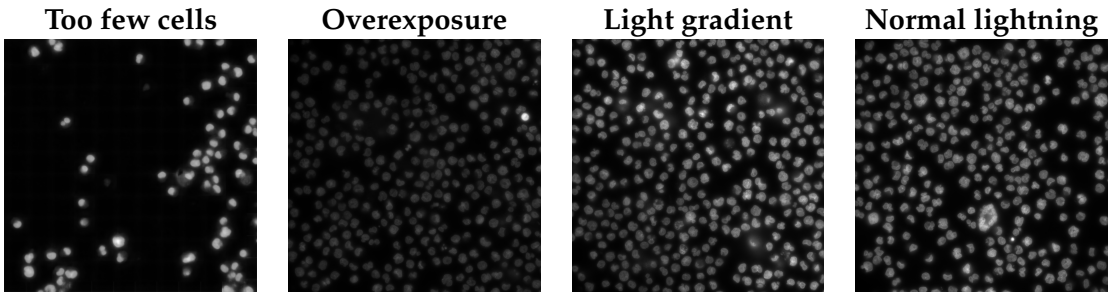


Figure 13: Different lightning conditions

For example, different brightness of the images that comes from different exposure during the photo taking process might make the nuclei segmentation more challenging. The same goes for the different lighting conditions. Different lighning conditions are presented in Figure 13. The following inconsistenties in lightning conditions are presented (from left to right): image contains too few cells, which leads to background being much darker than usually; overexposure of one cell, which leads to difficulties of segmenting the rest of the cells as they are hard distinguishable from the background; lighting gradient from darker (left bottom corner) to brighter (upper 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 the division. Also when some have already fully divided, they might still be located close to one another. The example of such situation is presented in Figure 14.

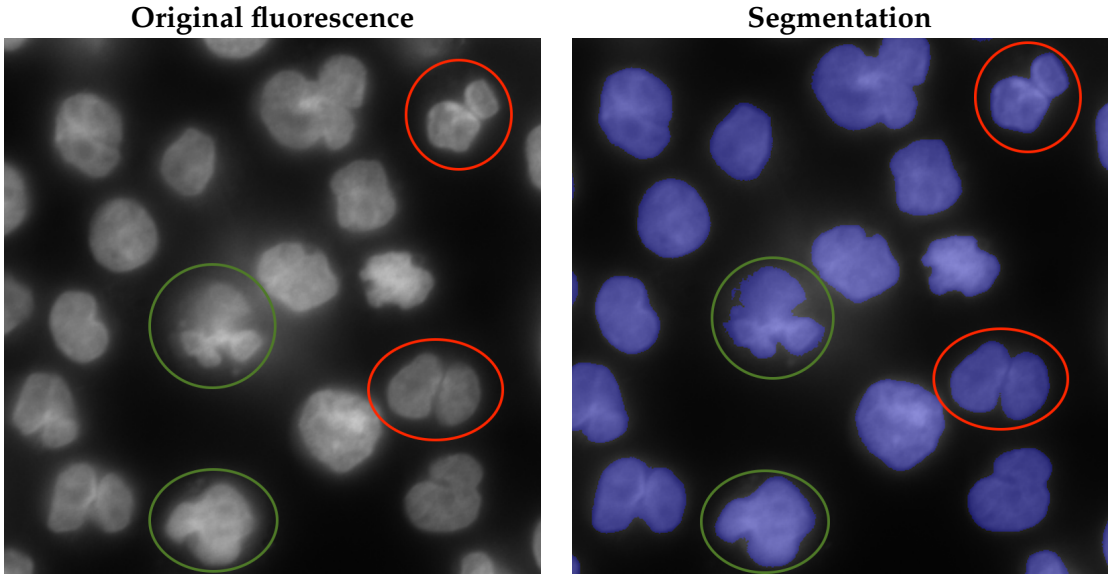


Figure 14: Closely located cells

Here cells, that are not yet fully divided are highlighted with the green cirles, and ones, that are fully divided, but just located too close to one another are highlighted with red

circles. You can see that the segmentation algorithm (described in TODO) recognises both such cases as one cell.

6.1.2 Thresholding

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 $x_i < T$ are assigned to be of class 0 (background) and all pixels that are larger than this threshold $x_i > T$ are assigned to be of class 1 (foreground). To find a threshold automatically Gonzalez et al. (cite Digital Image Processing (2nd Edition)) proposed the following algorithm:

Algorithm 1 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.

There are different ways of how one can define such initial threshold. There is also no single best solution for all of the cases. For example, when one has an assumption that the foreground occupies approximately the same area as the background, than initial threshold T should be chosen to be an average gray level. In this case global thresholding did not perfom well due to different intensities in different regions of the images (non-uniform illumination). Distribution of the cells also varies through the dataset and some images contain more cells, while other contain only few.

