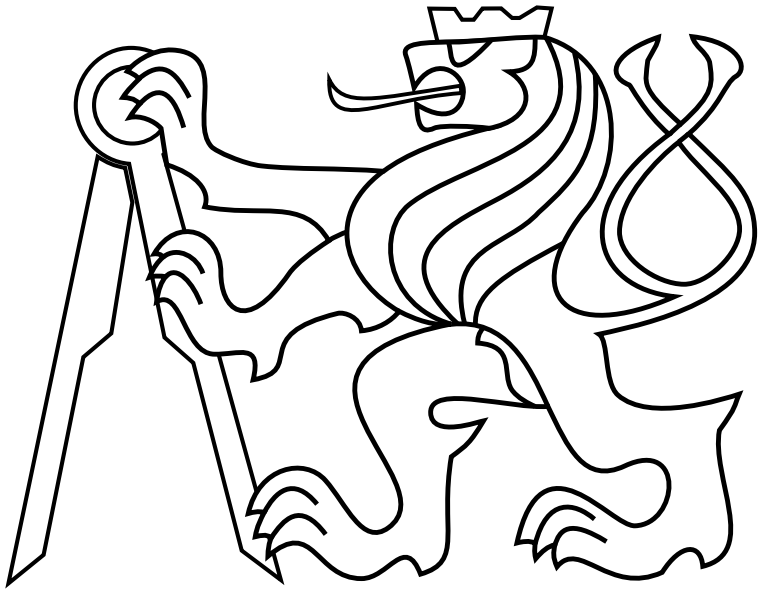
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

DIPLOMA THESIS



Hana Mertanová

Cell segmentation in microscopy using a reference modality

Department of Cybernetics

Thesis supervisor: **prof. Dr. Ing. Jan Kybic**

May 2021

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Mertanová Hana** Personal ID number: **465875**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Open Informatics**

Specialisation: **Artificial Intelligence**

II. Master's thesis details

Master's thesis title in English:

Cell segmentation in microscopy using a reference modality

Master's thesis title in Czech:

Segmentace buněk v mikroskopii za použití referenční modality

Guidelines:

1. Get familiar with modern methods for image segmentation, especially using convolutional neural networks.
2. Get familiar with our datset of fluorescence and and phase contrast microscopy images of cells and their nuclei, used to identify the effect of drugs on the
cells.
3. Design, implement and test a classical algorithm (e.g. based on thresholding) for segmenting nuclei from the fluorescence images.
4. Design, implement and experimentally evaluate a deep learning algorithm to identify nuclei from the phase contrast images, using the fluorescence image
or its segmentation as a reference.
5. Aim to improve the segmentation quality by suitable techniques such as distance function regression, positive/unlabelled (PU) learning, spatial regularization, or more powerful network architecture.
6. Design, implement and experimentally evaluate a pix2pix inspired algorithm for segmenting the cell nuclei.
7. [Optional:] evaluate shape features from the segmentations and evaluate their usefulness for the task of detecting the drug effects.

Bibliography / sources:

1. Goodfellow, Bengio, Courville: Deep learning book – MIT Press, 2016.
2. Ronneberger, Olaf, Philipp Fischer, and Thomas Brox.: U-net: Convolutional networks for biomedical image segmentation. International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015.
3. Isola, Phillip, et al.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. p. 1125-1134.
4. Naylor, Peter, et al.: Segmentation of nuclei in histopathology images by deep regression of the distance map. IEEE transactions on medical imaging 38.2 (2018): 448-459.
5. Kiryo, Ryuichi, et al.: Positive-unlabeled learning with non-negative risk estimator. Advances in neural information processing systems. 2017 Estimator.

Name and workplace of master's thesis supervisor:

prof. Dr. Ing. Jan Kybic, Biomedical imaging algorithms, FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **21.02.2021** Deadline for master's thesis submission: **21.05.2021**

Assignment valid until: **19.02.2023**

prof. Dr. Ing. Jan Kybic Supervisor's signature	Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Author statement for undergraduate thesis

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date ..... Signature .....

Acknowledgements

I would like to thank my supervisor Jan Kybic for giving me an opportunity to work with the neural networks, for his help, guidance, and wonderfully fast and excellent feedback. We appreciate the access to the data thanks to Ústav molekulární a translační medicíny Lékařské fakulty Univerzity Palackého (www.imtm.cz). I am grateful to my family for always supporting me and enabling me to seek an education. I also thank Vašek for encouraging me at all times with great patience and kindness.

Abstract

Microscopy allows us to study and understand biological processes on the cellular level and observe the biological structures. In this thesis, we aim to observe the drug effects on the cell population. Moreover, we attempt to automate the observation due to a large number of cells.

It is desirable to use phase-contrast microscopy instead of fluorescence microscopy because the phase-contrast microscopy is less laborious and applicable in-vivo. Therefore, this thesis focuses on nuclei segmentation in phase-contrast microscopy images. The common approach is to learn from the data annotated by the experts. Instead, we have access to the corresponding fluorescence microscopy modality.

We thus create the reference segmentation automatically and then apply the deep learning techniques.

We exploit multiple network architectures for segmentation, including the UNet ([41]) and the image-to-image translation technique ([25]). We improve the segmentation by changing the settings such as loss function and inspect the problems such as insufficient coloring in the reference modality. We perform quantitative analysis and select the best performing model, achieving the sensitivity of 88.8% and precision of 88.5% .

Finally, we evaluate the attributes describing the shape, amount, and size of the cells to identify the potential changes induced by the treatment. These descriptors serve as an input for an SVM classifier. Ultimately, the classifier predicts whether the treatment influences the cells' appearance.

Keywords: Phase contrast microscopy segmentation, Deep learning, Neural networks, Treatment classification, Image-to-image translation, Localization, Segmentation without expert annotations

Abstrakt

Mikroskopia nám umožňuje študovať a porozumieť biologickým procesom na bunkovej úrovni a pozorovať biologické štruktúry. V tejto práci sa zameriavame na pozorovanie vplyvu určitých látok na populáciu buniek. Naviac sa snažíme automatizovať pozorovanie kvôli veľkému množstvu buniek.

Je žiaduce použiť fázovú kontrastnú mikroskopiu namiesto fluorescenčnej mikroskopie, pretože fluorescenčná mikroskopia je menej pracná a je možné ju používať in-vivo. Preto sa táto práca zameriava na segmentáciu vo fázovej kontrastnej mikroskopii.

Bežný prístup je učiť sa pomocou dát anotovaných expertmi. My namiesto toho máme prístup ku korešpondujúcej fluorescenčnej mikroskopii.

Preto vytvárame referenčnú segmentáciu automaticky a následne aplikujeme metódy hlbokého učenia.

Využívame viacero architektúr neurónových sietí na segmentáciu, napríklad UNet ([41]) a techniku 'image-to-image translation' [25] (priamu techniku transformácie jednoho obrázku na druhý). Aby sme zlepšili segmentáciu, meníme rôzne nastavenia, ako napríklad stratovú funkciu a vyhodnocujeme problémy, ako napríklad nedostatočné farbenie v referenčnej modalite.

Kvantitatívne analyzujeme výsledky a vyberáme najlepší model, dosahujúci senzitivitu 88.8% a presnosť 88.5%.

V závere vyhodnocujeme atribúty popisujúce tvar, množstvo a veľkosť buniek, aby sme identifikovali potenciálne zmeny spôsobené aplikáciou liečív. Tieto deskriptory slúžia ako vstup pre SVM klasifikátor. Nakoniec klasifikátor predikuje, či aplikácia liečiva ovplyvňuje vzhľad buniek.

Kľúčové slová: Segmentácia fázovej kontrasnej mikroskopie, Hlboké učenie, Neurónové siete, Klasifikácia liečív, Image-to-image translácia, Lokalizácia, Segmentácia bez expertných anotácií

Contents

List of Figures

List of Tables

1 Introduction

Microscopy allows us to analyze biological processes and structures. Hence, we can observe various treatment effects on individual cells. In order to study these processes, it is desirable to automate the detection of the biological structures, as manual labeling is costly, and we want to process large amounts of data. Our aim is to distinguish individual cells and identify the changes induced by different treatments.

Fluorescence microscopy is often used for similar tasks ([11], [49], [21]) , because it provides nicely distinguishable structures. However, fluorescence microscopy has multiple disadvantages, such as difficult signal reproducibility, photo bleaching and the phototoxicity ([54], [50]). Moreover, fluorescence microscopy requires an application of the fluorophore on the cells.

On the other hand, phase-contrast microscopy images are harder to distinguish due to artifacts, deformable shapes, and low contrast ([50]). Thus, segmentation of the contrast microscopy images is a challenging task. Nonetheless, phase contrast microscopy does not require any special substance application and can be used in-vivo.

Therefore, it is beneficial to recognize cells in the phase-contrast microscopy images, as it lessens the biologists' efforts and saves time and financial sources. Our goal is to create a segmentation method for phase-contrast microscopy image cell segmentation. However, we do not have access to reference segmentation from experts. Instead, the model will learn from the fluorescence microscopy images.

The organization of this thesis is the following: We first describe the input images and the task in Chapter 2. Then we propose a solution for automatic nuclei detection in contrast microscopy images, using the fluorescence microscopy modality (Chapter 3).

After that, we provide a brief overview of existing methods related to our work in Chapter 4, such as segmentation, deep learning and image processing with minimal annotation.

We then proceed to individual steps of a solution for the contrast microscopy segmentation learning. The first step of the solution is to create a reference segmentation from the fluorescence image modality (more details in Chapter 5). That is, we apply multiple classical segmentation methods and select the best suited for our problem.

We continue with the learning of the segmentation from the contrast microscopy images. In these images, the nuclei are harder to recognize, and classical methods

are not so suitable for segmentation compared to the fluorescence modality. As we create the reference automatically, we have a large training set that allows us to focus on the deep learning methods.

Namely, we briefly describe and apply the UNet-based ([41]) architectures to our problem (Chapter 6). That means we try to learn to segment the contrast microscopy images while using the segmentation obtained from the fluorescence modality as a reference.

As a result, we design a pipeline for the dataset preprocessing, network training, and test-set evaluation. After that, we provide a quantitative analysis of the neural network output. We also inspect the impact of different network setting on neural network performance (Section 6.3). Moreover, we provide a comparison of the output segmentation successfulness on different datasets in Section 6.3.5.

Finally, we improve the neural network model with an image-to-image translation ([25]) deep learning method. We briefly describe the main idea and design a pipeline for nuclei segmentation in Chapter 7. We observe the impact of multiple model settings on neural network performance. We provide an analysis of the results and a comparison with previously used methods in Section 7.8.

We conclude the thesis by evaluating the drug effects on the cell nuclei in Chapter 8. We extract the attributes describing the shape, size, and amount of the nuclei in an image. We then train the classifier that predicts the biological treatment based on these extracted features. Ultimately, the classifier learns to distinguish between the treatment. We also find the nuclei attributes that are affected by the treatment.

2 Task Specification

We aim to segment the cell nuclei in the phase-contrast microscopy images using fluorescence images as a reference.

The main advantage of the fluorescence microscopy image is that, unlike in contrast microscopy image, the nuclei are nicely recognizable, even by a human eye and thus more easily segmented by the traditional methods. Figure 2.1b shows the example of the fluorescence microscopy image.

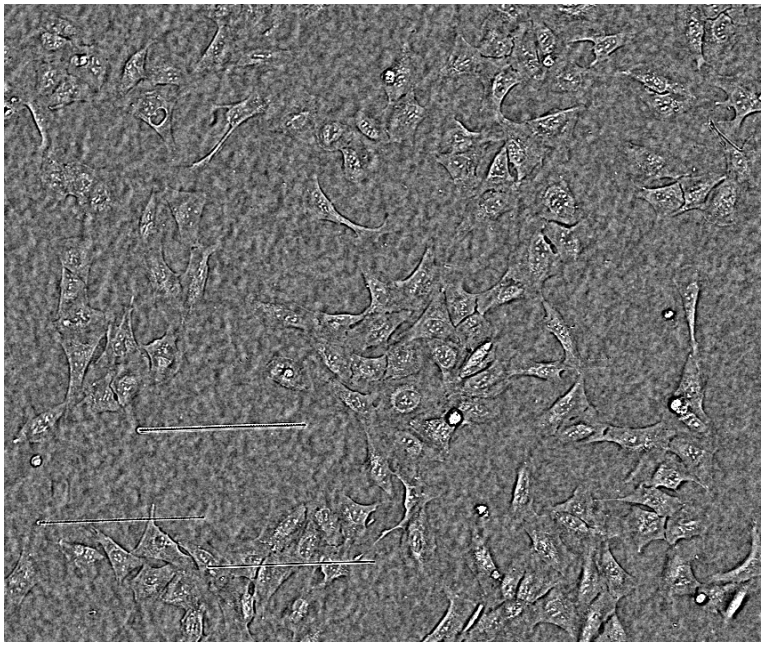
On the other hand, a special chemical treatment needs to be applied in order to obtain the fluorescence microscopy images. Moreover, fluorescence microscopy has multiple disadvantages, such as difficult signal reproducibility, photo bleaching and the phototoxicity ([54], [50]. If we were able to recognize the nuclei in contrast microscopy, we could avoid the nuclei coloring.

The main advantage of phase contrast microscopy is the observation of the cells without the necessity to apply any special substance. On the contrary, the images are often hard to process. Segmentation of the contrast microscopy images is challenging due to the unclear images with various artifacts and indistinct nuclei borders, presence of the shadow-cast artifacts, halo, and shade-off effects ([50]) , as well as high density, low contrast or deformable cell shapes.

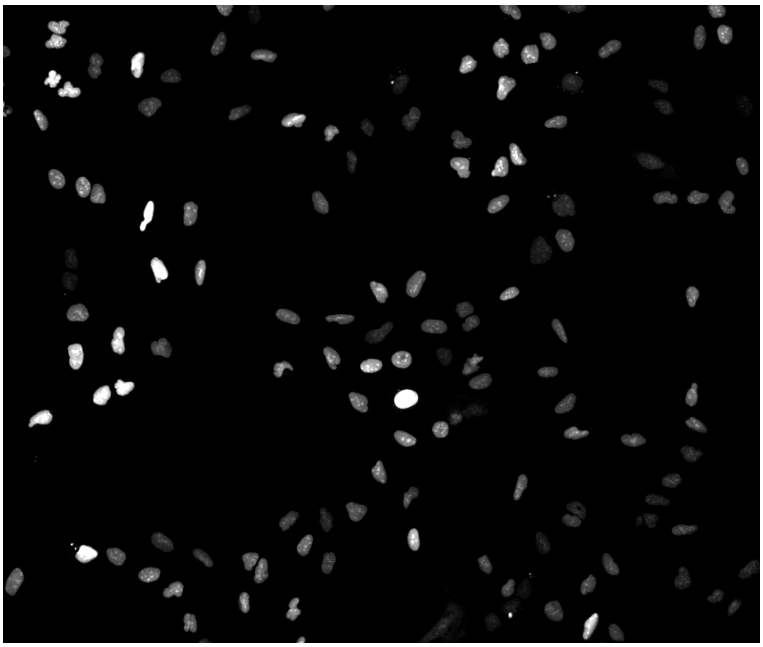
We show the example of the contrast microscopy image in Figure 2.1a .

Moreover, we do not possess precise reference segmentations (annotated by experts). Instead, we have the fluorescence microscopy images corresponding to the contrast microscopy. Both images of the tuple of the corresponding contrast microscopy image and the fluorescence microscopy image show the same view captured by a different method.

In the following section, we will describe our dataset.



(A) Contrast microscopy image



(B) Fluorescence microscopy image

FIGURE 2.1: Tuple of input images example

2.1 Dataset Describtion

Our dataset consists of four categories. Each dataset name refers to the different experimental phases. That is, the number of cells inserted into the experiment, the status of the untreated cells ('1500cells before') and the cells treated in a time interval.

We expect the differences among the datasets, namely in the size and shape of the nuclei, especially in the late phase of the experiment ('1500cells 72h'). The late phase of the experiment might also lead to lower numbers of the identified cells.

The dataset categories are the following:

1. 1500cells 24h: Cells with treatment, seeded 1500 cells per well, imaging done in time point 24 hours.
2. 1500cells 72h: Cells with treatment, seeded 1500 cells per well, imaging done in time point 72 hours.
3. 1500cells before: Cells without any treatment, seeded 1500 cells per well, imaging done in time point 0 hours.
4. only cells 2: Cells without any treatment, 1500 cells per well, imaging is done 24 hours following the seeding – large dataset 44 wells with non-treated cells. It can be used as a control for the teaching of nuclei identification.

We will refer to the presented categories as to the '24h' dataset, '72h' dataset, 'before' dataset, and 'only' dataset. Each dataset contains the tuples of a contrast microscopy image and a fluorescence microscopy image of size 2160 x 2560 pixels.

The images in each dataset can be further divided by the type of treatment (Topotecan, Daunorubicin, Etoposide, DSMO, or no treatment). The image names consist of three parts: A_B_C.png, where A is the name of the dataset and B represents the treatment. The treatment type B is encoded as follows:

1. Topotecan is represented by: C02-C05, D02-D05.
2. Daunorubicin is represented by: C14-C17, D14-D17.
3. Etoposide is represented by: C06-C09, D06-D09.
4. DMSO is represented by: C18-C21, D18-D21.
5. No treatment is represented by: C10-C13, D10-D13, C22-C23, D22-D23.

	Topotecan	Daunorubicin	Etoposide	DMSO	no treatment	total
’24h’	40	40	39	39	60	218
’72h’	40	40	40	39	59	218
’before’	40	40	40	39	60	219
’only’	40	40	40	40	60	220
total	160	160	159	157	239	875

TABLE 2.1: The table shows the number of the treatment occurrences in each dataset.

3 Proposed Solution

In this chapter, we will propose the solution to the task presented above. The brief outline of the solution consists of multiple steps:

1. As the datasets do not contain any reference, the first step is to obtain it. Thus, we try multiple segmentations on the fluorescence microscopy images and manually evaluate the successfulness of the distinct segmentation methods. Fortunately, the nuclei in the fluorescence microscopy images are nicely distinguishable.

2. - The second step is to segment the contrast microscopy images using the fluorescence image segmentation as a reference (Section 5).

     As we create the reference automatically, we take advantage of the large training set and use the neural networks to learn the segmentation. Moreover, the images in our dataset are too large to be processed at once by the networks we propose due to the memory limit. Therefore, we further expand the dataset size by the image slicing.
   - We will try two neural network architectures. The first architecture is widely used UNet ([41]), which we examine closely in Section 6. The second approach is image to image translation ([25]), which we introduce in Section 7.

3. Finally, we will provide the quantitative analysis of the segmentations given the automatically created references. We will also compare the architecture settings and try to improve the segmentation by various network alternations (Sections 6.3.3, 7.5). Then we will discuss the success of provided solution on different datasets (Sections 6.3.5, 7.6). We also calculate shape descriptors based on the segmentations and see if the different treatments can be distinguished (Section 8).

We provide the implementation details in Appendix A (README). We also enclose the implementation.

4 Existing Methods

In medical image processing, there are multiple problems that we encounter. Thus, we present the existing methods for these tasks solving in the following sections.

4.1 Segmentation

Segmentation is one of the essential tasks in image processing. It assigns each class (for example, human) or each instance of the class (individual people) in the image a label.

We distinguish two main areas of image segmentation. First, we know the traditional segmentation methods such as edge detection filters or mathematical methods (see Sonka et al. [47] for a detailed presentation of such methods). We also describe some of the traditional segmentation methods in Section 5. Second, we are aware of the deep learning approaches.

Nowadays, deep learning methods are emerging rapidly, and they are a leading machine learning tool in image analysis and computer vision ([34]). Deep learning improves the accuracy of various tasks, such as segmentation, detection, or recognition ([34]). Specifically, the convolutional neural networks reach great success in medical image analysis and other computer vision tasks ([20], [34]).

Figure 4.1 extracted from [53] shows example of segmentation of breast histopathology images.

Conventional neural networks ([9], [23]) typically consist of convolutional layers, activation layers with pooling layers (max-pooling or average pooling), followed by a fully connected layer.

There is a great number of different neural network architectures (we encourage a reader to see a compact Table 2 in [16]).

Schmidhuber ([42]) provides a historical overview of neural networks. LeCun et al. ([30]) show a review of supervised learning, especially CNNs and recurrent neural networks. Book ([17]) describes several established steps in deep learning.

We decided to briefly describe the following three networks that are considered widely known standards (for more, see [23], [14]):

1. AlexNet [27]: Alex net is a deep convolutional neural network that won the ILSVRC-2015 with a TOP-5 test accuracy of $84,6\%$. Architecture is relatively simple ([27], [14]), consisting of five convolutional layers, max-pooling layers, rectified linear units (ReLu), nonlinearities, three fully connected layers, and a

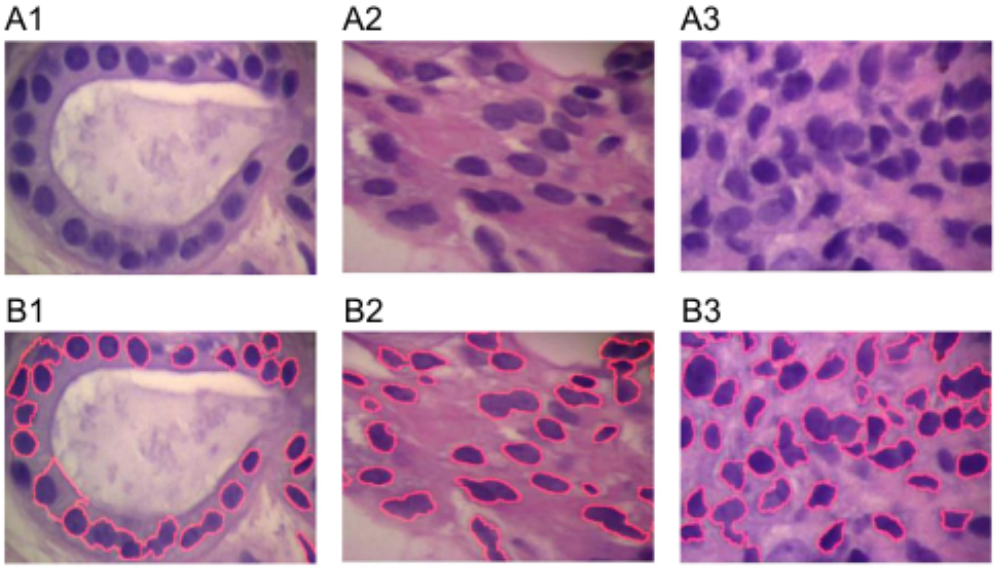


FIGURE 4.1: The figure extracted from [53] shows the segmentation results of breast histopathology images. (A1-A3): Original color breast images, (B1-B3) corresponding results of markercontrolled watershed based on adaptive H-minima algorithm for markers selection in [5].

dropout layer. Figure 4.2 extracted from [27] illustrates the AlexNet architecture.

2. ResNet [22]: ResNet is a neural network model that won ILSVRC-2016 with 96% accuracy [14]. ResNet is rather deep, consisting of 152 layers. It introduces the residual blocks (as presented in Figure 4.3, extracted from [22]). These residual blocks attempt to solve the problem of deep network training by skip connections. The main idea is that the next layer learns something different from what the input already encoded.

3. Re-Net [51]: Standard recurrent networks (RNN) contain recurrent connections which enable the network to memorize the patterns from the last inputs ([23]). Thus, RNNs are able to extract inter-slice contexts.

   Unlike standard recurrent networks, multi-dimensional RNN substitutes each recurrent connection with multiple connections ([8]). Visin ([51]) proposed ReNet that uses sequence RNN instead of multi-dimensional RNN. Each convolutional layer (convolution + pooling) is replaced by four RNNs that sweep the image horizontally and vertically, as shown in Figure 4.4 (extracted from [51]).

Another important method is transfer learning, which combines a pre-trained network with yet untrained layers. Thus, we can use the knowledge gathered in a previous domain for a novel task ([45]). The main idea is that the first layers of the pre-trained network might extract the important features from which the rest of the NN learns better. Naturally, transfer learning provides better results when the tasks are similar.

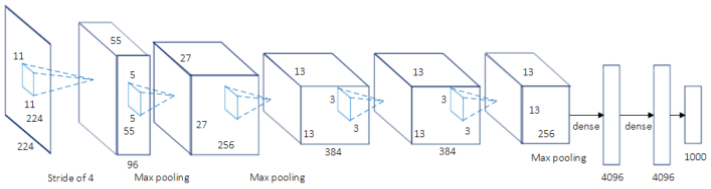


FIGURE 4.2: The figure shows AlexNet Convolutional Neural Network architecture. Figure reproduced from [27].

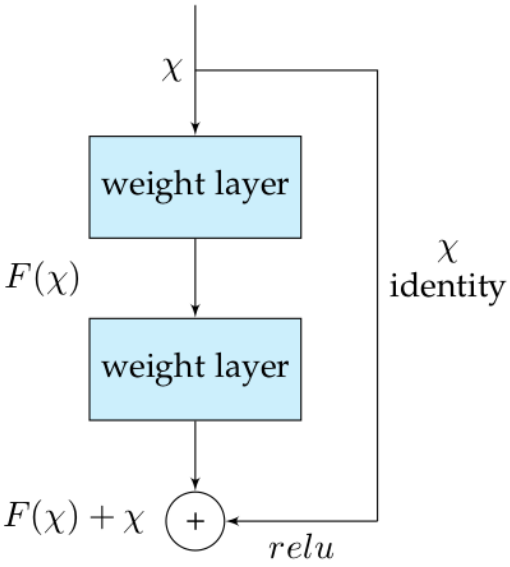


FIGURE 4.3: The figure shows the residual block from the ResNet architecture. Figure reproduced from [22].

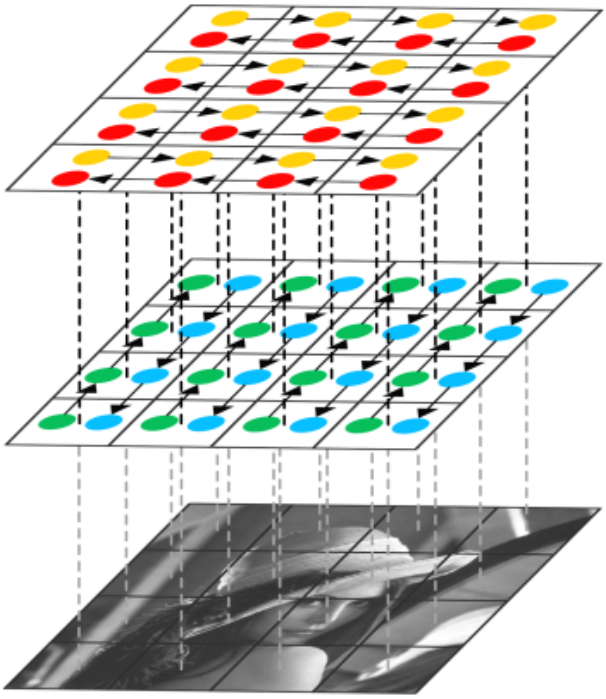


FIGURE 4.4: The figure shows one layer of ReNet architecture modeling vertical and horizontal spatial dependencies. Extracted from [51].

We also discuss UNet ([41]) in Section 6 and Image-to-image translation ([25]) in Section 7 . The main disadvantage of deep learning is a necessity of a large training set of annotated data, which is expensive in the medical domain ([20], [34]).

Therefore, learning from minimal annotation is desirable.

4.2 Learning from Minimal Annotations

Precise segmentation, localization, and counting are slow, costly, and error-prone if medical researcher performs these tasks ([33], [10]). Thus, automation is highly desirable.

Nonetheless, automation brings multiple challenges, such as object overlapping or shape variation. Moreover, to automate the presented tasks, we need the labeled images, which are uncommon in the microscopy area. Typically, only a few images contain ground truth. Thus the classical CNN approaches become ineffective ([33], [6]).

However, few papers deal with the limited training data. For example, Matuszewski et al. ([37]) designed a U-Net-based approach. Nonetheless, additional assumptions were needed, such as circular shape cells and perfect ground truth for at least hundreds of training images. The methods presented in the paper were the following: First, the training set was augmented by mirroring and rotations. As

only the centers of the cells were annotated, the ground truth had to be approximated. For the approximation, the centers were dilated by a structure element with an empirically observed size. Matuszewski et al. design a U-Net that predicts only obvious background/foreground, leaving the uncertainty regions to be decided by an underlying model.

Lomanov et al. [33] were able to improve models in the presence of incomplete and inaccurate annotations. As a result, they were able to train a UNet with only twelve images. Lomanov et al. discuss the influence of incomplete data on localization and segmentation.

Feng et al. ([12], [23]) used a CNN for a fully automated segmentation of lung nodules in weakly annotated data. Their method is based on the results from ([56], [23]), where the capability of CNN to identify the discriminative regions was demonstrated.

To sum up, a lack of annotated data can be countered by data augmentation, or introducing background knowledge, or creating an underlying model.

4.3 Automatic Image Analysis

Automatic image analysis helps us overcome subjective interpretation and reduce the workload of researchers.

We encouter numerous classification algorithms in literature, e.g. SVM ([15]), fuzzy c-means ([35]), Naive Bayes ([13]), k-NN ([1]), neural networks ([15]), decision trees and partial least squares regression ([55]).

These methods typically learn from the features extracted from the image, such as size, shape, or color of objects. It is desirable that the features are meaningful and show discriminative ability.

Moreover, it is desirable to have an ability to classify with a small number of features as it is easier to obtain them, and we also avoid overfitting.

Feature selection is a useful tool that selects the subset of features so that the combination of selected features has a high discriminative ability. The selection can improve the results from the classifier ([13],[28]).

For example, a diagnosis from histopathological images helps to detect breast cancer ([53]). This detection is based on feature extraction. For the purposes of the paper ([53]), the features are either shape-based or textural, based on color spaces.

5 Fluorescence Images Segmentation

The first step to solve the task presented in Section 2 is a segmentation of the fluorescence images. Although the nuclei in the fluorescence microscopy images are mostly quite well distinguishable, we need to segment them in order to create the reference images for the neural network.

In this chapter, we will therefore discuss various segmentation methods. We will then apply discussed methods on out datasets and compare the results of selected approaches. As we do not have any ground reference segmentation, we evaluate the results manually. That is, we combine the fluorescence images with the output segmentation of a particular method and see whether the segmentation corresponds to the brighter ellipsoid regions. Fortunately, the nuclei are easily distinguishable to the human eye. In the following sections, we will present the selected segmentation methods.

5.1 Otsu Segmentation

Otsu segmentation is a classical segmentation method introduced by Otsu in [39]. The main idea is the following: The pixel values in the image belong either to the foreground or to the background. Thus, if we construct the histogram of pixel values, there should be a valley between low values and high values.

In order to determine the threshold, Otsu tries to minimize the variance of both foreground pixel values and background pixel values. Therefore, the criterium minimized by Otsu is weighted within-class variance.

5.1.1 Results

We used the Otsu segmentation from the scikit-image library ([52]). Although we properly segmented some of the nuclei, the output segmentation was often incomplete. Namely, some of the nuclei were partially under-segmented or were not continuous.

Thus, we tried to improve the segmentation by first applying smoothing with the gaussian filter. We used the Otsu and filtering from the OpenCV library ([4]) for

this improvement. Although the results slightly improved, the difference was not so significant. We still encountered many under-segmented images.

Figure 5.1b shows the Otsu output segmentation example.

5.2 Mean Thresholding

In this section, we will introduce a rather simple and straightforward method. We noticed that images contain far more background pixels than foreground pixels. Thus, the mean is just a little higher value than the background value, and it serves as a quite nice boundary.

5.2.1 Results

Mostly, the results looked reasonable. This method's main drawback is that it does not work very well for images containing too low numbers of nuclei. In such images, the proposed method partially over-segmented the nuclei neighborhood.

In order to improve the results, we tried applying the closing operation on the output segmentation. As a result, the partial over-segmentation around the nuclei vanished almost in every image.

Figure 5.1c shows the example of the obtained result.

5.3 Quantile Threshold

As the method from section 5.2 reached nice results, it inspired us to also try the median and other quantile values as the threshold. The median value is often more robust than the mean value.

5.3.1 Results

Although the method seemed promising initially, it did not reach such good results as the previous methods. Setting the threshold to the median image value overally did not work.

On the contrary, the different quantile selection yielded better results than the median. However, neither setting the threshold to the different quantiles did not reach the performance comparable to mean thresholding in Section 5.2.

The main problem is that the quantiles were not universal enough. That is, when the quantile-based method segmented some of the images nicely, the rest was either under-segmented or over-segmented.

Moreover, both median/gaussian filtering and closing operation was required to achieve good results. We present the example segmentation in Figure 5.1d.

5.4 Combination of Mean and Quantile Thresholds

In order to use the advantages of both approaches presented in Sections 5.2, 5.3, we tried to apply the convex combination of these. Namely, we used the 0.8 quantile for it reached the best results out of all quantiles tried.

5.4.1 Results

Although the results were slightly better than either pure mean or quantile thresholding, the closing operation was still required. Figure 5.2a shows the example result of the combined thresholding.

5.5 Maximally Stable Extremal Regions

Maximally stable extremal regions (MSER) were introduced by Matas et al. ([36]). The original purpose was to establish correspondences between the images.

5.5.1 Results

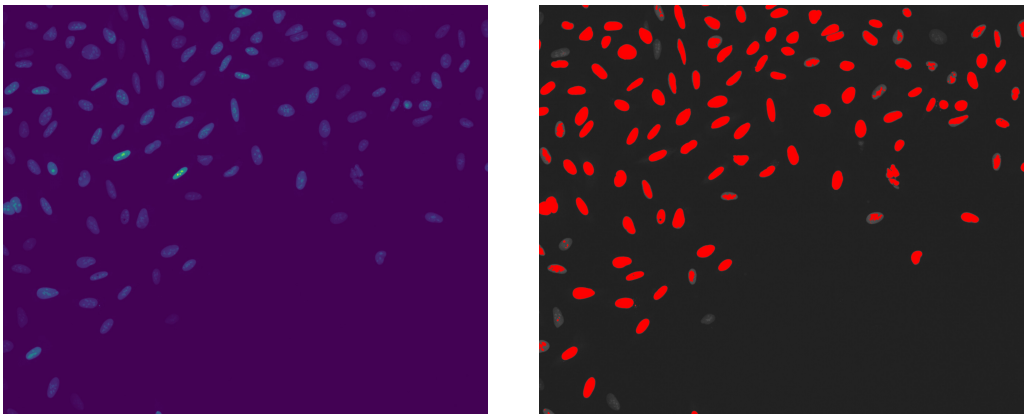
We used the implementation of MSER from the OpenCV library ([4]). Although the method was quite successful on some of the images, on most of the images, it did not recognize some of the nuclei. Figure 5.2b shows the example result.

5.6 Fixed Global Threshold

The proposed global thresholding method is the following: We segmented images by the mean thresholding described in Section 5.2. After that, we manually selected the image with a perfect segmentation. Consequently, we inspected the lowest pixel value in the selected image marked as foreground. Ultimately, we set this lowest pixel value as the global threshold for all images.

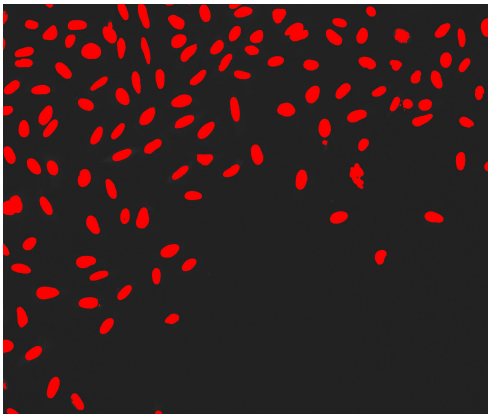
5.6.1 Results

The results of the proposed global thresholding are surprisingly good. The reason for this behavior might be that the fluorescence microscopy images have similar intensities of the marked nuclei. We show the example segmentation in Figure 5.2c .

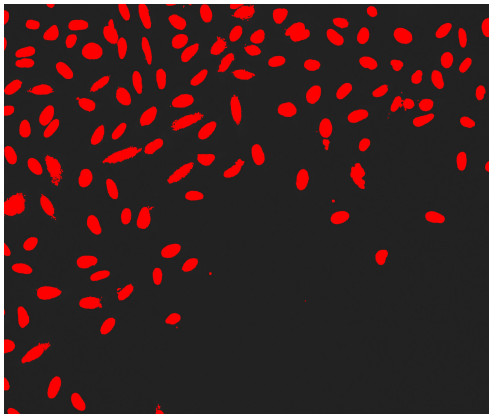


(A) Original Image

(B) Otsu

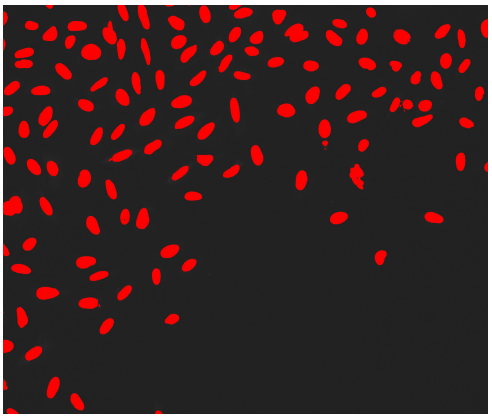


(C) Mean Threshold

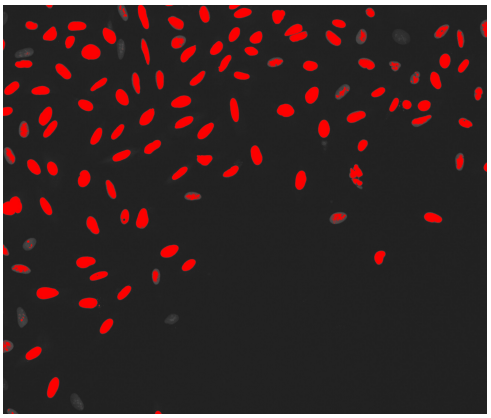


(D) Quantile 0.8 Threshold

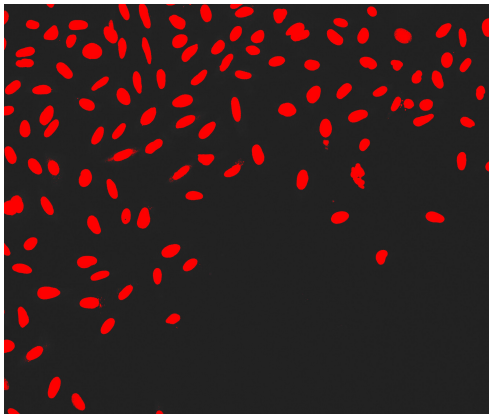
FIGURE 5.1: Figure shows the segmentation methods 5.1 - 5.3.
Subfigures show the output segmentation in red combined with the original fluorescence image.



(A) Mean and Quantile Combined Threshold



(B) Maximal Stable Region



(C) Fixed Global Threshold

FIGURE 5.2: Figure shows the segmentation methods 5.4 - 5.6.
Subfigures show the output segmentation in red combined with the original fluorescence image. More segmentation examples can be found in Appendix B (folder results_segmentations).

5.7 Final Evaluation

In the previous sections, we briefly described the methods used for segmentation. Now we will compare them to obtain the best segmentation.

The most successful methods were undoubtedly threshold as the combination of the mean and quantile and fixed global thresholding. Thus, we ran these methods on all of the images and checked the individual datasets' results.

The results from both methods were really similar. However, the global threshold was almost always slightly better than the mean and quantile combination.

Therefore, we selected the fixed global thresholding as the final segmentation method to create the neural network training references. In the following chapter, we will proceed to the next part of the task, which is finding the nuclei in the contrast microscopy images. We will use the segmentation obtained from fluorescence microscopy as a reference for the training of the neural networks.

6 UNet Architecture Application

In this chapter, we will first introduce the UNet ([41]) architecture, which is massively used in various biomedical segmentation tasks (i.e.; [44], [31], [57]).

After that, we will discuss a loss function in Section 6.2. Ultimately, we will analyze the obtained segmentations. Also, we will compare various settings of the loss function and other factors. We will then proceed to the quantitative analysis of results and conclude by the distinct datasets comparison.

6.1 UNet Introduction

In this section, we will briefly describe the UNet architecture (proposed in [41]). UNet is a fully convolutional neural network. It consists of the following two parts:

1. Contractive part further consists of several blocks, each block containing a convolutional layer followed by a max-pooling layer. Hence, the contractive part downsamples the image (reduces the height and width of the image). On the other hand, the number of channels increases.

2. Expansive part is the second part of the network. Each block of the expansive part consists of upsampling followed by a convolution that reduces the number of channels. The upsampling is done either by the transpose convolution or the interpolation. Also, the expansive part concatenates each block's output with the corresponding feature map with the feature map of the same size from the contracting part of the UNet. This concatenating is called the skip connection and enables precise localization.

Figure 6.1 (Ronneberger et al - [41]) illustrates the architecture in detail.

Since the UNet proves to be successful on various segmentation tasks (such as bladder cell segmentation in phase-contrast microscopy [24]), we will apply the UNet to our problem. As a skeleton implementation, we use a following git repository [1]. However, the UNet architecture slightly differs from the model proposed by Ronneberger in [41]. Namely, the contractive block is expanded by the Batchnorm and ReLu layer.