In order to segment the background from the foreground (nuclei), the following function was used:

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

This is where basic adaptive thresholding or local thresholding comes in handy. The advantage of this method hides in the fact, that it doesn't compute a threshold based on the full histogram of the image, but uses parts of it to compute different thresholds for different subregions of the image. This method is also known as adaptive or dynamic thresholding. The threshold value is the weighted mean for the local neighborhood of a pixel subtracted by a constant. (cite Digital Image Processing (2nd Edition)). With the image size of 2136x2136, the local neighborhood or a *block size* was chosen equal to 111 by experimenting with different values. The default method used on for local thresholding is *gaussian*. *offset* valut is a constant that will be subtracted from weighted mean of neighborhood during the calculation of the local threshold, by default this value is 0. (cite skimage)

Let z be a random variable that quantifies a gray-level value of the pixel, then the his-

togram 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 $p_@(z)$ weighted by the relative areas of these two classes (their number of pixels) P_1 and P_2. Then

$$p(z) = P_1 p_1(z) + P_2 p_2(z) \tag{6}$$

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

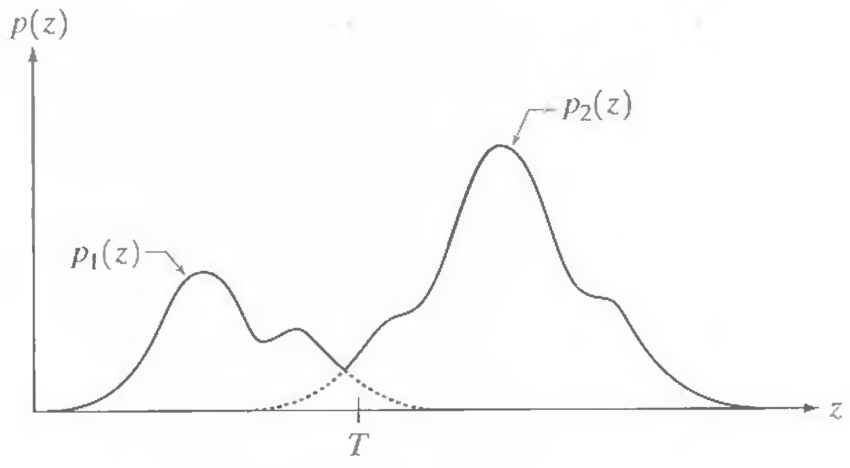


Figure 15: Histogram as a probability density function

Probability to falsely classify an background pixel as object then is:

$$E_1(T) = \int_{-\infty}^{T} p_2(z)\, dz \tag{7}$$

And probability to falsely classify an object pixel as background then is:

$$E_2(T) = \int_{T}^{+\infty} p_1(z)\, dz \tag{8}$$

The overall error is:

$$E(T) = P_1 E_1(T) + P_2 E_2(T) \tag{9}$$

By differentiating $E(T)$ wrt to T and equating the result to zero the optmal equation will be:

$$P_1 p_1(T) = P_2 p_2(T) \tag{10}$$

Local Threshold	Global Threshold
0.3 sec	17 sec

Table 1: Threshold timing

Since Gaussian distributions have been assumed then:

$$p(z) = \frac{P_1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(z-\mu_1)^2}{2\sigma_1^2}} + \frac{P_2}{\sqrt{2\pi}\sigma_2} e^{-\frac{(z-\mu_2)^2}{2\sigma_2^2}} \tag{11}$$

With μ_i and σ_i^2 for $i \in \{1, 2\}$ are the mean and variance of the Gaussian distribution $p_i(z)$. This results in the following solution for T:

$$AT^2 + BT + C = 0 \tag{12}$$

where

$$\begin{aligned} A &= \sigma_1^2 + \sigma_2^2 \\ B &= 2(\mu_1\sigma_1^2 - \mu_2\sigma_2^2) \\ C &= \sigma_1^2\mu_2^2 - \sigma_2^2\mu_1^2 + 2\sigma_1^2 2\sigma_2^2 ln\left(\frac{\sigma_2 P_1}{\sigma_1 P_2}\right) \end{aligned} \tag{13}$$