Moreover, we needed to perform several alternations to adapt the implementation for our task. Firstly, our images are too large and thus too memory demanding to pass through the network at once. Hence, we needed to implement our own

[1] https://github.com/milesial/Pytorch-UNet/tree/master/unet

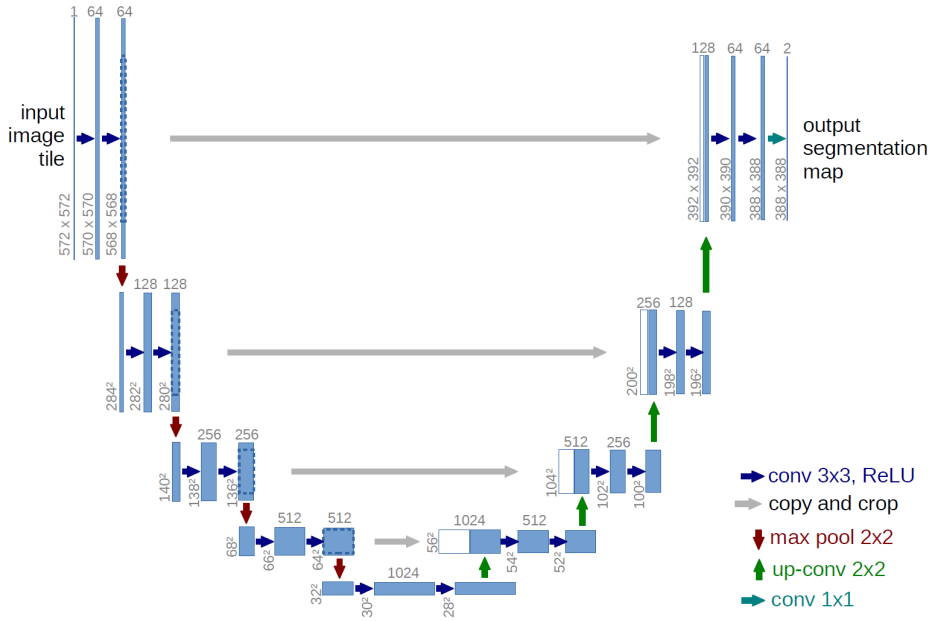


FIGURE 6.1: *Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower-left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.*
The description and the figure provided by Ronneberger et al. in [41].

dataset loader that splits the images into partially overlapping tiles. We then segment the tiles by the UNet and compose them into a full-sized image afterward.

Also, we enabled GPU processing (using PyTorch [40]) that enhances the training and testing speed tremendously. Another important alternation is a loss selection. We will discuss the losses in more detail in Section 6.2 .

Moreover, we added the possibility to continue the training after the unexpected training termination, as well as a possibility to test the network after an arbitrary epoch. As a result, we do not need to rerun the whole training process if the training stops prematurely.

We used the PyTorch library for the implementation ([40]). In the following sections, we will present the different loss functions used.

6.2 Loss Selection

For the classifier to learn, it is rather important to choose the proper loss function. So far, we have tried two approaches.

6.2.1 Pixel-wise Classifying

This section will describe the classifier that assigns each pixel the probability of the pixel being the foreground (nuclei). This classifier tries to predict the same output as the reference mask. That means the value 1 for pixel containing nuclei and 0 for the background pixel.

To achieve this, we used binary cross-entropy as the loss function. Additionally, we improved the performance of the neural network by weighting the components of cross-entropy loss by the occurrence ratio of the foreground and background. This change significantly enhanced the resulting classification. The resulting loss is, therefore, the mean of weighted binary cross-entropy losses of each pixel.

Let us have the input image I, reference segmentation $p_{F(x,y)}$, the neural network prediction, and foreground weighting factor as follows:

- I:
  - I represents the input image, $I_{(x,y)}$ is a pixel value on the position (x, y)
  - $|I|$ represents the number of pixels
- $\hat{p}_{F(x,y)}$:
  - $\hat{p}_{F(x,y)}$ is the predicted probability that the pixel (x, y) is foreground.
- $\hat{p}_{B(x,y)}$:
  - Analogically to the $\hat{p}_{F(x,y)}$, $\hat{p}_{B(x,y)}$ stands for the probability, that the cell is background
  - $\hat{p}_{B(x,y)} = 1 - \hat{p}_{F(x,y)}$
- $p_{F(x,y)}$:
  - This is the reference segmentation of the pixel with coordinates (x, y).
  - $p_{F(x,y)} = 1$ for foreground and $p_{F(x,y)} = 0$ for background
- w_f:
  - w_f represents the foreground weighting factor.
  - It changes the influence of the false-positive mistakes and the false-negative mistakes.
  - We decided to set it to an empirically observed background pixels versus foreground pixels ratio on the training set.

  - To approximate it, we randomly sample M reference pictures from the training set and for each compute the number of foreground pixels $F_i = |\{(x,y) : I(x,y) = 1\}|$ and the number of background pixels $B_i = |\{(x,y) : I(x,y) = 0\}|$.
  - $w_f = \frac{\sum_{i=0}^{M} B_i}{\sum_{i=0}^{M} F_i}$

- $w_{(x,y)}$:
  - weight for the pixel on the position (x,y)
  - $$w_{(x,y)} = \left\{ \begin{array}{ll} w_f, & \text{if } p_{F(x,y)} = 1 \\ 1, & \text{if } p_{F(x,y)} = 0 \end{array} \right\}$$

Consequently, we define the loss as follows:

- $l_{(x,y)}$:
  - loss for the pixel with coordinates (x,y)
  - $l_{(x,y)} = -w_{(x,y)}[\cdot p_{F(x,y)} \cdot \log \hat{p}_{F(x,y)} + (1 - p_{F(x,y)}) \cdot \log (1 - \hat{p}_{F(x,y)})]$
- $L(I)$:
  - loss of the whole image I
  - $L(I) = \frac{1}{|I|} \cdot \sum_{(x,y) \in I} l_{(x,y)}$

To increase the numerical stability, we implement the loss layer already combined with the previous sigmoid layer. This does not change the loss defined above. It only increases the numerical stability due to the log-sum-exp trick (for more details, see [3]). The combined layer is already available in Pytorch.

6.2.2 Distance Transform Segmentation

The main disadvantage of the pixel-wise approach is the following:
Suppose that classifier marks the pixel A as the foreground. However, the reference says the pixel A is the background. The classifier is, therefore, wrong. But the feedback is not really helpful. We do not get the information on how far from the boundary we are. This is especially important for our type of data where the exact location of the boundary is not well defined.

Contrary to the pixel-wise approach, the distance transform offers us more information about the mistake (how far the prediction is). Instead of the binary classification presented in the previous section, we formulate the problem as a regression (inspired by Naylor [38]).

The difference from the previous approach is the following:

1. We need to transform the reference. Each pixel will now hold the distance to the closest background pixel in the original reference. We achieve this transformation using python scipy [26] library.

2. We also need to change the loss function. We follow the literature ([38]) and choose the mean square error (MSE) loss as we try to predict the continuous variable (distance).

Having the same input image I and the reference segmentation $p_{F(x,y)}$ as in the pixel-wise approach, we introduce the input transformation and the loss as follows:

- $v_{(x,y)}$
  - value $v_{(x,y)}$ expresses the euclidian distance from pixel with coordinates (x, y) to the closest background pixel in the reference (ground truth)
  - $v_{(x,y)} = \min_{(x',y'),p_{F(x',y')}=0}\{d((x, y), (x', y'))\}$, d is euclidean distance
- $v_{\hat{(x,y)}}$
  - the estimated distance map
  - the neural network tries to predict for each (x, y) the value $v_{(x,y)}$.
- L
  - mean square error loss
  - $L = \frac{1}{|I|} \cdot \sum_{(x,y)\in I}(\hat{v}_{(x,y)} - v_{(x,y)})^2$

6.3 Quantitative Evaluation of Neural Network Performance

This section will analyze and compare the results from two neural networks described in Section 6.2 with different loss criteria.

First, we will describe the methods used to compare the reference image and the neural network's output. In the case of the second (distance-transform) approach, it is necessary to threshold the ouput in order to obtain the binary segmentation. Thresholding enables us to compare the results from the distance-transform-based approach with the reference and the pixel-wise approach regarding the sensitivity, precision, and Dice loss. We will discuss the details of thresholding in Section 6.3.2.

Initially, we preprocess the output of the neural network. This preprocessing includes the closing operation to eliminate artifacts and also the mentioned thresholding in the case of output from NN with the distance-transform loss criteria.

We will then proceed to the nuclei detection, namely clustering the pixels marked as nuclei into separate objects. Therefore, each connected component of pixels with value one is considered to be a nucleus. Two pixels are connected if they are adjacent or diagonally adjacent.

Consequently, we eliminate the noise. That means eliminating small areas that are too little to represent nuclei. These are determined as the regions smaller than one-tenth of the average size of nuclei in the particular image.

6.3.1 Comparison of the Reference Image versus the NN Classification

It is essential to compare the neural network output with the reference image to determine the performance of the neural network. In this section, we will therefore focus on the method that provides this comparison. We will also introduce the objective criteria which enable us to evaluate the performance of the neural network quantitatively.

The comparison of images is based on matching the nuclei in the reference with nuclei in our classification. In order to approximate the number of matched objects, it is sufficient to compare the centers of masses of the individual cells.

Let us have a connected component I : $\{(x_i, y_i), i \in \{1, ..., k\}\}$ (where k represents the size of the component). We compute the center of mass C_I of the component.

Let us also denote all components found in our segmentation $I_1, \ldots, I_n$, where n stands for the number of components in the output of the neural network. Analogously, we denote components found in the reference image $R_1, \ldots, R_m$. Likewise, m represents the total number of the detected components.

Next, we introduce the mapping f from $i \in \{1, \ldots, n\}$ to $j \in \{1, \ldots, m\}$, such that

$$f(i) = \operatorname{argmin}_j \|C_{I_i}, C_{R_j}\|. \tag{6.1}$$

Presented mapping finds each center of component I the closest component R, taking the euclidean distance. Similarly, we introduce the mapping g from $j \in \{1, \ldots, m\}$

to $i \in \{1, \ldots, n\}$, such that

$$g(j) = \underset{i}{\operatorname{argmin}} \|C_{I_i}, C_{R_j}\|. \tag{6.2}$$

Lastly, we propose the tolerance bound ϵ, representing the maximal distance acceptable to match the two nuclei.

At this point, we can proceed to count the number of detected true positive objects (matched centers), false-positive objects, and false-negative objects.

Consequently, we define the true positives as follows:

$$TP = |(i,j) \in \{1, \ldots, n\} \text{x} \{1, \ldots, m\} : f(i) = j, g(j) = i, \|C_{I_i}, C_{R_j}\| \leq \epsilon|. \tag{6.3}$$

That is, we accept the correspondence of the component I in the NN output with the component R in the reference image, if the center of R is the closest one to the center of I and vice versa. Moreover, the centers lie within the distance bound ϵ.

Then we determine the number of false-positives components in the NN output image, such that there is no corresponding component in the reference image satisfying the conditions presented above. That means the nuclei R in the reference image closest to the nuclei I in the NN image is either out of the ϵ bound, or there exists other nuclei J in the NN output image closer to the nuclei R.

Clearly, these are all of the components in the neural network output that are not true positives. Therefore, it naturally follows that

$$FP = n - TP. \tag{6.4}$$

Analogously, we define the false-negatives as the number of components in the reference image that are not true positives. That is, the number of components that are in the reference image but absent in the neural network output. Formally,

$$FN = m - TP. \tag{6.5}$$

A min-cost matching algorithm (e.g., Hungarian algorithm [29]) can further improve the matching in the future. On the other hand, our method is sufficient and computationally less expensive.

In order to determine the ϵ reasonably, we suggest the following: We measure the distances between the closest neighbors through the training dataset. After that, we let q be 99% quantile, that is, the distance such that 99% of all closest distances are within the upper bound q. Finally, we set the ϵ bound as half of the distance q:

$$\epsilon = \frac{1}{2}q$$

.

At this point, we are able to evaluate the results obtained from NN quantitatively. Namely, we show sensitivity, precision, and Dice loss (which is independent of the object level measures above).
Sensitivity equals $\frac{TP}{m}$, where m represents the number of components in the reference image.
Precision equals $\frac{TP}{n}$, where n represents the number of components in our classification.

We compute the Dice loss as

$$1 - \frac{2 \cdot \text{intersection}}{\text{union}}. \tag{6.6}$$

That means for two binary images (in our case, the reference and the binarized output of NN), the intersection is the number of pixels with value one in both images, while the union is the total number of pixels of value one summed through both images.

Formally,

$$\text{Dice}(I_{NN}, I_{Ref}) = 1 - \frac{\sum_{(i,j)} I_{NN}(i,j) \cdot I_{Ref}(i,j)}{\sum_{(i,j)} I_{NN}(i,j) + I_{Ref}(i,j)}. \tag{6.7}$$

Here I_{NN} stands for the binarized output from the neural network, and I_{Ref} represents the reference image.

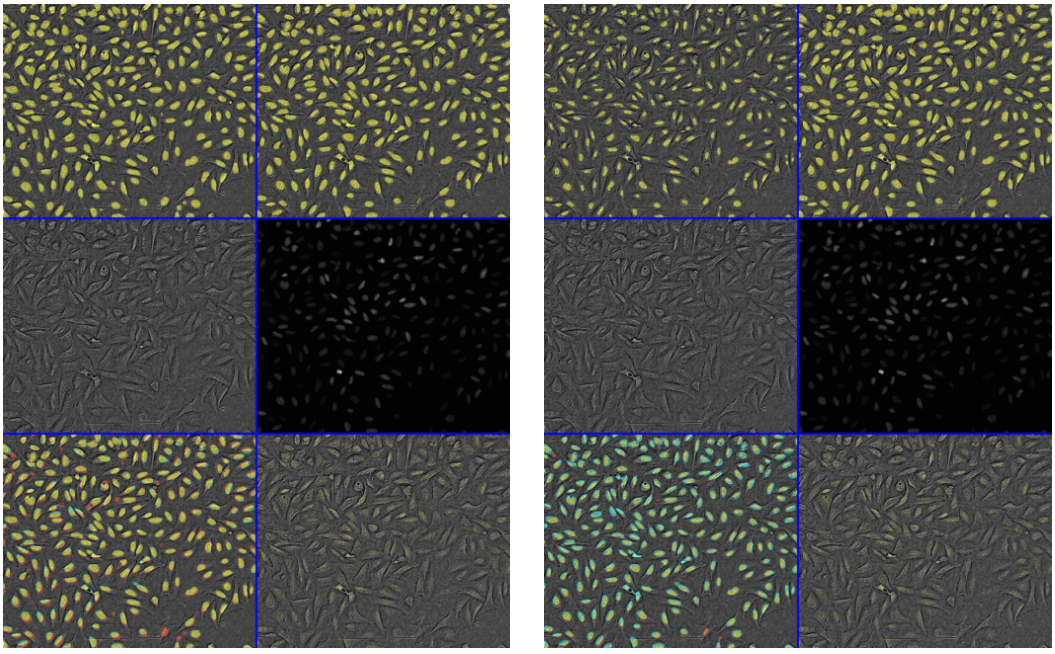
6.3.2 Determination of the Binarization Threshold

As the neural network's output with the distance-transform loss is the distance, it is necessary to apply binarization (by thresholding). As a result, we obtain the desired segmentation.

Theoretically, the threshold value should be zero, as the distance function from the background is zero. However, experimentally the threshold value zero does not perform well. Specifically, it marks a significant area from the background as the foreground. We aim to select the best threshold so that the segmentation aligns well with the reference image. Thus, we need the criteria to compare the thresholds and choose the best one.

In order to achieve the best possible alignment, we established the criteria as the Dice loss introduced in the previous section (Section 6.3.1). As the threshold determination is part of the classifier learning, we are bound to use the training set. Therefore, we evaluate the training set by the pre-trained neural network. We then use the obtained output to find the optimal threshold as described below.

We evaluate multiple thresholds and we try to find the minimal Dice loss.



(A) the optimal threshold value (1.6) (B) the threshold value set to 6.5

FIGURE 6.2: The figure shows the example of cells unaligned due to the threshold. Both subfigures 6.2a and 6.2b contain six images originating from the same sample. In the first row, there is the output from the neural network overlayed over corresponding contrast microscopy (left) versus the reference segmentation overlayed over the same contrast microscopy (right). The second row represents the original input images, i.e., the contrast microscopy (left) and the fluorescence microscopy (right). In the third row, we can see the results from the neural network combined with the contrast microscopy (left) with distinguished FPs, FNs, and TPs versus the original fluorescence combined with the contrast microscopy (right). The lower left-hand corner image shows the true-positives in yellow, while blue represents the false-negatives and red stands for the false-positives. For more examples of the UNet output segmentations, see Appendix B.

Formally, the optimal threshold

$$t_{opt} = \underset{t \in T}{\operatorname{argmin}} \frac{1}{|S_N|} \sum_{(I_N^i, I_R^i) \in S_N \times S_R} Dice(I_N^i(t), I_R^i) \tag{6.8}$$

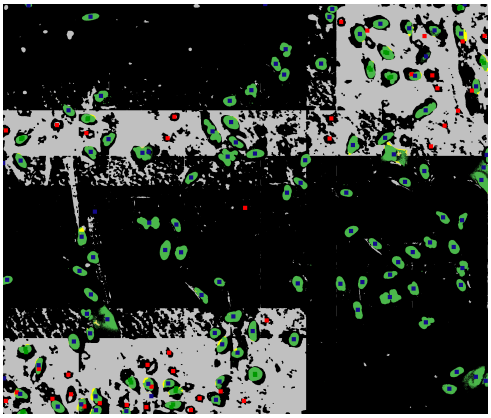
Where:

T: is the set of sampled thresholds.
S_N: represents the subset of the output from neural network on the training set.
S_R: stands for the subset of reference images corresponing to S_N.
$I_N^i(t)$: stands for the thresholded neural network output image.

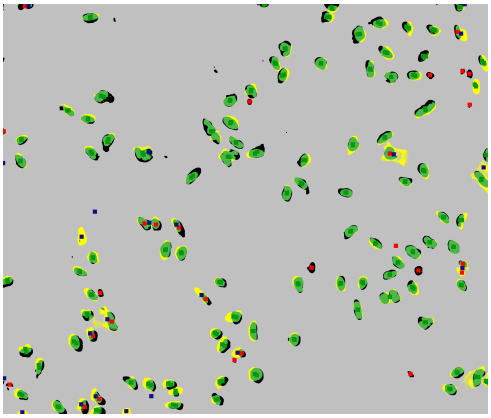
In other words, the best threshold minimizes the mean Dice loss on the training set.

Then we sampled the interval between the thresholds neighboring the optimal one in the same way as before. We present the impact of different thresholds selection in Figure 6.2.

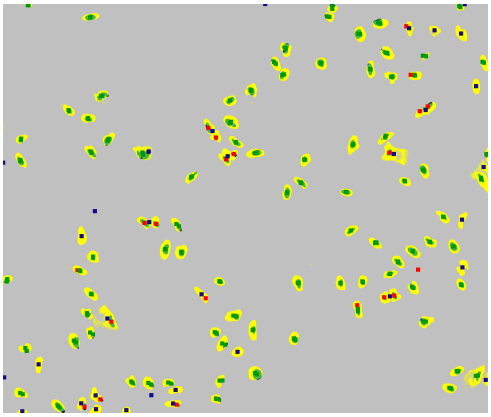
We can see that selection of the optimal threshold value in Subfigure 6.2a almost



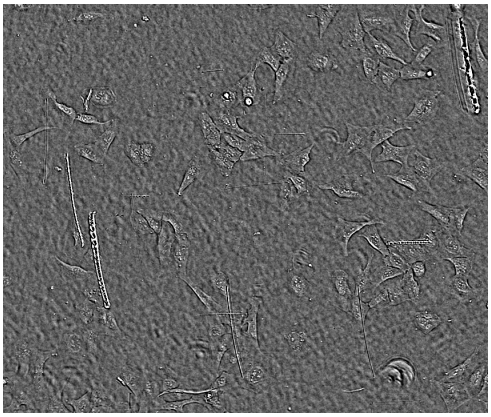
(A) threshold = 0.65



(B) threshold = 1.6



(C) threshold = 6.5



(D) input contrast microscopy

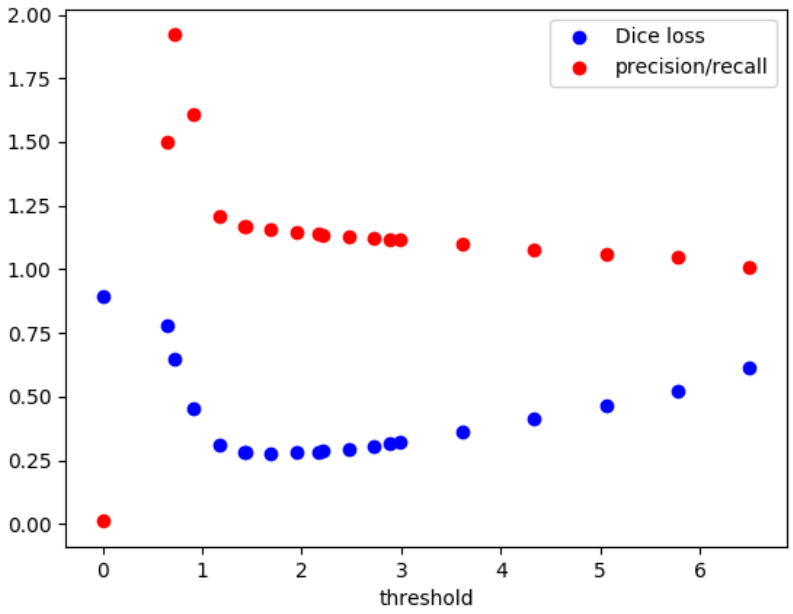
FIGURE 6.3: The figure shows the impact of the underestimating and overestimating of the threshold compared to the optimal one. All of the three subfigures (6.3a - 6.3c) show the distance-transform neural network output overlapped with the reference. The overlapping regions are marked green, false-positives are black with the red center (if the center did not match with reference), and false-negatives are yellow with a blue center (if reference did not match with the neural network output). Subfigure 6.3d shows the input contrast microscopy image.

suppresses all the blue color representing the false-negatives compared to Subfigure 6.2b where the threshold value is 6.5. Figure 6.3 shows the development of the classification by changing the value of the threshold.

We also show the relation between the threshold setting and the mean Dice loss as well as the precision/recall (sensitivity/precision) curve in Figure 6.4.

threshold	median sensitivity	median precision	median Dice loss	mean sensitivity	mean precision	mean Dice loss
6.5	0.744	0.832	0.669	0.657	0.763	0.669
1.6	0.88	0.8	0.249	0.832	0.719	0.305

TABLE 6.1: Impact of the threshold on the neural network based on distance-transform loss



(A)

FIGURE 6.4: Figure illustrates the impact of the threshold change on the Dice loss. Figure shows the thresholds in the range $0 - 6.5$. As the interval $0.65 - 3$ was the most promising, we sampled it with a higher density.

Moreover, this improvement caused that the neural network with the distance-transform loss beats the neural network with the pixel-wise loss in all categories presented in Section 6.3.1. Thus, we conclude that setting the threshold according to the criteria we proposed remarkably enhanced the distance-transform-based neural network's performance.

6.3.3 Experimental Results Comparison of Different Loss Settings

In this section, we will analyze the results from the neural network. We will also compare the neural networks according to the loss function, focusing on the criteria presented in the previous section.

In Table 6.2 we present the neural network results with the pixel-wise loss function compared with the results from the neural network using the distance transform loss function.

For each criterion, we show the mean and median on the testing dataset (we obtain the testing dataset as 30% of all images, these 30% images are not used for training). We can see that the neural network with distance-criteria provides much better results considering the sensitivity and precision. Namely, average precision is higher by $16,5\%$ while median precision is higher by $14,2\%$. Moreover, the sensitivity mean and median are greater by $4,7\%$ and 10.3% respectively.

Another complication might be merging touching nuclei in reference or neural network output while distinguishing individual nuclei in the other image. This results in multiple mistakes, namely removing true-positive detection and adding either false-positive or false-negative detection. Fortunately, this can happen only in really dense nuclei clusters that are not so common, as we will see in Section 6.3.4.1.

method	median sensitivity	median precision	median Dice loss	mean sensitivity	mean precision	mean Dice loss
dst-transf NN	0.88	0.8	0.249	0.832	0.719	0.305
pixelwise-NN	0.641	0.69	0.362	0.61	0.598	0.426

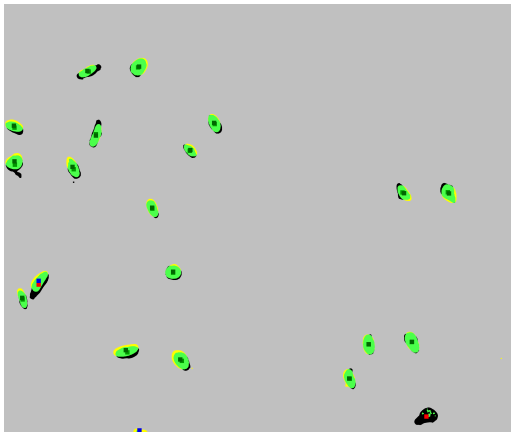
TABLE 6.2: Comparison of results using the pixelwise and distance-transform loss based Unets

Finally, we present the visualization of the nuclei centers found in both reference and NN-classification images in Figure 6.5.

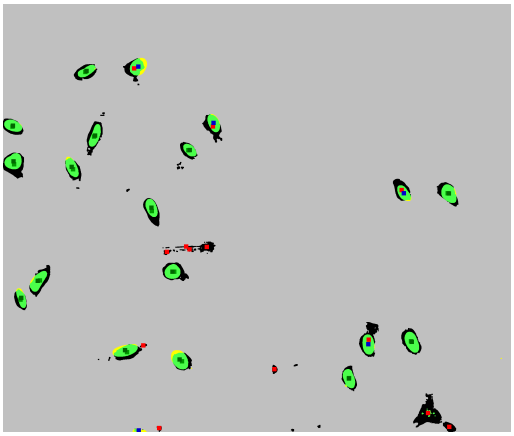
We can see that the nuclei mostly correspond. However, in some of the images, there are many false positives, as shown in Figure 6.6.

(A) contrast microscopy image

(B) fluorescence microscopy segmentation

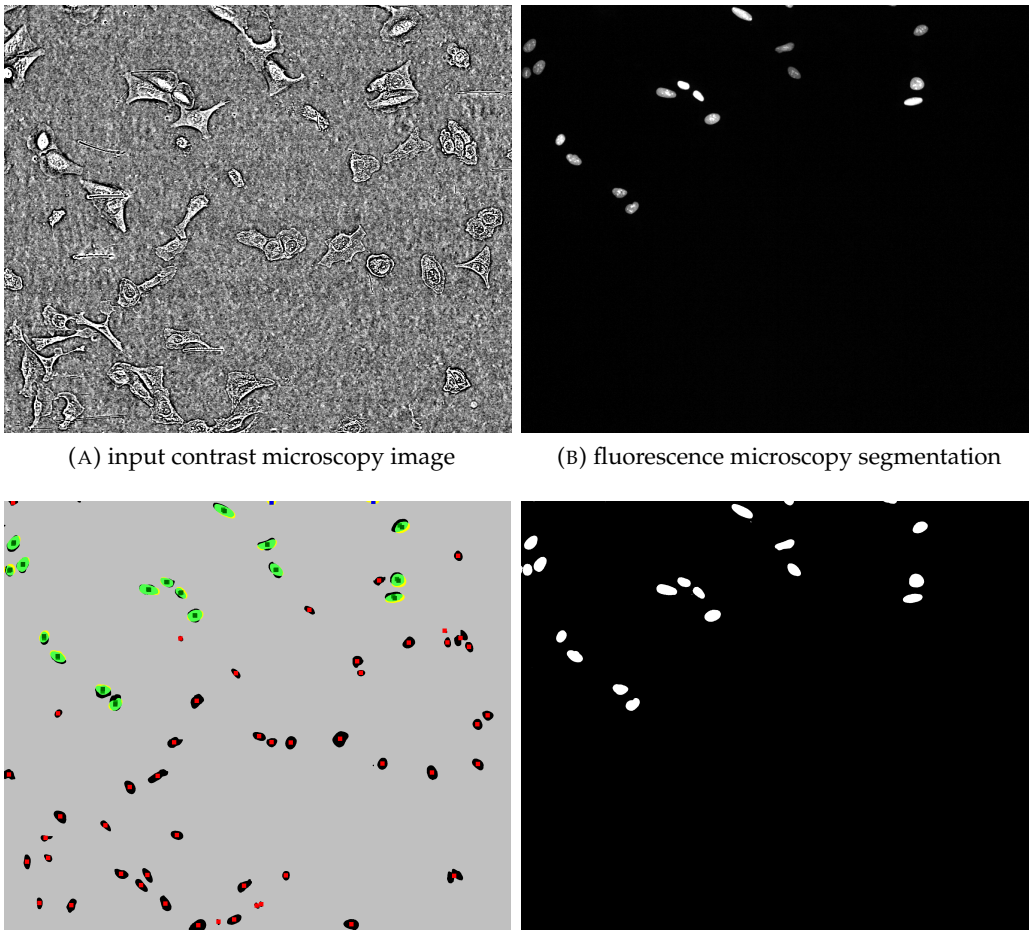


(C) NN output (distance-transform loss)



(D) NN output (pixel-wise loss)

FIGURE 6.5: The figure shows the comparison of centers of the nuclei among the neural network output and reference. Subfigure 6.5c and Subfigure 6.5d show the output from the neural network with the distance-transform loss and with the pixel-wise loss, respectively, combined with the corresponding reference. We can see overlapping nuclei (green) and partially overlapping nuclei. The yellow color shows the nuclei area present only in the reference image (FN), while black shows the areas present only in the neural network output (FP). Moreover, the matched centers are marked green, and the rest is either red (FP) or blue (FN).
Subfigure 6.5a shows the corresponding contrast microscopy input, while 6.5b shows the segmentation obtained from the fluorescence microscopy.



(A) input contrast microscopy image (B) fluorescence microscopy segmentation

(C) NN output (distance-transform loss) (D) segmenation obtained from fluorescence

FIGURE 6.6: The figure shows the neural network output combined with the reference, similarly to Figure 6.5. The colors represent the same features ad in Figure 6.5. We observe many red dots representing the nuclei centers in neural network output that did not match with any centers in the reference. However, we can see the detected nuclei in the contrast microscopy image.

This might be caused by the insufficient coloring in the reference pictures. We will discuss this matter in Section 6.3.4.

On the other hand, the distance-transform-based U-Net handles the artifacts present in microscopy very well. Figure 6.7 illustrates the difference between the distance-transform-based U-Net output and pixel-wise-based U-Net output when input contains the artifacts.

To sum up, the neural network with the distance loss function performs significantly better than the one with the pixel-wise loss function in both sensitivity and precision.

6.3.4 False Positives Detection Problem

In Section 6.3.1 we presented the quantitative statistics associated with the neural network performance. We carefully examined the datasets to identify the most frequent mistakes. As a result, we encountered a few images that detected nuclei not present in the corresponding fluorescence microscopy, although we can clearly distinguish nuclei on contrast microscopy.

This phenomenon leads us to the hypothesis that there is an insufficient response to the color or the color absence in some areas of the fluorescence images. In other words, the fluorescence microscopy from which we obtain reference might not be 100% true. Thus, the ground truth to which we compare the results is not always correct.

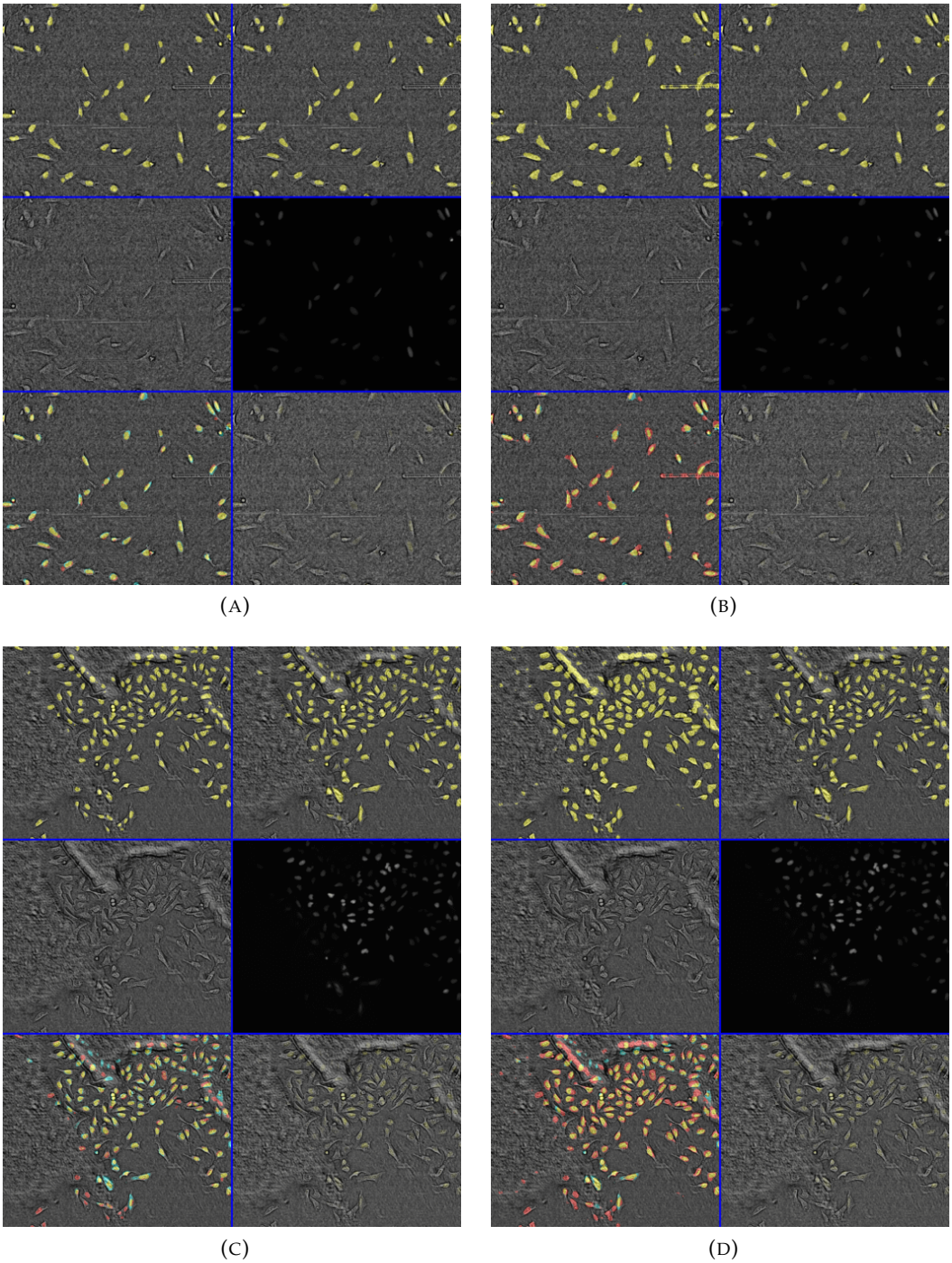
Figure 6.8 illustrates the proposed problem.

On the contrast microscopy in the second row, we see the object that absent in the corresponding fluorescence microscopy in the second row. On the third row, these false-positive nuclei are marked red, while the true positives are marked yellow.

After consulting with the biology area experts, they confirmed the fluorescence microscopy might miss some of the nuclei.

In order to identify the quantity of such possible misleading reference images, we constructed the histogram presented in Figure 6.9.

The histogram shows the number of images with the given range of false-positives detections. Satisfactory, the histogram values decrease rapidly with the increasing number of false-positives detection. It means that many images are not causing problems related to the false positives. However, there are a few images with a rather high amount of false-positives. The highest number of false-positives is between 262 and 271 nuclei in one image, which might have significantly influenced the statistics presented above. Hence, we assume that there might also be another problem. Therefore, we further examine the datasets. In the following section (Section 6.3.5), we consider specific datasets' impact on the neural network performance. We will also inspect the most suspicious images with the highest number of false-positives detections.



(A) (B)

(C) (D)

FIGURE 6.7: Figure shows four subfigures. Subfigures 6.7a and 6.7c show the output from the distance-transform-based U-Net, while Subfigures 6.7b and 6.7d show the output from the pixel-wise-based U-Net. We can see that in Subfigure 6.7a the artifact is supressed. Both subfigures contain six images: In the first row, there is the output from the neural network overlayed over corresponding contrast microscopy (left) versus the reference segmentation overlayed over the same contrast microscopy (right). The second row represents the original input images, i.e., the contrast microscopy (left) and the fluorescence microscopy (right). In the third row, we can see the results from the neural network combined with the contrast microscopy (left) with distinguished FPs, FNs, and TPs versus the original fluorescence combined with the contrast microscopy (right).

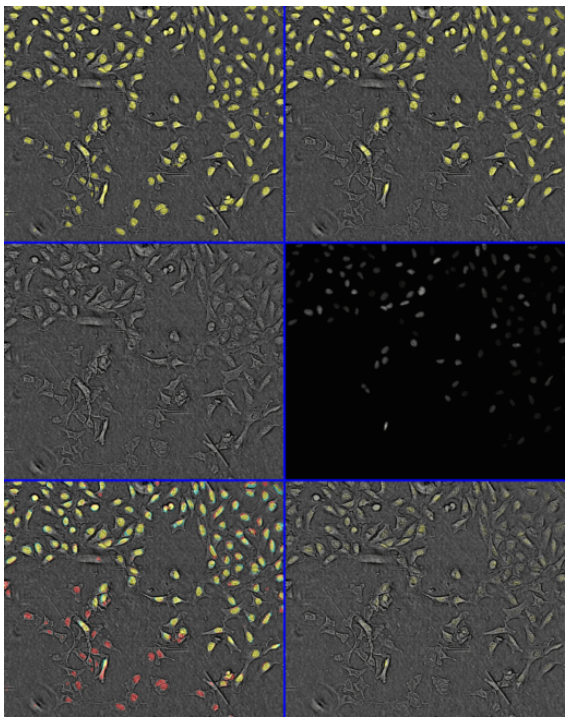


FIGURE 6.8: The figure shows the false positive detections. The figure layout is the same as described in Figure 6.7. In the bottom left-hand corner image, we observe many false-positive nuclei. However, in the input image (second row, left), we can see the nuclei-like shapes. On the other hand, there is not a trace of such nuclei in the reference image.

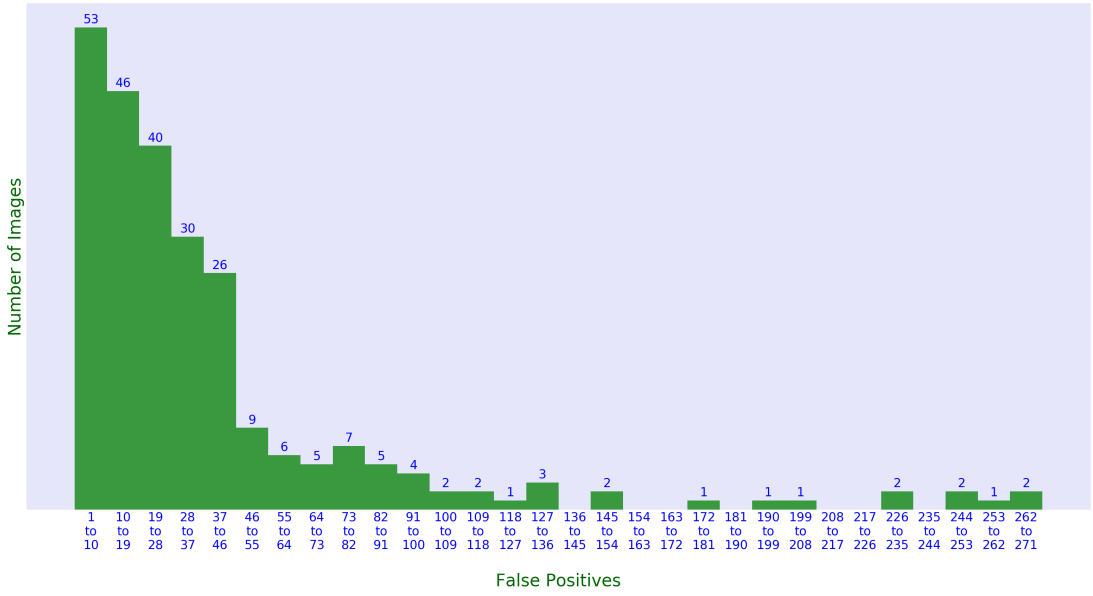


FIGURE 6.9: The number of the images with the given range of the false-positives per image

6.3.4.1 False Positives Detection Related to the Touching Cells

As we discussed in Section 6.3.3 and Section 6.3.4, the neural network detects many unexpected nuclei. One reason might be the insufficient coloring. In order to reveal other problems, we sorted the test-set outputs by the number of false-positives detections. Consequently, we discovered two other possible explanations.

The first observation is that the previously introduced method for alignment is not well suited for images with a high nuclei amount. Obviously, if the two cells touch each other in the reference but not in the neural network's output, the result is a significant shift of the estimated center. Hence, we are totally incapable of aligning the nuclei properly. Moreover, one nucleus is lost in that way as it merges with the touching one. The merging in just one of the images naturally leads into both false-positive and false-negative detection.

As we have seen in the histogram (Figure 6.9), the images with many touching cells are not so regular. However, such images seriously bias the mean value of both sensitivity and precision as they add hundreds of mistakes.

Figure 6.10 shows the image with many nuclei detections. We can see that almost none of the nuclei align correctly.

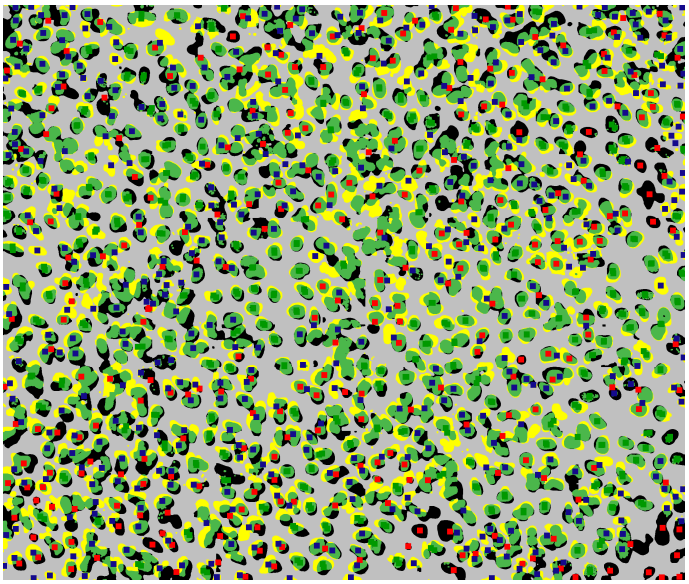
Nonetheless, the ruined accuracy in our estimation does not imply the wrong classification. We notice that the neural network does not have a problem detecting the nuclei. We also observe that the classification is similar. The main problem is the instability of the proposed evaluation on the images with many touching cells. In other words, a small difference in the classification has a high impact on the resulting accuracy. Even one misclassified pixel might cause the neighboring cells to merge and thus totally misplace the center. On the other hand, the Dice criterium does not suffer due to the images with a high nuclei density.