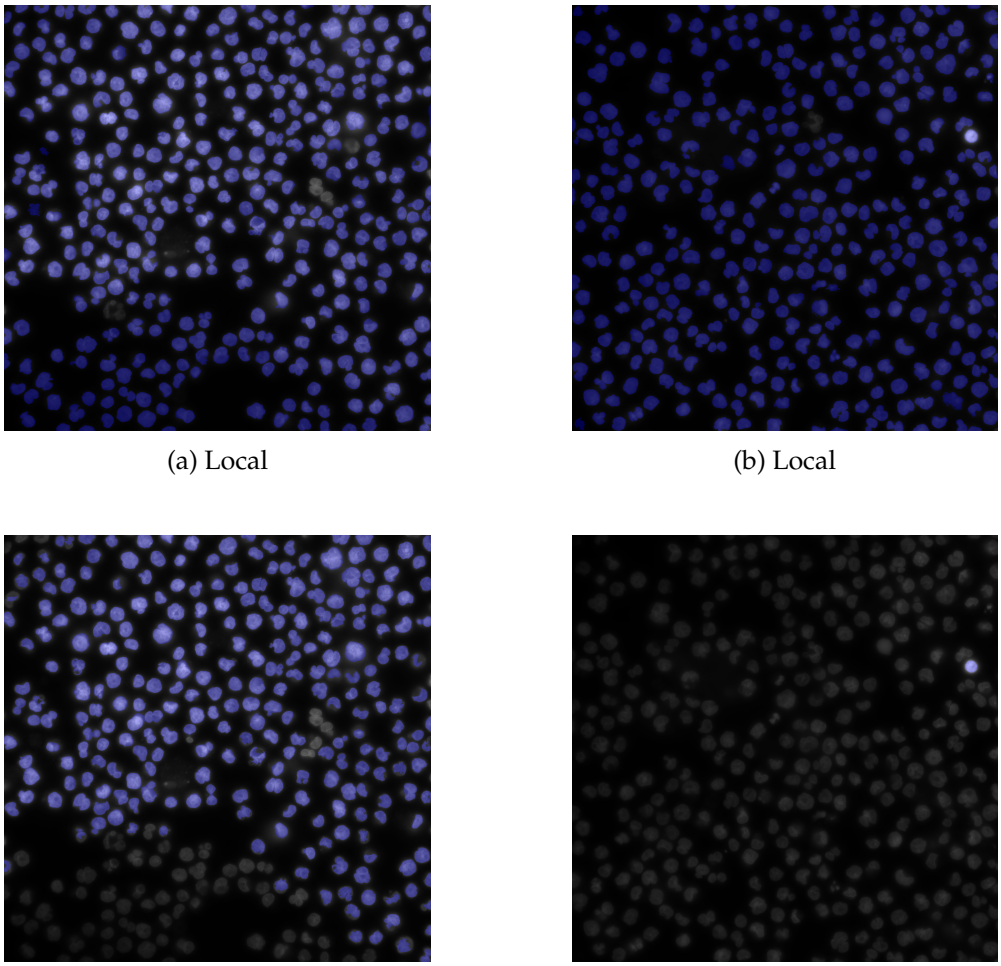
To escape two optimal solutions of the quadratic equation it may be assumed that $\sigma_1 = \sigma_2 = \sigma$ and then:

$$T = \frac{\mu_1 + \mu_2}{2} + \frac{\sigma^2}{\mu_1 - \mu_2} ln\left(\frac{P_2}{P_1}\right) \tag{14}$$

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

Of course local thresholding approach takes a longer time: TODO table with timing. Therefore there as an alternative one can use *minimum thresholding*. This is a global thresholding approach which performs visually a bit worse that a local threshold, however it is much faster (see Table 1).

Comparison of the prdections in difficult lightning conditions of minimum thresholding and local adaptive thresholding are presented in the Figure TODO. In extreme cases of difficult lighting conditions the local adaptive thresholding is much better than the minimum thresholding, however on the images of the better quality TODO (see Figure 17) minimum thresholding might successfully sibstitute the local adaptive thresholding when the time of processing is crucial.