Figure 6.11 illustrates the actual differences between the images. We can see the poor quality of both contrast microscopy and fluorescence microscopy in the second row. However, we observe large numbers of yellow (true positive) nuclei in the bottom left-hand corner image that align with the nuclei in the upper right-hand corner image.

To sum up, the impact of touching cells on sensitivity and precision seems to be rather higher than the nuclei absence in some of the reference images.

6.3.5 Dataset Impact on the NN Performance

We discovered another interesting observation about the neural network performance while sorting the images by the number of false-positives occurrences. Namely, the images sorted by the number of false-positives are also almost sorted by the dataset they originated from. We described individual datasets in Section 2.1. Particularly, we detect the most false-positives in the dataset '72h', following by the dataset 'before', '24h', and 'only cells'. However, the last three are less strictly separated.



(A) the distance-transform neural network output

FIGURE 6.10: The figure shows the case where many nuclei touch. The green color represents the overlapping nuclei areas, while yellow and black stand for false-negatives and false-positives, respectively. Although we can see a high amount of green, we see that the centers did not align correctly. The centers marked green are aligned, while red centers represent the false-positive centers, and blue represents the false-negative centers.

Therefore, we made a quantitative statistics evaluation on each of the datasets separately.

Table 6.3 presents the obtained results.

dataset	median sensitivity	median precision	mean ± SD sensitivity	mean ± SD precision	median Dice loss
24h	0.904	0.839	0.897 ± 0.054	0.764 ± 0.196	0.215
72h	0.724	0.571	0.648 ± 0.225	0.523 ± 0.237	0.314
before	0.889	0.789	0.86 ± 0.105	0.703 ± 0.209	0.278
only	0.847	0.815	0.841 ± 0.044	0.81 ± 0.045	0.248

TABLE 6.3: The table shows the performance of distance-transform-based NN on individual datasets. The training was performed on the mixed dataset.

We can see quantitative statistics of each dataset on a separate row. Table 6.3 shows both median and mean value to criteria introduced in 6.3.3. We also observe that the results from the dataset '24h' are better than all of the other datasets in all of the criteria except mean precision, where the dataset 'only' is the best. However, the differences between the datasets 'only' and 'before' and '24h' are not so significant.

On the other hand, on the dataset '72h', the classification does not achieve high precision nor sensitivity. Nonetheless, the worse classification results on the dataset '72h' is not so surprising. As we already discussed in the previous section, many touching cells are problematic regarding sensitivity and precision. Since the dataset

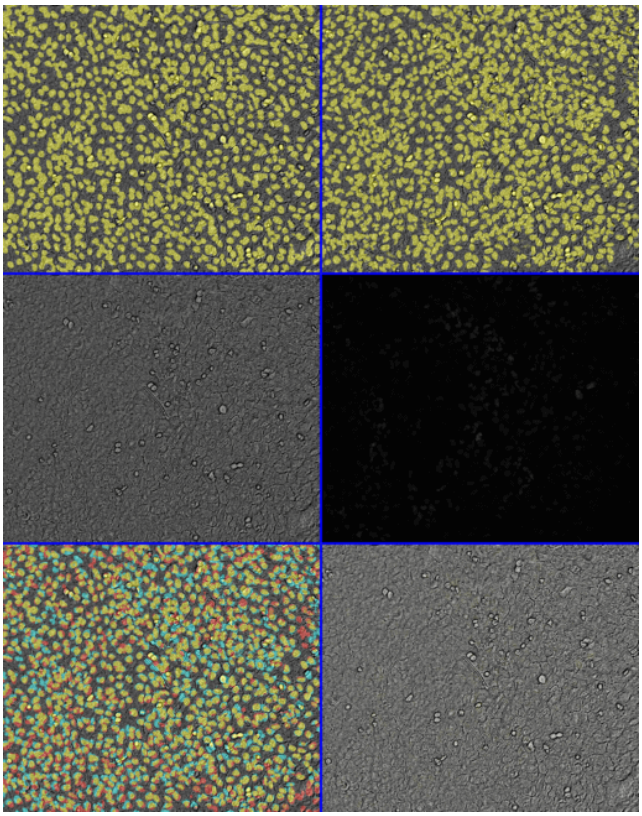


FIGURE 6.11: The figure shows the aligning of many touching cells. The figure layout is the same as described in Figure 6.2. We observe many touching cells in the output segmentation that cause problems with the nuclei matching. We also observe a poor reference image, where the nuclei are almost invisible.

'72h' contains the most cases of such merging (see Figure 6.10), it is natural that the quantitative evaluation achieves worse results than the other datasets. Also, the nuclei in the dataset '72h' seem to be less distinctively recognizable (as we presented in Figure 6.11). Admittedly, both the median and mean Dice loss of the dataset '72h' is the highest amongst the datasets. However, the difference between median Dice loss of the dataset '72h' and the second worse median Dice loss (the dataset 'before') is not so significant (3.6%).

6.3.6 Further Analysis of the '72h' dataset

While comparing the mean and median of the criteria in one dataset, we notice that they do not differ significantly, except for the precision of the datasets '24h' and 'before' and all of the criteria in the dataset '72h'. That leads us to the assumption that there exist poorly classified cases biasing the overall results. In order to inspect this assumption, we constructed histograms in Figure 6.12 separately for each dataset.

Accordingly, we see the difference between the dataset '72h' and the others. Namely, the maximal number of false-positives detected in '72h' dataset fairly exceeds the numbers of false-positives in the other datasets. Moreover, in the datasets

'24h', 'before', and 'only', the number of images decreases with the increasing number of false-positives. In comparison, in the dataset '72h' we observe more images on the right side of the false-positives spectrum.

Thus, we conclude that the impact of the dataset from which the image originated is rather important. Specifically, the dataset '72h' is problematic as it significantly biases the overall statistics presented in Table 6.1. Also, the dataset '24h' yields the best results amongst the datasets, according to our criteria.

6.3.7 Possible Improvement of the Classification

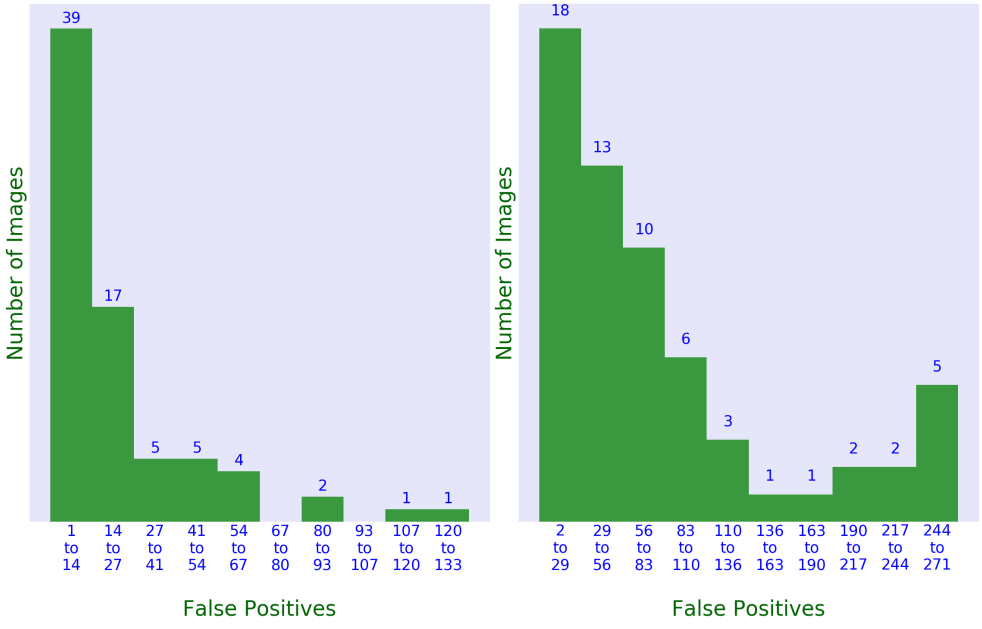
As we have discussed in the previous sections, some of the images contain false-positive detections. The reasons for these detections can be either insufficient reference coloring or merging the cells. However, merging the cells does not affect the Dice loss criteria. Another reason might be the worse quality of some of the images.

Therefore, we inspected a change of the precision and sensitivity on the training set when we omitted a part of the training images. In Figure 6.13 we demonstrate this change when omitting 5% to 25% images containing the most false-positives. Naturally, both sensitivity and precision increase. However, the improvement is quite smooth, and there is no sudden growth.

We also present Figure 6.14a and Figure 6.14b that show a distribution of the omitted images by the dataset they originated from. Figure 6.14a shows the distribution when the omitting is based on the number of false-positives, while Figure 6.14b is based on omitting the images with the worse Dice loss.

Both figures (6.14a, 6.14b) confirm our assumption that the dataset '72h' reaches worse results than the others in both false-positives and Dice loss. We can also see that a minimal number of the worse classified images originate from the dataset 'only'.

The results presented in this section lead us to an assumption that it could be beneficial to omit some of the images during the training of the neural network. In order to further support this idea, we present two images (6.15a,6.15b) in 20% quantile of images sorted by the number of false-positives. We can clearly see the nuclei in the contrast microscopy, while in the reference, we do not observe all of them. The nuclei missing in the reference are marked red.



(A) The dataset '24h' (B) The dataset '72h'

(C) The dataset 'before' (D) The dataset 'only'

FIGURE 6.12: The figure shows the histograms of false-positive detections on each dataset. Each subfigure shows the number of the images containing the given range of the false-positives detections.

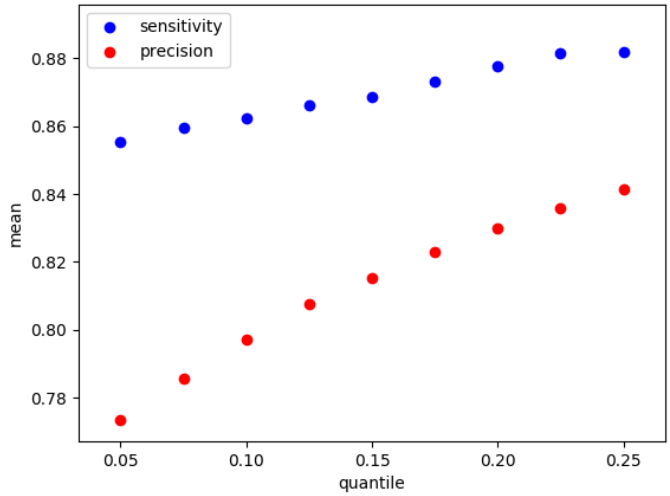
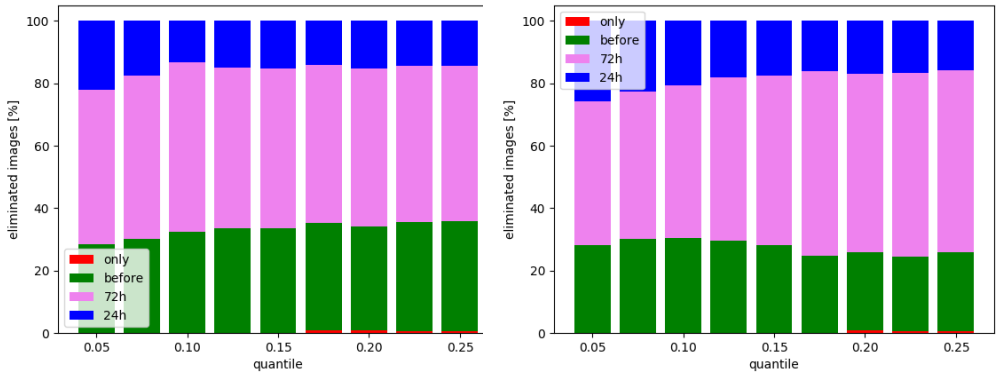
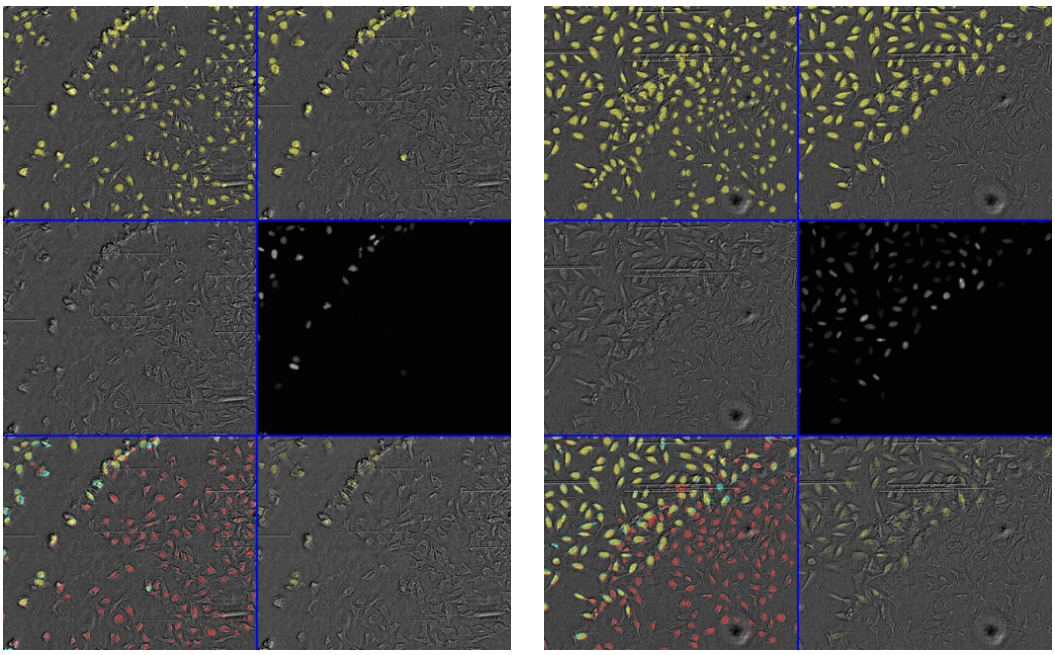


FIGURE 6.13: The figure shows the evaluation of the sensitivity and precision while not taking into account the worse images regarding the false-positives detections.



(A) (B)

FIGURE 6.14: Both subfigures show the distribution of the worse images over the datasets for various quantiles. In Subfigure 6.14a the images are sorted by decreasing Dice loss. That means the percentual number of images with the higher losses is shown in the figure. In Subfigure 6.14b the images are sorted by decreasing false-positives number. That means the figure shows the occurrences of images with the higher FP numbers.



(A) (B)

FIGURE 6.15: Figure shows the examples of images in 20% quantile of training images descendingly sorted by the false-negatives occurrences. Both subfigures 6.15a and 6.15b contain six images. In the first row, there is the output from the neural network overlayed over corresponding contrast microscopy (left) versus the reference segmentation overlayed over the same contrast microscopy (right). The second row represents the original input images, i.e., the contrast microscopy (left) and the fluorescence microscopy (right). In the third row, we can see the results from the neural network combined with the contrast microscopy (left) with distinguished FPs, FNs, and TPs versus the original fluorescence combined with the contrast microscopy (right). The lower left-hand corner image shows the true-positives in yellow, while blue represents the false-negatives and red stands for the false-positives. We can see many false-positives detections in the areas where we see objects in the input contrast microscopy image. The figure illustrates the insufficient reference-coloring problem.

6.3.7.1 Training-set Pruning

In the previous section, we discussed the problems with the classification. We have seen some images that are missing a part of the reference (Figure 6.15), as well as images where the nuclei are harder to recognize (Figure 6.11).

These observations lead us to an assumption that some of the images in the training set could harm the network training. In other words, samples with the wrong reference did not change the network's output into the desired classification.

Therefore, we tried to distinguish the possibly harmful samples from the good ones and remove the bad samples from the training set.

From Figure 6.13 we estimated that omitting the 15% of the worst images from the training set could improve the training reasonably well. In this context, the worst images have the highest false discovery rate after the classification.

Thus, we trained the neural network. After that, we listed the worse 15% images regarding the false positive rate. We trained a new neural network with the training set without the omitted images in the list. Finally, we evaluated the obtained classifier on the test set and saw the results.

Unfortunately, the training set preprocessing did not improve the output classification. Table 6.4 shows that the previous approach without preprocessing outperforms the classifier with the preprocessing on each dataset. The original results without preprocessing are in the rows $1-4$, while the results from the training with the preprocessed dataset are in the rows $5-8$.

method	dataset	median sensitivity	median precision	mean $\pm$ SD sensitivity	mean $\pm$ SD precision
dst-transf NN	24h	0.904	0.839	0.897 $\pm$ 0.054	0.764 $\pm$ 0.196
	72h	0.724	0.571	0.648 $\pm$ 0.225	0.523 $\pm$ 0.237
	before	0.889	0.789	0.86 $\pm$ 0.105	0.703 $\pm$ 0.209
	only	0.847	0.815	0.841 $\pm$ 0.044	0.81 $\pm$ 0.045
dst-transf NN-pre	24h	0.811	0.621	0.74 $\pm$ 0.195	0.554 $\pm$ 0.213
	72h	0.367	0.36	0.392 $\pm$ 0.252	0.306 $\pm$ 0.216
	before	0.815	0.546	0.736 $\pm$ 0.223	0.475 $\pm$ 0.236
	only	0.723	0.554	0.647 $\pm$ 0.18	0.552 $\pm$ 0.073

TABLE 6.4: Dataset comparison with the training set preprocessing

7 Image to Image Translation

In this chapter, we will discuss a new technique, which could improve the classification. Namely, we will apply the pix2pix method introduced by Isola et al.([25]) to learn to translate our phase-contrast images to resemble the segmentation of the fluorescence images. It is a modern approach in deep learning, successfully applied in various tasks (e.g. [32], [7])

The main idea of the pix2pix architecture is the following: The pix2pix model consists of two separate neural networks, that is, a generator and a discriminator. The task of the generator is to generate the image I_G from the input image I_{in}. The task of the discriminator is to distinguish between the images created by the generator and the original reference image. Specifically, the discriminator is given a tuple consisting of the input image I_{in} and an image I_U. The image I_U can be either the original reference or the image I_G produced by the generator.

The training works as follows: At each step of the training epoch, the generator gets the input image I_{in} and produces the output image I_G. After that, the generator and the discriminator evaluate the loss and update the weights via gradient descent to minimize the loss.

The generator loss consists of two different components. The first component $L_{\text{similarity}}$ punishes the generator for the difference of the output image I_G from the reference I_R corresponding to the input image I_{in}. The second component L_{fake} punishes the generator when the discriminator recognizes that the image I_G was artificially generated by the generator.

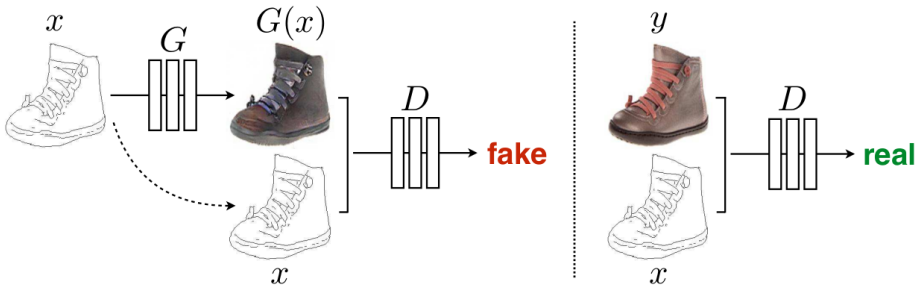


FIGURE 7.1: *Training a conditional GAN to map edges → photo. The discriminator, D, learns to classify between fake (synthesized by the generator) and real {edge, photo} tuples. The generator, G, learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe the input edge map.*
Figure with the description extracted from [25].

The two losses are combined with a ratio λ. As a result of the described losses combination, the generator should create images both similar to the reference and unrecognizable from the reference by the discriminator.

On the other hand, the discriminator gets two tuples at each step of the training epoch: (image I_{in}, image I_R), (image I_{in}, image I_G). Both tuples contain the same input image I_{in}, while image I_R represents the corresponding reference image, and image I_G stands for the image produced by the generator. Importantly, the discriminator does not know which of the tuples contains I_G and which contains I_R.

After that, the discriminator predicts for each input tuple whether the second tuple component is the true reference I_R or the generated image I_G. These predictions are independent of each other, the order of the tuples is arbitrary.

The objective of the described model is formally defined in [25] as follows:

- G: $\{I_{in}, n\} \rightarrow I_G$
  - G stands for the generator mapping from the input image and a random noise vector n to the output image I_G.
- D: $\{I_{in}, I_U\} \rightarrow \{0,1\}$
  - D stands for the discriminator mapping from the tuple of the input image I_{in} and the unknown origin image I_U to 0 (if the prediction is that I_U is created by the generator) or 1 (if the prediction is thath I_U is the original reference image).
- $L_{cGAN}(G, D) = \mathbb{E}_{I_{in},I_R}[logD(i_{in}, I_R)] + \mathbb{E}_{I_{in},n}[log(1 - D(I_{in}, G(I_{in}, n)))]$
  - L_{cGAN} represents the criterion of the discriminator. The discriminator's goal is to maximize it, while the generator tries to minimize it.
- $L_{L1}(G) = \mathbb{E}_{I_{in},I_G,n}[\|I_R - G(I_{in}, n)\|_1]$
  - $L_{L1}(G)$ is the expected value of the difference of the generated image from the reference image, in this case L_1 norm.
- G* = $\text{argmin}_G \max_D L_{cGAN}(G, D) + \lambda L_{L1}(G)$
  - G* stands for the optimal generator. We can see that the final objective consists of the similarity component L_{L1} and the component representing the indistinguishability of the I_G and I_R by the discriminator.

In the following sections, we will discuss the possible application of the pix2pix method to our problem. We will also introduce small alternations to the presented approach (e.g., omitting the noise vector, alternating L_{cGAN} criterion so that the output of the discriminator is not just binary). We will also discuss the architecture of the generator, as well as possible alternations to the L_1 loss component.

7.1 Our Problem Adaption

In the previous section, we briefly described the pix2pix technique. The adaption to our problem is the following: The generators' aim will be to create the reference segmentation from the input contrast microscopy image.

The skeleton of the implementation is located in the public git repository of Erik Linder-Norén [1]. However, multiple alternations were needed to reach satisfactory results.

Namely, we totally changed the generator architecture. The original source (Isola et al. [25]) suggests that the U-Net architecture is suitable for the generator. As the original U-Net (A) from the skeleton implementation (Erik Linder-Norén) did not learn anything (generated only the background values), we replaced the U-Net model with a U-Net (B) used in the previous chapter. The main differences between the two U-Nets are the following:

1. The U-Net (A) contains eight downsample layers, and eight upsample layers, while the U-Net (B) contains only four downsample and four upsample layers.

2. The downsample layer in U-Net (A) contains convolution and dropout, while the downsample layer in U-Net (B) contains convolution and max-pooling.

3. U-Net (A) has a final tanh activation layer.

The result was a huge enhancement in the classification performance.

In regards to the computational performance, we added the possibility of GPU computing to the skeleton implementation by Erik Linder-Norén ([1]) (implemented with PyTorch [40]), thus boosted the speed of classification (roughly 70 times faster). Similar to the previous chapter, we needed to divide the input image into tiles (of size 560x560 pixels that partially overlap: stride equals 400 pixels) because the full-image-at-once classification was too memory demanding. After classification, the original-sized image is reconstructed by combining the tiles that partially overlap.

Also, the original purpose of the skeleton implementation (by Erik Linder-Norén) was to generate the right part of the image having the left part. Therefore, we needed to modify slightly the loss computation of the discriminator.

Moreover, the implementation lightly differs from the model introduced in [25]. Particularly, the discriminator does not have a binary output for the whole image, the discriminator rather classifies each pixel separately, and the loss is a sum of these pixel-wise losses.

In the following section, we present the definitions of losses updated for our purposes.

[1] https://github.com/eriklindernoren/PyTorch-GAN/blob/master/implementations/pix2pix/pix2pix.py

7.2 Formal re-Definitions of Objective for Our Purposes

In section 7, we showed the formal definitions of the model losses as presented in Isola's et al. paper [25]. Although the main idea remains unchanged, in our implementation, we slightly altered some features, such as pixel-wise classification of the discriminator. Also, we did not add noise into the input of the generator. However, the generator itself contains skip-connections.

Therefore, we introduce the altered definitions as we use them in our solution.

- G: $I_{in} \rightarrow I_G$
  - Values of the I_G image are in the range $\langle 0, 1 \rangle$.
  - G stands for the generator mapping from the input image to the output image I_G.
- D: $\{I_{in}, I_U\} \rightarrow I_{pred}$
  - Values of the I_{pred} image are in the range $\langle 0, 1 \rangle$.
  - D stands for the discriminator mapping from the tuple of the input image I_{in} and the unknown origin image I_U to the prediction image I_{pred}. Each pixel of I_{pred} is predicted individually. The prediction pixel value one indicates the discriminator classified it as real image (I_R), while the prediction pixel value zero indicates the discriminator classifies it as fake (I_G).
- $L(D) = 0.5\frac{1}{|I_{in}|}\sum_{i,j}\|D(I_{in}, I_R)_{(i,j)} - 1\|_2 + 0.5\frac{1}{|I_{in}|}\sum_{i,j}\|D(I_{in}, G(I_{in}))_{(i,j)}\|_2$
  - $L(D)$ represents the loss of the discriminator (used by Erik Linder-Norén in skeleton implementation [1]).
  - We can see that the loss increases when the prediction of $G(I_{in})$ contains ones, and when the prediction of I_R contains zeros.
  - Thus, the discriminator needs to minimize $L(D)$.
- $L_{L1}(G) = \|I_R - G(I_{in})\|_1$
  - $L_{L1}(G)$ represents the part of the generator loss. Namely, the part of loss based on the difference from the reference image.
- $L_{GAN}(G, D) = \frac{1}{|I_{in}|}\sum_{i,j}\|D(I_{in}, G(I_{in}))_{(i,j)} - 1\|_2$
  - $L_{GAN}(G)$ represents the L_{fake} component of the loss.
  - If the predictor predicts that the generated image $G(I_{in})$ is fake (zeros), the loss increases.
  - Therefore, the generator needs to minimize the L_{GAN} and L_{L1}.

- G* = $\text{argmin}_G L_{GAN}(G, D) + \lambda L_{L1}(G)$
  - G* stands for the optimal generator given D. We can see that the final objective consists of the similarity component L_{L1} and the component representing the indistinguishability of the I_G and I_R by the discriminator.
- D* = $\text{argmin}_D L(D)$
  - D* stands for the optimal discriminator. That is the discriminator minimizing the $L(D)$ by correctly distinguishing between reference and the image produced by the generator.

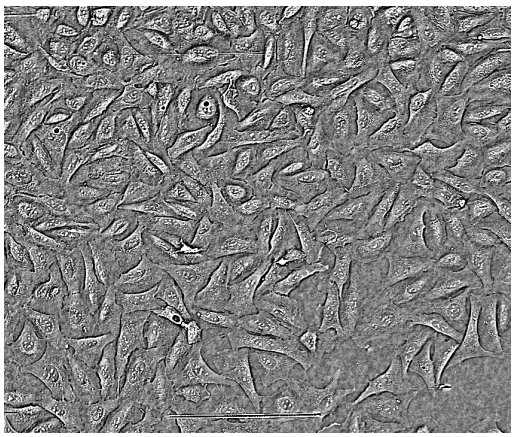
In the following sections, we will analyze the impact of the different model settings on the classification performance.

7.3 Loss Selection

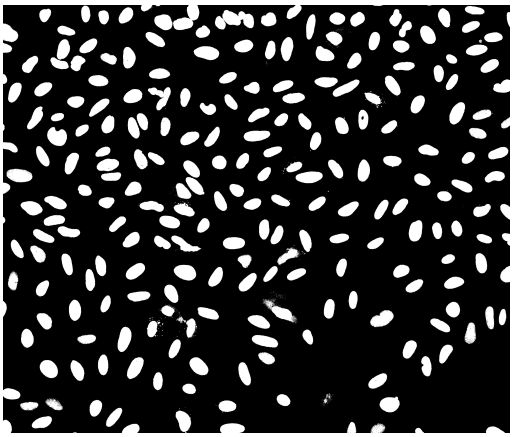
Isola et al. suggest that the generator loss L1 is better than the L2 loss because the L1 loss produces less blurred images. Thus, we tried the L1 loss and the L2 loss on a small version of our problem (seven training images and three epochs).

Figure 7.2 shows that L1 loss is indeed more promising.

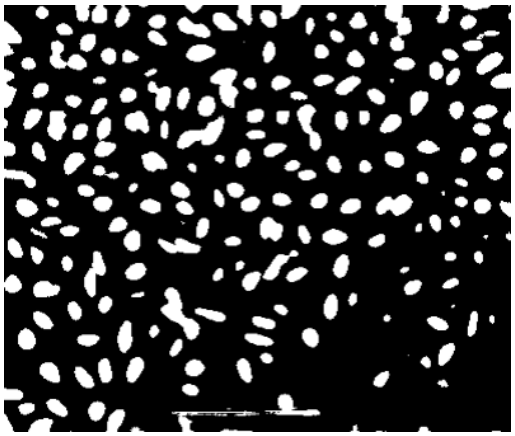
Lastly, we tried the binary cross-entropy loss in order to see if the discriminator can improve the neural network with the pixel-wise loss criterion. (Section 7.5).



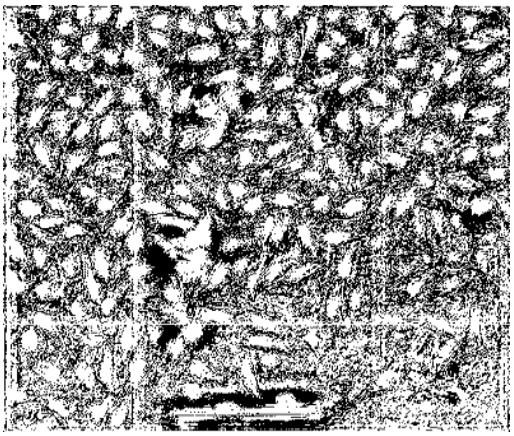
(A) Input contrast microscopy image



(B) Segmentation obtained from the corresponding fluorescence microscopy



(C) Result segmentation from pix2pix neural network with the L1 generator loss



(D) Result segmentation from pix2pix neural network with the L2 generator loss

FIGURE 7.2: The figure illustrates the learning of pix2pix with L1 loss compared to the pix2pix with L2 loss on a restricted training dataset with seven samples and three epochs.

7.4 Impact of Lambda Ratio

In Section 7.2, we presented the generator's training loss as a combination of two losses. The first loss component expresses the output's difference from the reference. The second represents the generator's ability to fool the discriminator into claiming the generator's output as a true reference.

The ratio between these losses could influence the results of the pix2pix approach significantly. In this section, we will therefore inspect whether we can improve our solution by changing the λ. In the skeleton implementation, there was the ratio λ set to the value 100. We suspected this would diminish the impact of the discriminator.

Thus we tried different λ settings to see if the results improve. We present the comparison between different lambda settings in Figure 7.3. Although the difference is not too significant, the λ equal to ten has the best performance amongst the values we tried. However, the difference is not large enough to expect the fine-tuning of λ to improve the results much.

To sum up, the changing of λ did not have a significant impact. However, the λ set to 10 performed better than other settings. Therefore, we will set it to 10 from now on.

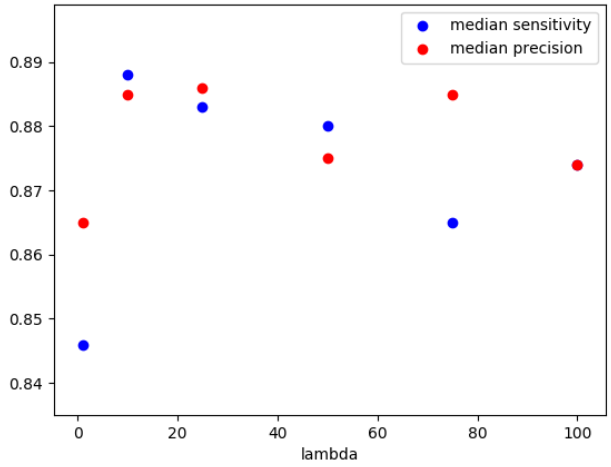


FIGURE 7.3: We see the comparison between pix2pix performance when λ changes. The other pix2pix settings are the same.

7.5 Experimental Evaluation

In this section, we will compare and analyze the results of the pix2pix algorithm with parameters mentioned in Section 7.3 on our test dataset.

We compare the original pixel-wise-based approach with the pixel-wise-loss-based pix2pix. This comparison illustrates the impact of the discriminator presence in pix2pix architecture on the results.

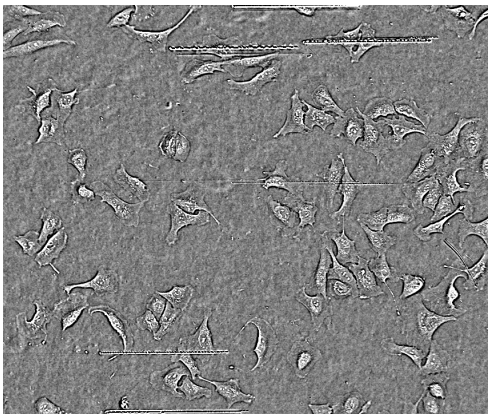
We also compare the pix2pix model with the L1 generator loss with the pix2pix model with pixel-wise-loss (BCE). We show the results in Table 7.1.

method	median sensitivity	median precision	mean $\pm$ SD sensitivity	mean $\pm$ SD precision	median Dice loss
pixelwise-based UNet (no discriminator)	0.652	0.645	0.631 $\pm$ 0.15	0.555 $\pm$ 0.227	0.357
pix2pix, generator loss BCE	0.669	0.729	0.652 $\pm$ 0.18	0.66 $\pm$ 0.192	0.33
pix2pix, generator loss L1	0.874	0.874	0.831 $\pm$ 0.156	0.807 $\pm$ 0.19	0.24

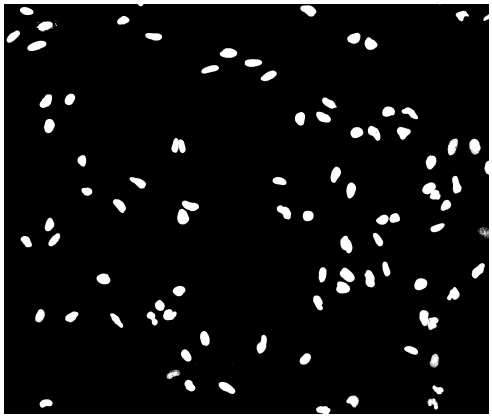
TABLE 7.1: The table shows the comparison between pix2pix with pixel-wise loss and the original pixel-wise-loss-based UNet without discriminator (first two rows). Finally, we show the performance of the pix2pix with L1 generator loss (line 3).

Thus, the suggested $L1$ loss indeed performs the best amongst the losses we tried.

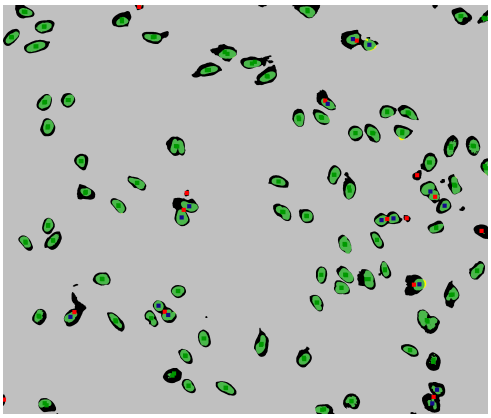
In the following sections, we will try to improve the results of the pix2pix approach further.



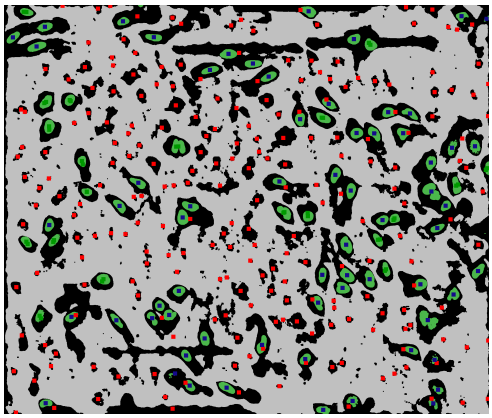
(A) Input contrast microscopy image



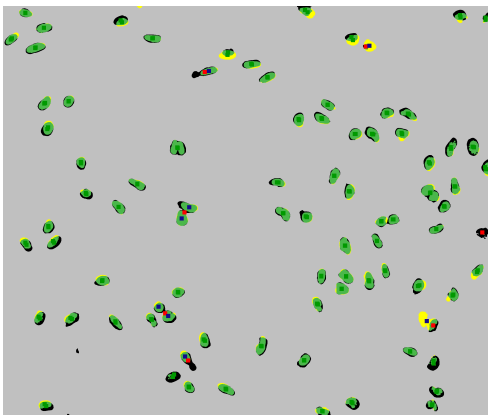
(B) Segmentation obtained from the fluorescence microscopy



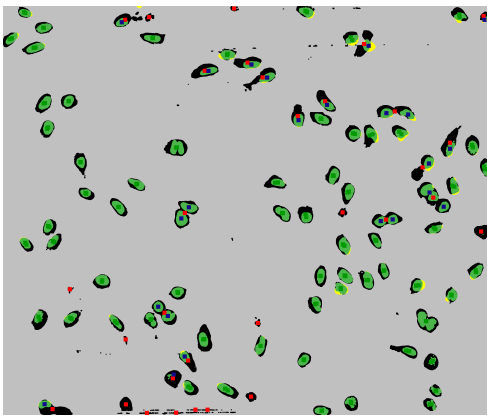
(C) Pix2pix with the binary cross-entropy loss, Adam optimizer



(D) Pix2pix with the binary cross-entropy loss, RMS optimizer



(E) Pix2pix with the L1 generator loss, Adam optimizer

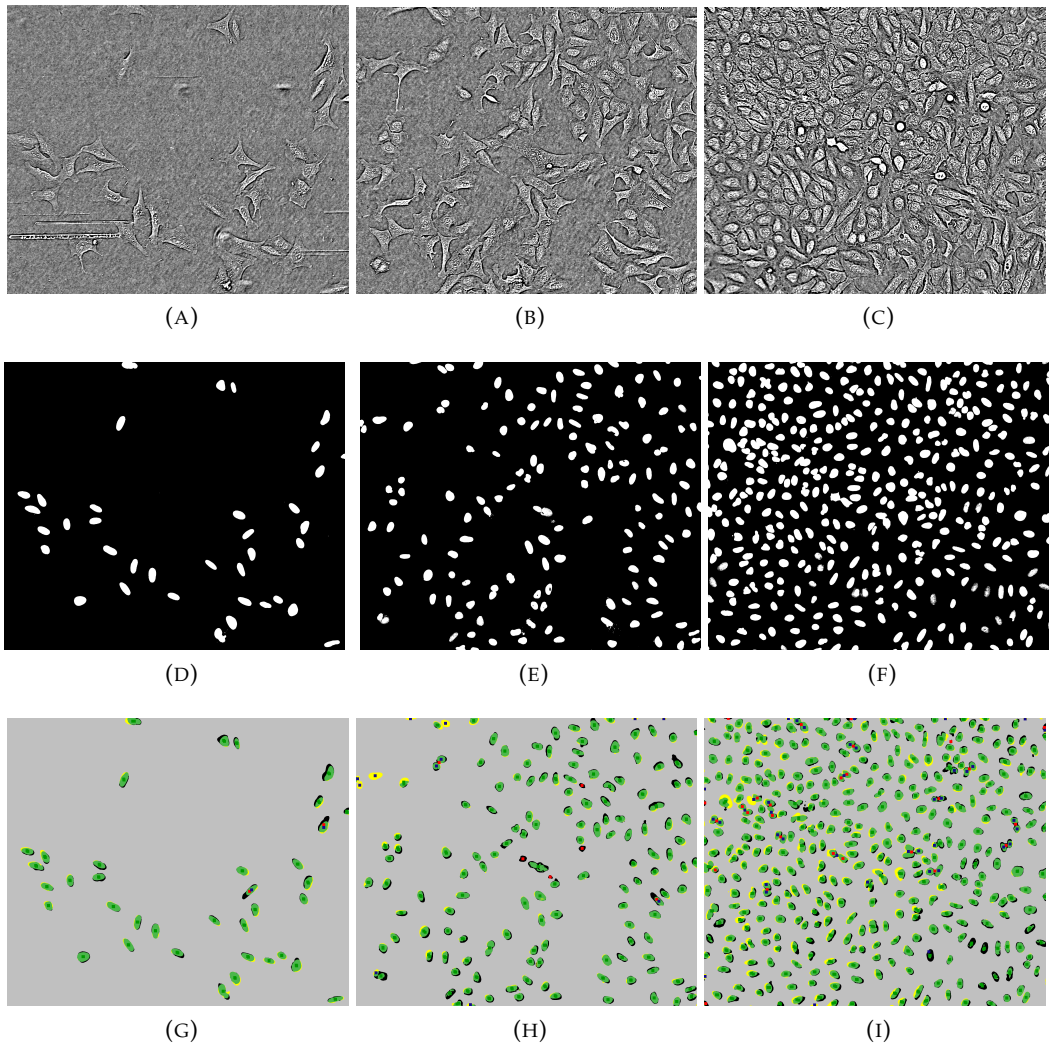


(F) Cross-entropy-loss-based UNet, RMS optimizer

FIGURE 7.4: Figures show the output segmentation of the particular neural network overlapped with a reference segmentation. Each image (7.4c - 7.4f) contains green regions corresponding to the true-positives, black regions representing the false-positives, and yellow regions representing the false-negatives. Moreover, the nuclei centers matched with the reference nuclei centers are marked green, while unmatched are marked red (false-positives) or blue (false-negatives).

Subfigure 7.4e shows the output of the pix2pix NN with the L1 generator loss, and the optimizer preset the same as in the skeleton implementation. Subfigures 7.4c and 7.4d illustrate the difference between the used optimizer settings. The NN with the output in Subfigure 7.4c was trained with the preset optimizer settings as in the skeleton implementation, while the NN with the output in Subfigure 7.4d has the same optimizer settings as the previously used pixel-wise-loss-based UNet.

Subfigure 7.4f show the example result from the pixel-wise-loss-based UNet presented in the previous chapter. The optimizer settings of the UNet (7.4f) and the settings of the pix2pix with output in Subfigure 7.4d are the same. The only difference between 7.4f and 7.4d is the presence of the discriminator in 7.4d.