(a) Local

(b) Local

(c) Global, minimum

(d) Global, minimum

Figure 16: Local vs. Global thresholding

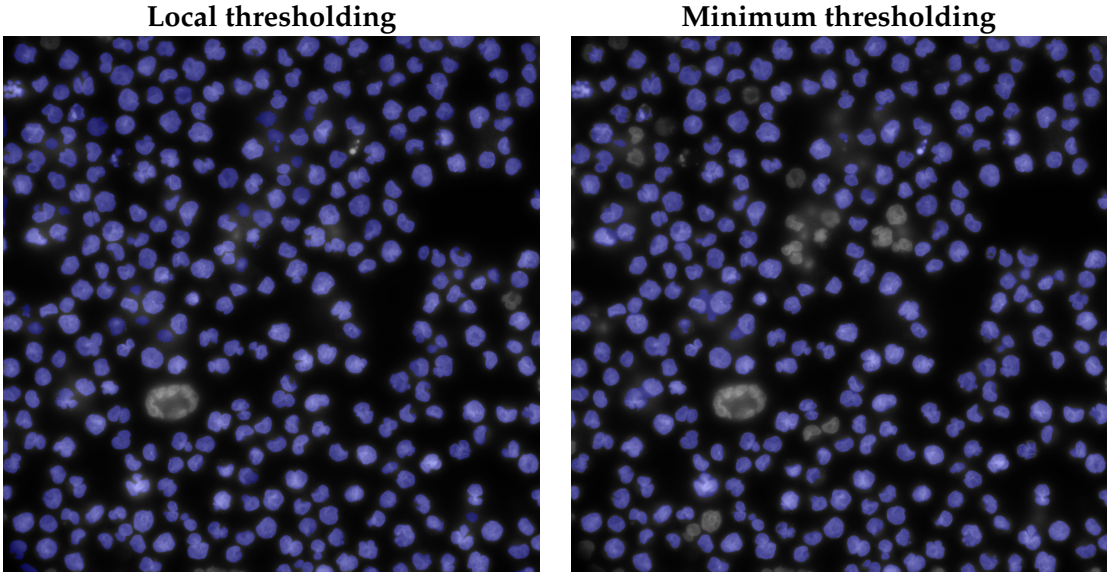


Figure 17: Local vs. Global thresholding (normal conditions)

According to skimage documentation, minimum thresholding works in the following way: (cite https://doi.org/10.1111/j.1749-6632.1965.tb11715.x) it assumes that the histogram of the image is bimodal, meaning that it has two clearly defined peaks, then it iteratively smoothes the histogram using a running average of size $k = 3$ (see Equation 15) until only 2 local maximas are left. Afterwards, the lowest point between these two peaks is found and assigned to be a threshold value.

$$a_k = \frac{1}{k} \sum_{i=n-k+1}^{n} p_i \tag{15}$$

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

$$x_k \leq T \leq y_k \tag{16}$$

TODO arrange this equation better

That is why also images which histograms have very unequal peaks or a broad and flat valley will be unsuitable for this method. (cite skimage)

6.1.3 Overall algorithm

With the chosen type of thresholding, the following algorithm is applied to the image to obtain the mask of nuclei:

Algorithm 2 Fluorescence segmentation

1. Normalize image.
2. Apply chosen 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 insides the nuclei.
4. Run *findContours* from opencv in order to obtain separate regions and filter them based on the following criteria: filter out too big regions (measure the biggest possible nuclei manually), too small regions (measured manually as well), regions that have a shape that it not very 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.

Subsequent steps of such algorithm are presented in the Figure 18

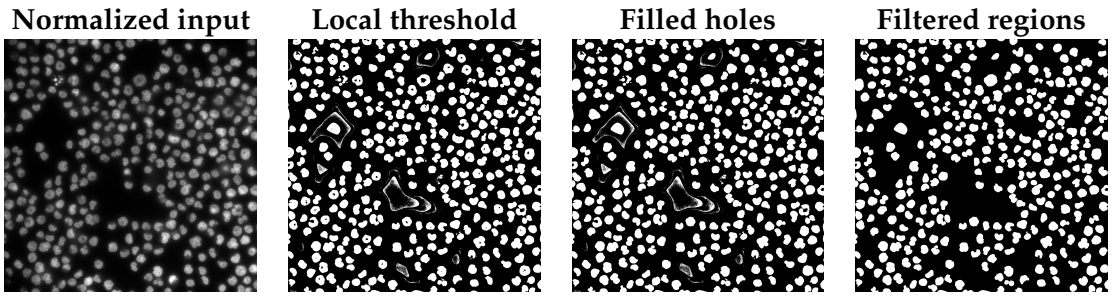


Figure 18: Fluorescence segmentation

6.2 ER-segmentation

Endoplasmic reticulum or ER is another cell organelle. It is a continuos membrane system that forms a series of flattened sacs within the cell. It has many important functions such as synthesis, folding, modification, and transport of proteins. (Rogers, Kara. "endoplasmic reticulum". Encyclopedia Britannica, 11 Nov. 2020, https://www.britannica.com/science/endoplasmic-reticulum. Accessed 16 May 2022.) Therefore detecting ER within the cell and its quaility and its further analysis can give insights on the protein production of the cell. For example the proximity of the ER to the nucleus allows it to control the protein production. For example, when the protein os misfolded or incorrectly folded it will accumulate in the ER lumen and will be a signal to activate misfolded protein response.

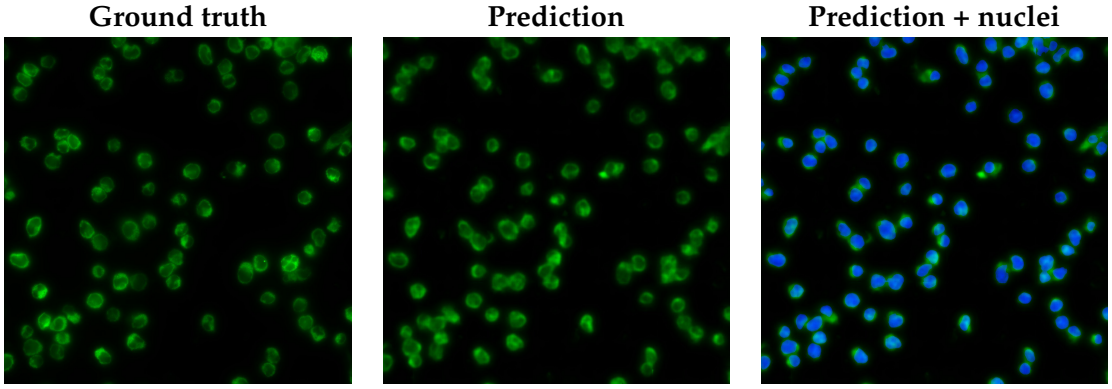


Figure 19: ER prediction

For staining this organelle Fluorescein isothiocyanate (FITC) method was used. The process of ER fluorescence segmentation is somewhat different to nuclei segmentation. These fluorescence staining has a stronger "shining" around the ER itself and therefore a simple background removal would be helpful to reduce it. Another big reason to use this approach hides in the escaping of the background noise recognized as a signal by the further local thresholding algorithm as in Figure 18 c. Such regions from the background

noise cannot be filtered out that easily for ER-fluorescence. Ther reason is the following: the main criteria for filtering such regions in nuclei fluorescence imaging was the shape of the region. All nuclei are almost round and convex, while background noise might be not convex at all. However in FITC imaging some of the cells are located so close one to another that they may form a long non-convex object that would be filtered out based on previous criteria. Therefore removing the background noise if an improvement for pre-processing of images on which the local thresholding would be performed.

In order to do that one can first apply a rough over-predictive global thresholding over the fluorescence image, that will cover a true signal fully and ignore the background noise. Mean thresholding algorithm was chosen for this as it perfectly does the over prediction for these images (see Figure TODO ref). The mask created with the mean thresholding approach is used to zero out all the pixels that are not covered in the mask. Only after that the local thresholding is applied (in comparison to nuclei where one can apply local thresholding directly) with the TODO 181 (Fugure TODO). Then algorithm covers all the holes in the middle on the cells in fluorescence that might have appeared during the thresholding. Morphological opening (see Section TODO) and Gaussian Blur with the squared kernel 3x3 are applied. Connected components are detected afterwards and filtered based on the limit of the area they occupy, this filters out mostly very small components from the mask which might be produced by the left out background noise. The whole algorithm overview you can find here:

Algorithm 3 Fluorescence segmentation

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

Segmentation steps are also iilustrated in Figure TODO

Figure 20: ER prediction

6.3 Golgi

7 Stability and generalizability

7.1 Stability

In order to measure the stability or robustness of the models 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. One is the real corrupted images made in the laboratory. Corruptions in this cade may come from different sources: for example, bubble of oil that landed on the microscope lenses, low density of the cell on the image, over- or under-exposure. Another source would be images with pseudo corruptions created manually.

In this subsection the results from evaluating the model on TODO types in image corruption are presented, namely: defocus blur, gaussian bluer, gaussian noise, .. TODO. Every corruption will have different effects on the prediction of the model based on the severity of the corruption. Therefore it is important to evaluate the error-rate (in this case a loss function) for the predictions for different severity levels of each of the corruption types. It is also important to take visual evaluation of the prediction qualities as well. Each of the corruptions c will have 5 different severity levels s, meaning that $1 \leq s \leq 5$, however it is important to keep in mind that although the severity levels were chosen to be as much comparable between each other as possible, they still might have differences in the strengths, as for example brightness has much stronger effect on predictions than gaussian blur (TODO).

In the next subsections several artificial corruptions are presented.

7.1.1 Gaussian Noise

In order to intorduce Gaussian noise to the image one first can normalize the image between 0 and 1 and then add a Gaussian noise (17) with mean 0 and standard deviation σ to the normalized image, where σ is a severity level. The resulting image is then denormalized to the original range.

$$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \tag{17}$$

7.1.2 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 kernel. There are two tunable parameters for this corruption type: first one is the readius of the circle in the kernel r, and the second one is the blur strength parameter s. Examples of the kernel with radius r is shown in the Figure below. Such kernel is then simply applied to an image via $cv2.filter2D$ function.

7.1.3 Gaussian Blur

Gaussian blur corruption is another type of blur corruption but with a Gaussian kernel. The center element of such a kernel is the largest one, it corresponds to the peak of Gaussian discribution. This value is descreasing along each side of the center element of the kernel in a symmetrical way. The blur strength parameter s is the standard deviation of the Gaussian kernel. The kernel is again simply applied to an image and one recieves a blurred image as the result. For this corruption a $gaussian()$ function from $skimage.filters$ library was used.