(A) (B) (C)

(D) (E) (F)

(G) (H) (I)

FIGURE 7.5: The figure shows the alignment of the reference and the generator's output. The matched nuclei pixels are green, while false positive pixels are black and false-negative pixels are yellow. The matched centers are green, while the false-positive centers are red, and false-negative centers are marked blue. We used the L1 generator loss. Subfigures 7.5a-7.5c show the input contrast microscopy images, while Subfigures 7.5d-7.5f show the corresponding masks, and Subfigures 7.5g-7.5i show the output pix2pix segmentation overlapping with the masks.

7.6 Dataset Impact Analysis

We already observed the lower performance of the neural network on the '72h' dataset. Therefore, we select the best-performing pix2pix neural network so far and evaluate it on each dataset. That is, the generator loss is set to L1 loss, and λ is set to 10. We present the results in Table 7.2 and the example segmenations in Figure 7.6.

dataset	median sensitivity	median precision	mean $\pm$ SD sensitivity	mean $\pm$ SD precision	median Dice loss
24h	0.939	0.929	0.925 $\pm$ 0.054	0.909 $\pm$ 0.073	0.16
72h	0.784	0.736	0.625 $\pm$ 0.295	0.631 $\pm$ 0.246	0.353
before	0.9	0.884	0.873 $\pm$ 0.098	0.836 $\pm$ 0.152	0.206
only	0.884	0.896	0.878 $\pm$ 0.042	0.893 $\pm$ 0.032	0.223

TABLE 7.2: The table compares the pix2pix performance separately on each dataset (Training, however, is performed on the combined set). The pix2pix settings are the following: L1 generator loss, 20 training epochs, $\lambda = 10$. The network is trained on the combined datasets while it is evaluated separately on each dataset.

Similarly, as in Section 6.3.5, we see that model performs the worse on the '72h' dataset. We discussed the possible classification problems in Section 6.3.7. Also, we can see in Subfigure 7.6e the classifier under segmented many nuclei. On the other hand, the classifier handles the problematic image with many touching cells nicely (Subfigure 7.6f). When we used the distance-transform-loss-based UNet in the previous chapter, the segmentation of this particular image (Figure 6.10) did not align so well with the reference. However, the problems with the nuclei merging and thus mismatching of the nuclei remain.

The output segmentation on the rest of the datasets is pleasing. We will provide a final analysis and comparison of pix2pix versus distance-transform-based neural network in Section 7.8.

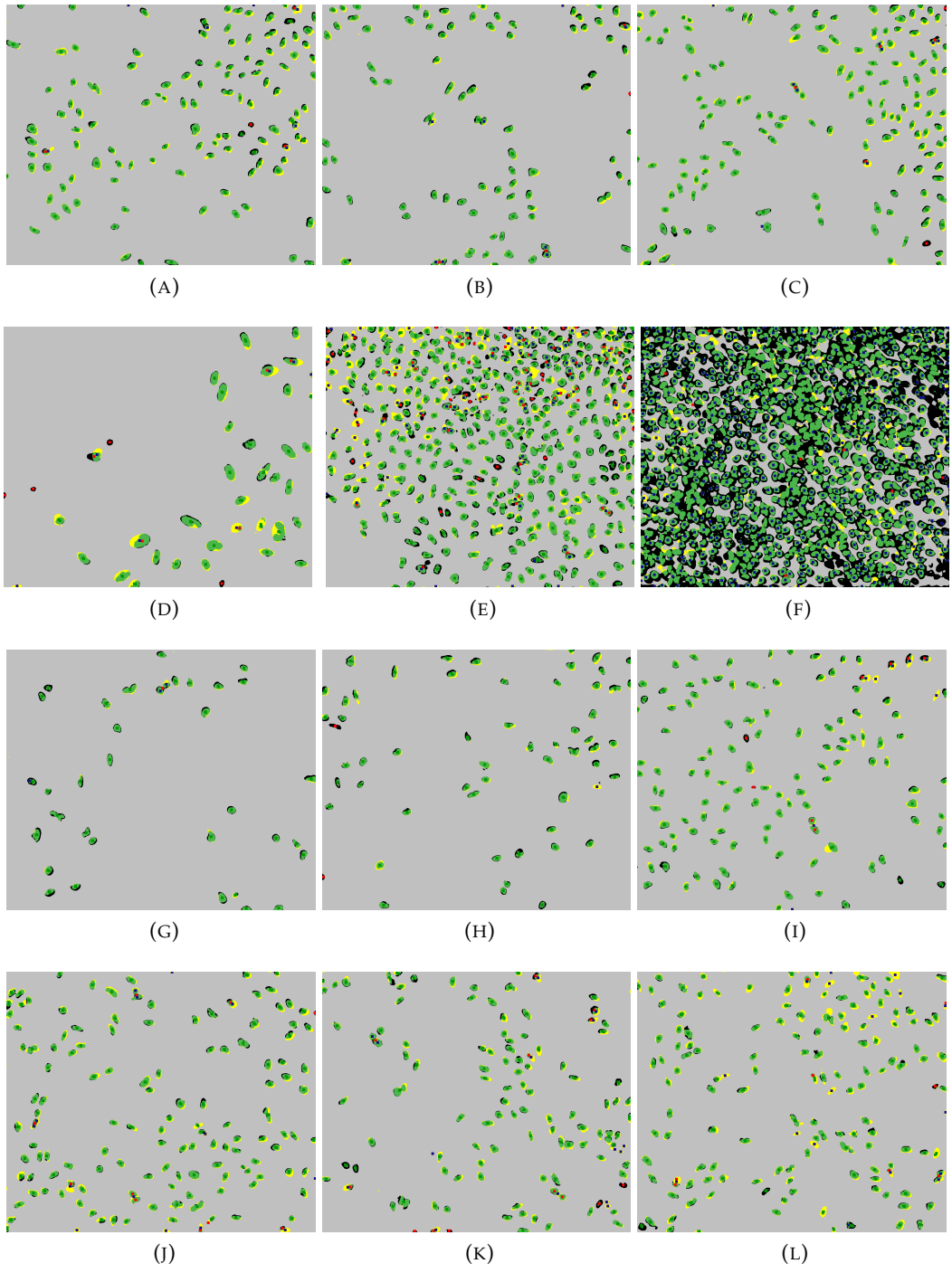


FIGURE 7.6: Figures show the combined output segmentation of the pix2pix network over a reference segmentation. Each image contains green regions corresponding to the true-positives, black regions representing the false-positves and yellow regions representing the false-negatives. Moreover, the nuclei centers matched with the reference nuclei centers are marked green, while unmatched are marked red (false-positives) or blue (false-negatives). Subfigures 7.6a-7.6c show the example output segmentations on the '24h' dataset, while Subfigures 7.6d-7.6f originate from the '72h' dataset, Subfigures 7.6g-7.6i from the 'before' dataset, and Subfigures 7.6j-7.6l from the 'only' dataset.
For more examples of the pix2pix output segmentation see Appendix B.

7.7 Overfit Preventing

In this section, we will apply a technique to prevent the network from overfitting. That is, preventing the network from reducing the training set error while not improving the classification of unknown samples. Namely, we will use the early stopping method described by Goodfellow et al. ([18]).

Early stopping acts as the regularization method (Bishop [2], Sjöberg [46]). The main idea is the following: We create a validation set, which is neither a subset of the training nor testing set. After each training epoch, we observe the loss on the validation set. If the validation set's loss did not decrease, we keep remembering the most successful model (in regards to the validation loss) so far. Otherwise, we assign the current model to be the most successful so far.

Moreover, if the validation loss does not decrease for a certain number of epochs, we stop the training. We can also stop if the improvement is not big enough.

The advantage of this technique is a reasonable determination of the number of epochs and preventing overfitting. On the other hand, validation while learning requires more time.

7.7.1 Early Stopping Results

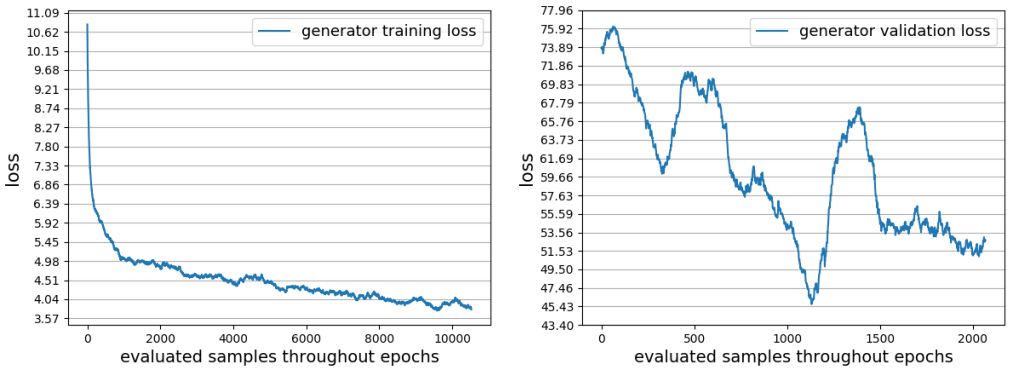
In Figure 7.7, we present the training loss and validation loss development throughout the epochs with the best model so far.

When applying the early stopping, the output classifier is the classifier obtained after 11 training epochs. Hence we compare the quantitative results obtained on the test-set from both classifiers trained on 20 and 11 training epochs. We present the results in Table 7.3.

However, the neural network trained on 20 epochs overperforms the neural network trained on 11 epochs.

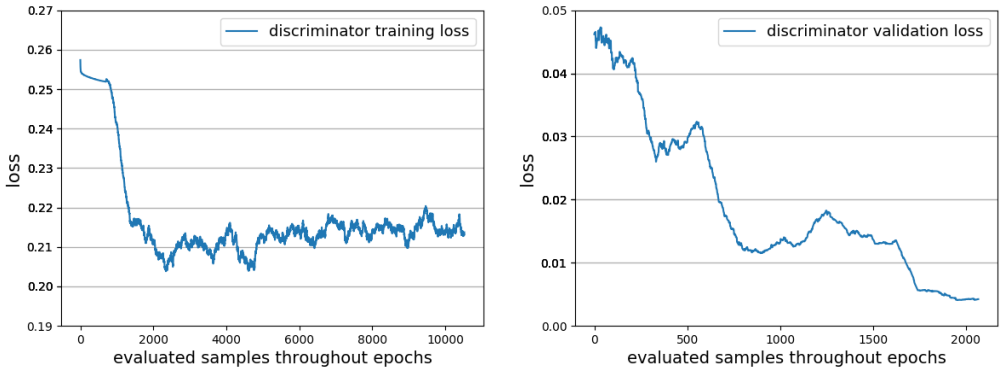
training epochs	median sensitivity	median precision	mean $\pm$ SD sensitivity	mean $\pm$ SD precision	median Dice loss
11	0.85	0.851	0.801 $\pm$ 0.187	0.798 $\pm$ 0.169	0.274
20	0.888	0.885	0.829 $\pm$ 0.196	0.822 $\pm$ 0.184	0.214

TABLE 7.3: The table shows the comparison of output segmentation of pix2pix with 11 epochs (which is the number minimizing the validation loss) and full 20 training epochs. The pix2pix settings are the same (L1 generator loss, $\lambda = 10$).



(A) The generator's training loss

(B) The generator's validation loss



(C) The discriminator's training loss

(D) The discriminator's validaiton loss

FIGURE 7.7: Figures show the moving average of sample loss during training or validation. The averaging window size is set to the size of training (500) and validation set (100), respectively. The generator's loss contains both the similarity loss and the loss L_{fake} for the generated image recognized by the discriminator. The pix2pix model settings are the following: generator loss is L1 loss and $\lambda = 10$. We assume that increasing the number of learning epochs might further improve the results.

7.8 Pix2pix Performance Recapitulation

In previous sections, we introduced the pix2pix approach. After that, we proposed multiple improvements in order to achieve better performance. In this section, we will inspect the results of the best pix2pix model we encountered. That is, pix2pix with the generator L1 loss, λ set to 10 and evaluated on 20 epochs. We also compare the results with the distance-transform-based approach in Table 7.4.

dataset	method	median sensitivity	median precision	mean $\pm$ SD sensitivity	mean $\pm$ SD precision	median Dice loss
24h	p2p	0.939	0.929	0.925 $\pm$ 0.054	0.909 $\pm$ 0.073	0.16
	dst	0.904	0.839	0.897 $\pm$ 0.054	0.764 $\pm$ 0.196	0.215
72h	p2p	0.784	0.736	0.625 $\pm$ 0.295	0.631 $\pm$ 0.246	0.353
	dst	0.724	0.571	0.648 $\pm$ 0.225	0.523 $\pm$ 0.237	0.314
before	p2p	0.9	0.884	0.873 $\pm$ 0.098	0.836 $\pm$ 0.152	0.206
	dst	0.889	0.789	0.86 $\pm$ 0.105	0.703 $\pm$ 0.209	0.278
only	p2p	0.884	0.896	0.878 $\pm$ 0.042	0.893 $\pm$ 0.032	0.223
	dst	0.847	0.815	0.841 $\pm$ 0.044	0.81 $\pm$ 0.045	0.248
combined datasets	p2p	0.888	0.885	0.829 $\pm$ 0.196	0.822 $\pm$ 0.184	0.214
	dst	0.857	0.786	0.815 $\pm$ 0.159	0.703 $\pm$ 0.216	0.249

TABLE 7.4: The table shows the comparison of the pix2pix approach and distance-transform loss-based UNet on both individual and combined datasets. The pix2pix settings are: L1 generator loss, $\lambda = 10$, and 20 training epochs. SD stands for standard deviation.

We can see that the pix2pix approach overperforms the distance-transform loss based UNet in each dataset. Especially, we see a considerable improvement in '72h' dataset precision. Overall performance of the pix2pix approach on all datasets combined is, therefore, better than the performance of the Unet we discussed in the previous chapter.

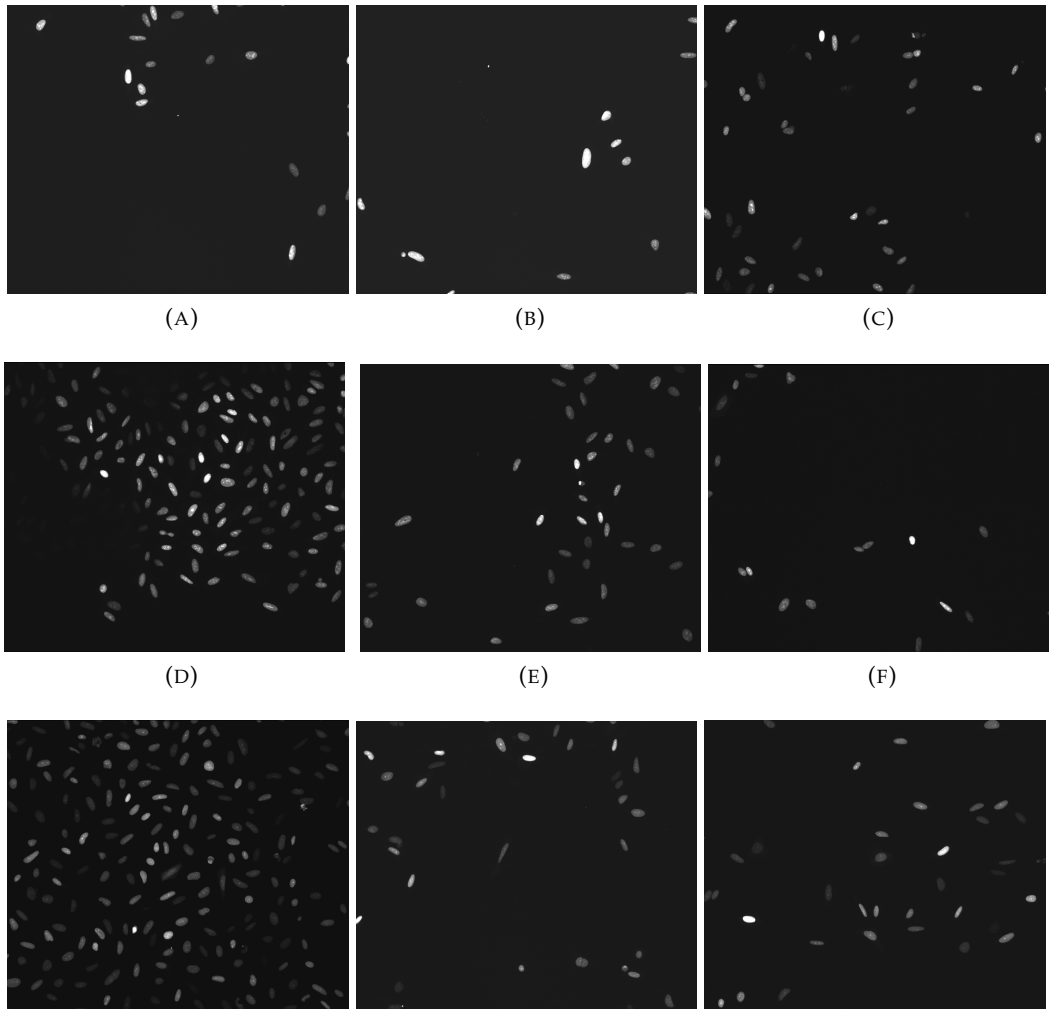
8 Evaluation of the Drug Effects

Images in all datasets can be split into multiple categories according to the biological treatment of the cells. These treatments include Topotecan, Daunorubicin, Etoposide, DMSO, and no treatment. We assume that the treatment may influence some properties of the cells, which can be calculated from the image, such as the density, size, or shape of the cell nuclei.

Figures 8.1, shows the example fluorescence microscopy images from distinct categories. However, we cannot see any significant differences between them.

In this section, we aim to train a classifier that predicts a treatment given a nuclei segmentation. First, we create a classifier on the reference image segmentation (originating from the fluorescence modality).

Then, we compare the performance of the classifier on the reference images with the performance on the neural network output segmentation.

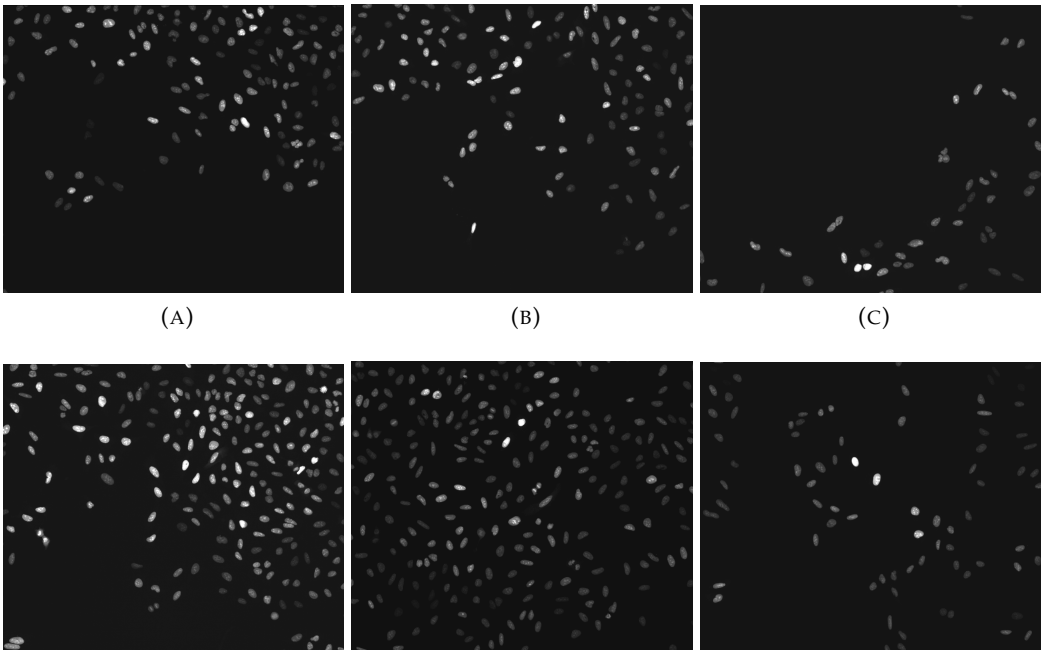


(A) (B) (C)

(D) (E) (F)

(G) (H) (I)

FIGURE 8.1: The figures show fluorescence microscopy images with different treatments. All of the figures originate from the '24h' dataset. Subfigures 8.1a-8.1c show Topotecan treatment, Subfigures 8.1d-8.1f show Daunorubicin treatment, and Subfigures 8.1g-8.1i show Etoposide treatment.



(A) (B) (C)

(D) (E) (F)

FIGURE 8.2: The figures show fluorescence microscopy images with different treatments. All of the figures originate from the '24h' dataset. Subfigures 8.2a-8.2c show the DMSO treatment, while Figures 8.2d-8.2f show the microscopy images without any treatment.

8.1 Preprocessing

We preprocess the segmentations so that each image consists only of connected components representing the nuclei.

Moreover, the nuclei should not touch each other so that we correctly determine their number and size.

The first preprocessing step is noise elimination. That is, we apply morphology closing operation. Furthermore, we eliminate components smaller than one-tenth of the mean nuclei size in the particular image.

We already observed touching nuclei in some of the image segmentation. Thus, the first step before the feature extraction is to split the individual nuclei. There are multiple methods (e.g. [48], [43]) available. We applied the watershed method from the OpenCV library [4]. Generally, the watershed method performs well.