7.1.4 Brightness

Different brightness levels are also an important image corruption to test on, which appears often in the dataset during image accquisition. In oder to change the brightness of the image, the image from the RGB format was translated into HSV format, which stands for hue, saturation and value. This is also one of 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 of the pixels correspondingly. This parameter is often called bias. The bigger absolute value of this change the stronger a corruption will be.

$$\hat{x}_{i,j} = x_{i,j} + s \tag{18}$$

7.1.5 Contrast

In contrast to adding a constant pixelwise to an image to chnage the brightness level, 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.

$$\hat{x}_{i,j} = s * x_{i,j} \tag{19}$$

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

Here are presented the results of all the corruption types and their comparison. One can clearly notice that model is quite stable towards different brightness levels and contrast. Adding more contrast to an image even seems to slightly improve the predictions (as the loss becomes lower).

In order to compare the effect of different corruption types one can calculta the changes in the losses for a subset of the images chosen for the evaluation. The bigger the loss is the worse the prediction is, meaning the stronger the severity of the corruption is. In Figure TODO the comparion of different corruptions with different severities against the loss are presented. The loss is calculated as the Pearson correlation loss for a subset of 10 images that were chosen from the test dataset, therefore the model has never seen these images before.

Howver not only comparing the loss of the network is a good way to evaluate the robustness of the model. One can also compare predcition from corrupted images with the ground truth in terms of the downstream metrics mentioned in the Section TODO. As the model is essetially will be used to predict the downstream metrics, having the robustess estimations for them is as important. The results are presented in Figure TODO.

8 Drift Detection

8.1 Embeddings visualization

Emeddings of the UNet represent themselves a highly compressed and dense information in comparison to a UNet input - an image itself. Visual information (images) is extremely redundant. Having a value of one pixel it is easy to predict the neighboring ones. UNet however compresses this information into a smaller-dimensions representation. In the architecture of the UNet presented in Figure TODO the size of the embeddings in the bottleneck layer of the network is *256x16x16*, meaning that there are 256 filters were applied to a previous layer and the image was compressed from the resilution of *256x256* to *16x16*, meaning that the image was essitially compressed by the factor of *16x16 = 256*. This compression is done by the *bottleneck* layer.

One might wonder why the information from the small objects like Golgi for example won't be lost with such a compression. The answer for this is a receptive field of the network. Receptive field is the region in the input that produces the feature in the hidden layer (cite https://distill.pub/2019/computing-receptive-fields/ Andre Araujo). In the architecture presented the compression happens due to the *MaxPool* layers with a kernel size of *2x2* with a stride of 2. After the first *MaxPool* layer one pixel in the embedding will correspond to *2x2=4* pixels of the input. However already after the second layer of compression one pixel in the embedding will contain *4x4 = 16* pixels of the input. This is how the receptive field of the network increases with the compression while the redundancy of the input information decreases.

To visualize the embedding of the UNet one has to first flatten the bottleneck layer into a vector, in the current network implementation the size of such vector will be *655536* which too high dimesional. In order to comprehend the embeddings better one has to first perfom any dimensionaluty reduction algorithm. One of the options is to compress the vector to still a somewhat high-dimentional vector, however with a fewer dimention, or to compress it up to 2D or 3D-dimentional representation, which can be easily comprehendable by humans. Both compessions are presented in this section.

8.2 Drift detection vs Outliers detection

After the developement phase or a training of the model is finished, it will be moved into a deployment or a production, where it is supposed to maintain an expected quality of predictions. However input data is not always a stable source, therefore one should constantly maintain the quality of predictions and do a regular check-ups for outliers as well as to alert the drift of the data early enough. Drift detection happens on the raw data in absence of the ground truth labels and serves as a signal that the input data differs a lot from the training data.

There is a significant difference between distinguishing drift of the whole source of data in comparison to detecting single outliers. When one talks about a 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.

There are two possible reactions after the drift is detected, one alerts the user that the predictions became unreliable, and therefore one should consider expanding the dataset by adding the labeled data from the drifted distribution to include it in training or apply some different logic on the model outputs. Although when an outlier is detected, the model might request a human assistant for some particular input, because this input is too unfamiliar to the model and it possible won't give a good prediction on this one.

The goal of outliers detection is to decide on the single instances whether or not they are different from training on unusual in some wy or another. They of course might appear both in training and predictions.

Data drift and outlier detection can co-exist. It might be that the input is drifted, but there are no outliers, it might be that there are a lot of outliers, but the data was not drifted. (cite https://towardsdatascience.com/what-is-the-difference-between-outlier-detection-and-data-drift-detection-534b903056d4) But during the production the good approach is to monitor both.

Important observation here is that the drift detector should be robust to outliers. The system should not send an alert as soon as it sees a suspicious sample due to the fact that outlier might be present in the original data distribution as well. But the alert should happen when there are many such samples. To compare original training data distribution and the new one from inputs different statistical tests like Kolmogorov-Smirnov, Chi-squared and etc. can used.

The need of maintaining drift detection or outlier detection depends on the costs on the errors. If the cost of a single error is too high, one should use an outlier detection, but when one needs a test to decide when to label new data - drift detection would be a better approach.

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, whereas the outlier detector aims at finding unusual single instances uin the inputs.

8.3 Drift Detection

Assume that during training labeled data comes from a distribution p, meaning $\{(x_1, y_1), ..., (x_n, y_n)\} \sim p$ and syring deployment unlabeled data comes from a distribution q, meaning $\{x'_1, ..., x'_m\} \sim q$. The goal of the drift detection is to determine if $q(x')$ is the same data distribution as $p(x)$. Or if to put it more formally $H_0 : p(x) = q(x)$ and $H_A : p(x) \neq q(x)$

Having the samples from both distributions or the representation of these samples in lower dimension, one can then choose a statistical hypothesis test to compare these dis-

tributions.

cite https://arxiv.org/pdf/1605.09522.pdf

8.3.1 Kernel methods and two-sample testing

The test for determinig wether two previously mentioned distributions are the same that was used in this work is one of the multivariate kernel two-sample tests: 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 wether the difference in them is statistically significant.

MMD is a kernel-based technique and it allows to distinguish between two distributions based on their mean embeddings (cite and rephrase https://arxiv.org/pdf/1810.11953.pdf) in a reproducing kernel Hilbert space (RKHS).

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. By doing this all powerful kernel methods are allowed to be used for probability measures, resulting in methods like kernel two-sample testing for instance. One of the widely known kernel methods if 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. From cite Smola 2002 if the kernel function if positive definite, then there always exists a dot product space $\mathscr{H}$ as well there exists a function that maps a space from which theses elements are coming from into a mentioned high-dimensional space: $\phi : \mathcal{X} \to \mathscr{H}$ 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. Therefore if there exists an algorithm that can be expressed through the dot product of $\langle x, y \rangle$ cite Schölkopf Smola 1998 then the kernel function can be applied to this dot product and this is called a *kernel trick*.

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:

$$\phi(\mathbb{P}) = \mu_{\mathbb{P}} := \int_{\mathcal{X}} k(x, \cdot) d\mathbb{P}(x) \tag{20}$$

Here $k(x, \cdot)$ is a positive definite symmetric kernel function. Generally the main goal is to map a distribution $\mathbb{P}$ to an 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 $\mathbb{P} = \mathbb{Q}$, which means here that $\mathbb{P}$ and $\mathbb{Q}$ is the same distribution. Additionally the fact that the mapping is injective makes it possible to use such characterization of a distribution to be used in two-sample homogenneity tests, which is exactly what is needed here.

To estimate a the kernel mean embedding is much easier than to estimate the distribution

itself, this is successfully used in data-generating processes as well as it improves some statistical inference methods like two-sample testing again. It is also a helpful approach when instead of a data points for examples of testing and training datasets there are probability distributions.

Inner product $\langle x, y \rangle$ can be viewed as a similarity measure between x and y. This inner product incudes 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:

$$\phi : \mathcal{X} \to \mathcal{F} \tag{21}$$

where

$$\phi : x \to \phi(x) \tag{22}$$

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

$$k(x, y) := \langle \phi(x), \phi(y) \rangle_{\mathcal{F}} \tag{23}$$

with

$\langle \cdot, \cdot \rangle_{\mathcal{F}}$ denoting an innter product in of $\mathcal{F}$.

Now $k(x, y)$ is already a non-linear similarity measure between x and y. Now to get a non-linear version of the algorithms that use dot product, $\langle x, y \rangle$ can be simply sustituted 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 8.1 (Kernel mean embedding)**.** cite Berlinet and Thomas Agnan 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

$$\mu : M_+^1(\mathcal{X}) \to \mathscr{H}, \mathbb{P} \to \int k(x, \cdot) d\mathbb{P}(x) \tag{24}$$

However usually there is no access to the distribution $\mathbb{P}$ and that is why one cannot directly compute $\mu_{\mathbb{P}}$. Fortuntely 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 the tru kernel mean embedding $\mu_{\mathbb{P}}$. One of such approximations could be the following unbiased estimate:

$$\hat{\mu}_{\mathbb{P}} := \frac{1}{n} \sum_{i=1}^{n} k(x_i, \cdot) \tag{25}$$

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

Definition 8.2 (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 the 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 8.3 (IPM)**.** Let $\mathbb{P}$ and $\mathbb{Q}$ be two probability measures on some measurable space $\mathcal{X}$. Then IPM is defined as follows:

$$\gamma[\mathcal{F}, \mathbb{P}, \mathbb{Q}] = \sup_{f \in \mathcal{F}} \left\{ \int f(x) d\mathbb{P}(x) - \int f(y) d\mathbb{Q}(y) \right\} \tag{26}$$

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

8.3.2 Maximum Mean Discrepancy

Let's assume that $\mathcal{F} := \{f \mid \|f\|_{\mathscr{H}} \leq 1\}$, it means that the supremum in Definition 8.3 is taken over functions in the unit ball in RKHS, then the mean embedding metric will be called *maximum mean discrepancy* (MMD).

Definition 8.4 (IPM)**.** Maimum mean discrepancy is defined as a distance between two mean embeddings of distributions:

$$\begin{aligned} \text{MMD}[\mathscr{H}, \mathbb{P}, \mathbb{Q}] &= \sup_{\|f\| \leq 1} \left\{ \int f(x) d\mathbb{P}(x) - \int f(y) d\mathbb{Q}(y) \right\} \\ &= \sup_{\|f\| \leq 1} \{ \langle f, \mu_{\mathbb{P}} - \mu_{\mathbb{Q}} \rangle_{\mathscr{H}} \} \\ &= \|\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}\|_{\mathscr{H}} \end{aligned} \tag{27}$$

From 8.4 if $\mathscr{H}$ is characteristic follows that $\text{MMD}[\mathscr{H}, \mathbb{P}, \mathbb{Q}] = 0$ if and only if $\mathbb{P} = \mathbb{Q}$.

Similarily to the estimation of the kernel mean embedding through the samples drawn from a distribution, a biased empricial 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:

$$\widehat{\text{MMD}}_b^2[\mathscr{H}, X, Y] := \sup_{\|f\|_{\mathscr{H}} \leq 1} \left\{ \frac{1}{n} \sum_{i=1}^{n} f(x_i) - \frac{1}{m} \sum_{j=1}^{m} f(y_j) \right\} \tag{28}$$

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

$$\widehat{\text{MMD}}_b^2[\mathscr{H}, X, Y] = \|\widehat{\mu}_{\mathbb{P}} - \widehat{\mu}_{\mathbb{Q}}\|_{\mathscr{H}}^2 \tag{29}$$

However if an unbiased estimator of MMD through the kernel function k is needed then one could use the following estimator from cite Borgwardt et al 2006 Corollary 2.3

$$\begin{aligned}\widehat{\mathrm{MMD}}_b^2[\mathscr{H}, X, Y] =& \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i}^{n} k(x_i, x_j) \\ &+ \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \neq i}^{n} k(y_i, y_j) \\ &- \frac{2}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} k(x_i, y_j)\end{aligned} \tag{30}$$

The most common application of MMD in statistics is a two-sample testing. Particularily, 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 cautios here, even when both samples came from the same distribution it still might be that the MMD will be not zero due to the fact that this is an etimate and not a precise value and we have a finite number of smaples for this estimate.

RKHS is defined by a positive definite kernel in (22) RKHS methods now can be applied to probability measures, which results in finding out many useful applications one of which is kernel two-sample testing. Such a kernel is then also called a reproducing kernel.

8.4 Practical application of MMD

The drift detection of corrupted samples for the problem of the thesis has been performed using Alibi DETECT open source library, tha focuses specifically on outlier, adversarial and drift detection algorithms (cite https://github.com/SeldonIO/alibi-detect). Which implements statistical hypothesis testing algorithms for detecting drifts in data.

In works in the following way, before observing some data, one can specisfy the null hypothesis H_0 and the alternative hypothesis H_1 about the generating process behind the data (its distribution for ex.) and also specofy the test statistics $S(X)$ that one expects to be small under the hypothesis H_0 and large under the hypothesis H_1. Then during the observation of the new data after computing the the value of this test statistic $S(X)$, one computes $p = P(S(X)|H_0)$ which is called p-value - a probability that such an extreme value of the test statistic could habe been observed under the null hypothesis. If this probability if below the threshold t, then the assume that this is a drifted data, and not otherwise. So low p-value refuses the null-hypothesis.

8.4.1 Online MMD

More specific an online MMD detector was used for etimating wether the new samples are drifted from the training distribution. The radial basis kernel function (Equation 31)

is used there by default, however there are also options to use any kernel function of preference.

$$k(x, y) = exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right) \tag{31}$$

One os the charateristics of an online drift detector is that it assumes that there is a big dataset of a reference data, that can be used to training as an example of a "correct" distribution. Online algorithms focus on single input at a time during its run. This single input would be sent into a test window where two-sample test-statistics (MMD essentially in this case) will be calculated. As soons as the test-statistic exceeds some pre-defined threshold the drift alert is send to a 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 hyper parameter here is hidden in a size of a test-window. Becase the larger the window is, the more chance is there to detect a very slight drift, however 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. However in this case the performance was exceptional even for high-dimensional embeddings of the UNET model. However there are also options on how to reduce the dimensionality of the data from the authors of the framework. They recommend even to use an untrained AutoEncoder (UAE), as even an untrained one contains some relatively good information about the global structure of the data. (cite someone?)

9 Summary

List of Figures

List of Tables