Nonetheless, in several cases, it splits one nucleus into two or does not split two touching nuclei. Figure 8.3 illustrates such phenomenon.

As the touching nuclei are not very common, it might be more damaging to accidentally split one nucleus into more than not performing the watershed at all. Thus, we will evaluate both possibilities and see which leads to the better classification.

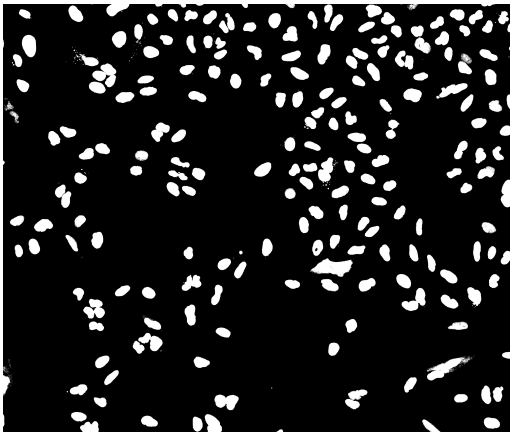
In the next section, we will discuss the image feature extraction.

(A) Original segmentation (zoomed in)

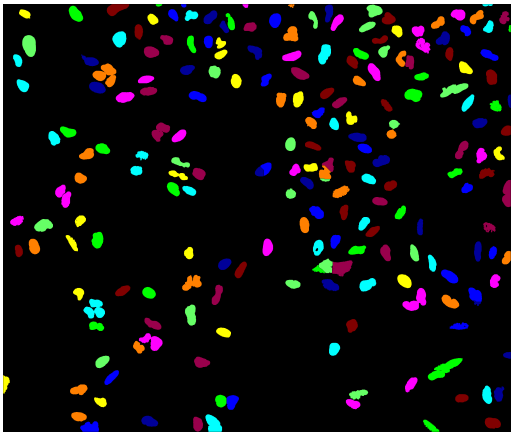
(B) Watershed preprocessing (zoomed in)

(C) Original segmentation

(D) Watershed preprocessing



(E) Original segmentation



(F) Watershed preprocessing

FIGURE 8.3: The figure shows examples of the images with each separated nucleus colored with a different color.
In zoomed Subfigure 8.3b we can see that the watershed nicely split the two nuclei in the bottom right-hand corner, while the two pink merged cells in the top left-hand corner remain merged. Generally, some of the nuclei are split, and some remain attached.

8.2 Feature Extraction

In order to train the classifier, we first need to extract the features from individual images. In this section, we present the selected features. Let us have a connected component C corresponding to a nucleus, which is a set of the (x,y) coordinates belonging to the component (we use the 8-connectivity on the grid). Each image contains multiple such components (nuclei).

We define the component fetures as follows:

1. A_C (area of nucleus) represents the size of the component, $A_C = |C|$.
2. G_C (granularity of nucleus) ([48])
   - Let us define a perimeter P_C as the number of pixels at the edge of the component.
   - $P_C = |\{(x,y) \in C : \exists (x_n, y_n) \notin C : |x - x_n| \leq 1 \wedge |y - y_n| \leq 1\}|$
   - In other words, P_C is the number of such pixels that there exists a pixel in the 8-neighborhood not belonging to the component, $|\ |$ denotes cardnality.
   - Then the granularity of the component is the ratio between the component perimeter and the area.
   - $G_C = P_C / A_C$
3. Hu-invariants L_{23}, L_{24} ([19], [48]) are based on central moments of inertia. Žunić ([58]) says L_{23} can be seen as a circularity measure, because the minimal value $(1/2\pi)$ is obtained for a circular shape. L_{23} and L_{24} are defined as follows:
   - $L_{23} = \eta_{20} + \eta_{02}$
   - $L_{24} = (\eta_{20} + \eta_{02})^2 + 4\eta_{11}^2$
   - $\eta_{pq} = \frac{\mu_{pq}}{(\mu_{00})^{\frac{p+q}{2}+1}}$
   - $\mu_{pq} = \sum_{(x,y)\in C} (x - x_c)^p (y - y_c)^p$
   - (x_c, y_c) equals the center of mass of the component.
     $(x_c, y_c) = (\frac{1}{|C|}\sum_x x, \frac{1}{|C|}\sum_y y)$
4. Major and minor axes lenghts, C_{maj}, C_{min} (as the components resemble ellipses). We compute them as the two times square roots of the coordinates covariance matrix (D) eigenvalues.
   $D = \frac{1}{|C|}\sum_{(x,y)\in C}[x - x_c, y - y_c][x - x_c, y - y_c]^T$, where $[x,y][x,y]^T$ denotes outer product.
   Let $\lambda_1 \geq \lambda_2$ be the eigenvalues of the D.
   $C_{maj} = 2\sqrt{\lambda_1}$, $C_{min} = 2\sqrt{\lambda_2}$.

5. $R_{aspect} = C_{maj}/C_{min}$ denotes the aspect ratio.

We aggregate the extracted features of individual components through each image. Namely, the final features are means, medians, and standard deviations of features 1-5.

Moreover, we aggregate some of the categories through the normalized histogram with a specified range (306 - 11924 for A_C, 50.0 - 784 for P_C, 1.0713 - 37.9643 for aspect ratio) and bin number of value ten. The features aggregated through histograms are A_C, P_C and R_{aspect}.

The last feature is the total nuclei area in the image. That is the sum of individual nuclei areas. The result of the feature extraction is 52 element feature vector for each image.

8.3 Classifier Training

We decided to train an SVM classifier to predict the nuclei treatment. Each image has one of five possible treatments.

However, the standard SVM is meant for two categories. We used the SVM implementation from the sklearn library with an RBF kernel.

First, we created a set of feature vectors F. That is, we applied the processing discussed in Section 8.1 and then the feature extraction as described in Section 8.2 on the reference-segmentation images from the fluorescence modality. Then, we establish a set of feature vectors N obtained from the neural network output images (135 elements). These two sets (F and N) originate from mutually exclusive images (different microscopy samples). After that, we split the set F into the training set (60% of the set F: 165 elements) and the validation set (110 elements).

We then trained the classifier with different regularization constants and compared the performance between training set, validation set, and the performance on the neural network output segmentation.

To fine-tune the parameters on a pairwise classification between Topotecan and Daunorubicin (thus, we eliminate the other elements from the training, validation, and test set).

We define accuracy as the number of correctly determined images divided by the number of the classified images.

As the feature vectors are quite long, we first perform a feature selection. That means we choose only a subset of features that give us the most information. The selection is performed via sklearn method SelectKBest based on χ^2 test between the features. Figure 8.4 shows the influence of the number of selected features on the accuracy. We observe a local maximum at a value of 15. We will use this number for the next experiments. These features include:

- mean, median and standard deviation of the nuclei area (A_C),
- mean and median of the granularity (G_C),

- mean, median, and standard deviation of both axis (C_{maj}, C_{min}),
- mean and median of the aspect ratio R_{aspect},
- total area of nuclei in the image ($\sum_C A_C$),
- and the eight element of the perimeter histogram.

Figure 8.5 shows that the best regularization constant equals 10.1 for polynomial kernel and 16.2 for RBF kernel. The accuracy of the polynomial kernel on the validation set is 58% , while on the neural network output, it reaches 50.3%. On the other hand, the RBF kernel reaches accuracy 73.6% on the validation set, and on the neural network output, it reaches the same accuracy as the polynomial kernel.

The fine-tuning of the second polynomial kernel parameter does not impact results much, as we can see in Figure 8.6

Consequently, we compare the results obtained with and without the preprocessing discussed in Section 8.1. We observe a similar performance of the classifier without the preprocessing and the classifier with the preprocessing. With the preprocessing, we reached the accuracy value 91% on the training set, 74% on the validation set, and 53% accuracy on the neural network output segmentation.

With the preprocessing, we reached the accuracy value 83% on the training set, 74% on the validation set, and 50% accuracy on the neural network output segmentation.

In Table 8.1, we show the pairwise classification performance also for other pairs of treatment.

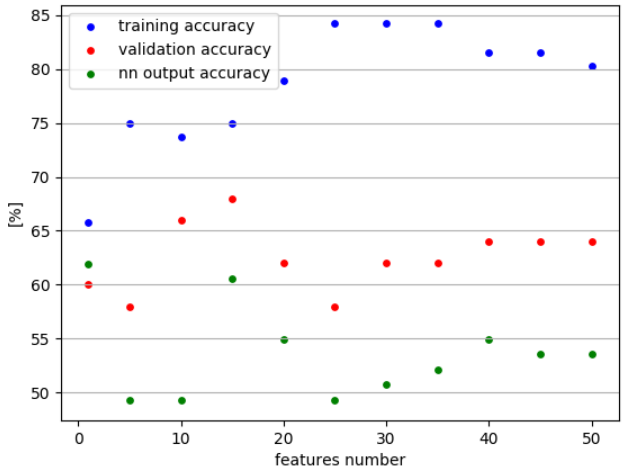
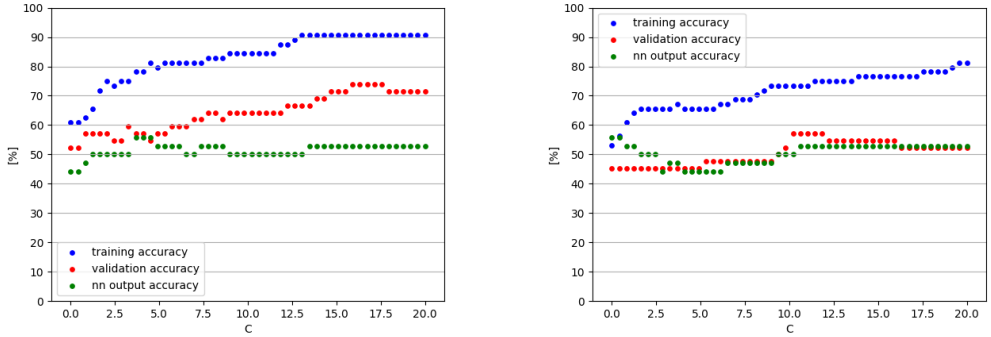


FIGURE 8.4: The figure shows the impact of the number of selected features on the accuracy.



(A) RBF kernel (B) Poylynomial kernel

FIGURE 8.5: The figure shows the impact of the regularization constant on the accuracy.

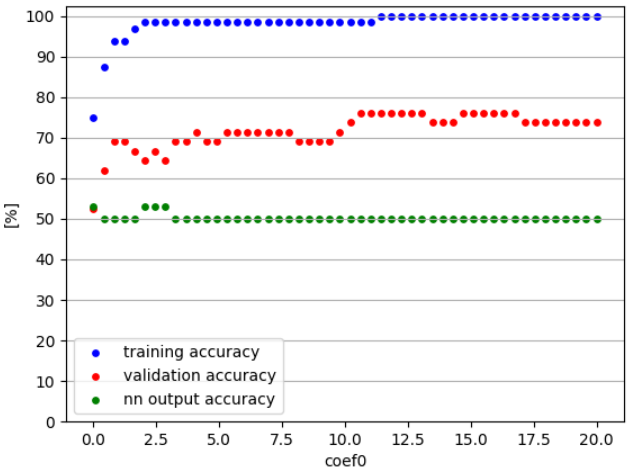


FIGURE 8.6: The figure shows the performance with the polynomial kernel with different coef0.

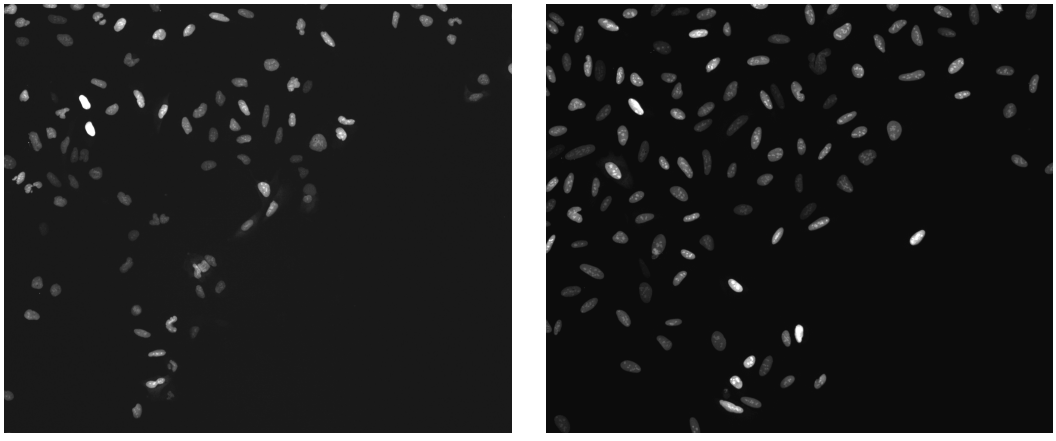
Pairwise comparison			training accuracy	validation accuracy	nn output accuracy
Topotecan	vs	Daunorubicin	0.84	0.52	0.41
Topotecan	vs	Etoposide	0.95	0.85	0.78
Topotecan	vs	DMSO	1.0	0.98	0.79
Topotecan	vs	No treatment	1.0	0.96	0.77
Daunorubicin	vs	Etoposide	0.93	0.95	0.79
Daunorubicin	vs	DMSO	1.0	1.0	0.91
Daunorubicin	vs	No treatment	1.0	1.0	0.95
Etoposide	vs	DMSO	1.0	0.97	0.72
Etoposide	vs	No treatment	0.99	0.98	0.77
DMSO	vs	No treatment	0.8	0.64	0.54

TABLE 8.1: The table illustrates our ability to distinguish between different cell treatments pairwise. We obtain the training and validation dataset by splitting the reference segmentations set. After the model training, we compare the classifier's performance on both the validation dataset and the output of the neural network. Results presented above emerge from the experiment without the watershed preprocessing.

We can see that some of the categories are well distinguished (e.g., Daunorubicin vs. No treatment), while some of the categories are not well distinguishable (e.g., Topotecan vs. Daunorubicin). We assume that the pairs with around 50% accuracy cannot be well-distinguished given the extracted features (the treatment might have a similar effect). We also suspect that the internal structure of nuclei has to be taken into account. Thus, the detailed segmentation of the nuclei components might improve the results in the future. The successfulness of the classification on the fluorescence-based segmentation is better than the results on the neural network output segmentation.

Figure 8.7 illustrates the impact of Daunorubicin on the nuclei after 24 hours.

As for the multiclass classification, we used the one-versus-rest generalization. That means the ultimate classifier consists of multiple one-versus-rest classifiers and



(A) Nuclei before the treatment application (B) Nuclei 24 hours after the Daunorubicin application

FIGURE 8.7: The figure compares the nuclei before and after treatment application.

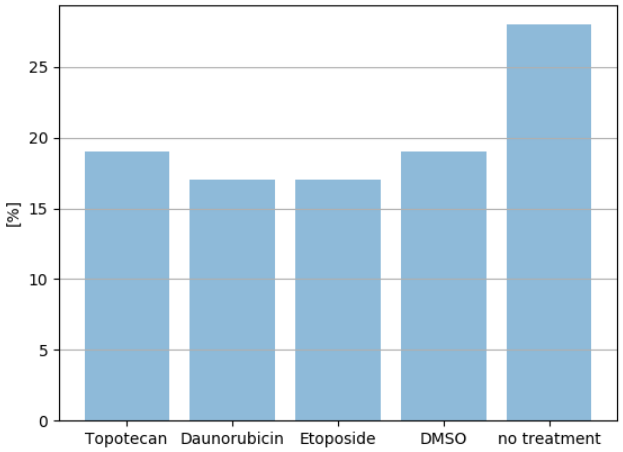


FIGURE 8.8: The figure shows the distribution over the classes in the training dataset.

therefore is capable of predicting more categories.

With the multiclass predictor (with RBF kernel), we obtained the validation accuracy 56%, and neural network output accuracy 44% .

Figure 8.8 shows the distribution over the treatment categories. The dummy classifier predicting the most numerous category has an accuracy of 29% .

Thus, the SVM overperforms the dummy predictor. However, the classifier is not able to predict all the categories well. This might be caused by similar effects of the treatment on the cells. It might be necessary to also study the inner components of nuclei, not only the features we extracted. However, that would need more detailed segmentation. It might also be beneficial to train a different classifier, such as a decision tree, to gain an intuition about the important features.

9 Conclusion

The automatic nuclei detection in the contrast microscopy images removes the necessity of the manual labeling, which is time-consuming and costly. Contrary to fluorescence microscopy, phase contrast microscopy does not require any substance application. However, the segmentation of contrast microscopy images is challenging as the nuclei are not clearly distinguishable.

This thesis focused on the deep learning methods for segmentation learning in contrast-microscopy images. First, we created a reference segmentation from the fluorescence microscopy modality. Hence, we tried multiple segmentation methods on the fluorescence images; e.g. the Otsu segmentation ([39]), maximal stable regions ([36]), or global thresholding (see Section 5 for more details). The segmentation based on the informed global threshold described in Section 5.6 yielded the best results.

Then, we designed a UNet based pipeline for contrast microscopy image processing (Chapter 6). Initially, we trained a pixel-wise-loss-based UNet. We then improved the pixel-wise-based UNet by predicting the distance from the background rather than the binary class label. We evaluated the obtained segmentation quantitatively on all datasets. The best model we trained (distance-transform-loss-based UNet with a binarization threshold equal to 1.6) had median sensitivity 86% and precision 79% . We also recognized some problems in the segmentation, such as insufficient coloring in the fluorescence images as well as many merged cells in the '72h' dataset.

As the second approach, we applied the image-to-image translation model (Chapter 7). We tested the losses (the L1, L2, and the binary cross-entropy generator loss) and optimizers to obtain the best segmentation. Ultimately, the image-to-image translation model overperformed the distance-transform-loss based UNet model, reaching median sensitivity and precision of 89%.

After that, we evaluated the attributes from the image segmentation, such as nuclei shape, size, granularity (see Section 8.3 for more details). We used these features to train an SVM classifier. This is a groundwork for a real application to analyze whether the treatment influences the cell nuclei. The classifier distinguished well between some of the classes (e.g., Daunorubicin vs. no treatment, Daunorubicin vs. DMSO, Daunorubicin vs. Etoposide, Etoposide vs. no treatment, etc.). Some of the treatments were not distinguishable with the given features (Daunorubicin vs. Topotecan, DMSO vs. no treatment). The effect of these treatments might be similar regarding the presented attributes. Table 8.1 shows the full comparison.

In the future, it is possible to make the following improvements:

1. It might be beneficial to segment also the internal structures of the cells. This requires the training data, including the labels of distinct organelles.
2. Another improvement in treatment classification might be involving other classifier types, such as random forests.
3. The classifier can be trained directly on the neural network output segmentation.
4. It would be useful to perform statistical analysis of individual descriptors to determine their relevance.
5. It is also possible to learn descriptors directly instead of applying the currently proposed features.

A Implementation and User Guide

We attach the implementation, user guide, and the obtained results. All of the code is in the archive `implementation` uploaded to KOS. The rest of attached data is in individual archives (described in Appendix B). The rest of attached data is located on the cmp grid (due to the limited upload size in KOS):
`/datagrid/Medical/temporary/microscopy/identifikace_leciv/hana_mertanova/diploma_output/`
README contains the user guide, including requirements, the details on the individual python scripts, , and usage. README_data contains information about the input and output data and visualization.
Figure A.1 shows the diagram of the pipeline steps.
Folder implementation contains the following files:

1. Fluorescence images segmentation:
   - `segmentation_functions.py` : Implementation of distinct segmentation methods.
   - `seg_files.py` : Creates the segmentations of the fluorescence images.
2. UNet:
   - `transf.py` : Implementation of the transformations applied to the images during the dataset loading.
   - `loader.py` : Creates the pytorch Dataset from the input image directory names containing the images or from the list of image names.
   - `parts.py` : Implementation of the individual neural network layers.
   - `model.py` : The pytorch nn class concatenating individual layers.
   - `dice_coef.py` : Contains a function for computing dice coefficient from pytorch tensors.
   - `main.py` : Loads the dataset, trains the UNet network and tests the network on the test set.
   - `visualization_results.py` : Concats multiple images into one visualization.
3. Analysis of the neural network output:
   - `eval_kvantiv_separate2pix.py` : Compares the nn output segmentation with the reference segmentation obtained from the fluorescence images.

- `eval_stats_pix.py` : Processes the input pickle containing the output from eval_kvantiv_separate2pix.py.
- `find_eps.py` : Finds the 1% quantile minimal distance between cells in the images.
- `find_threshold.py` : Finds the best threshold for binarization for distance transform criteria.

4. Pix2pix implementation:

   - Same helper scripts as in 2 (`transf.py`, `dice_coef.py`, `loader.py`, `model.py`, `parts.py`).
   - `models.py`: The pytorch nn class concatenating individual layers of image-to-image architecture.
   - `losses_from_log.py`: Processes the output log of the pix2pix neural network and plots the log progress throughout epochs.
   - `pix2pix_Unet.py`: Loads the dataset, trains the pix2pix network and tests the network on the test set.

5. Treatment classification:

   - `descriptors.py`: Helper script containing support methods for Hu-invariants and eigenvalues.
   - `extract_features.py`: Script that extracts features from each image and saves the output feature data into the pickle.
   - `process_features2.py`: Script that processes the pickle created by feature extraction.

6. `requirements.txt`: pip3 install -r requirements.txt to install necessary libraries

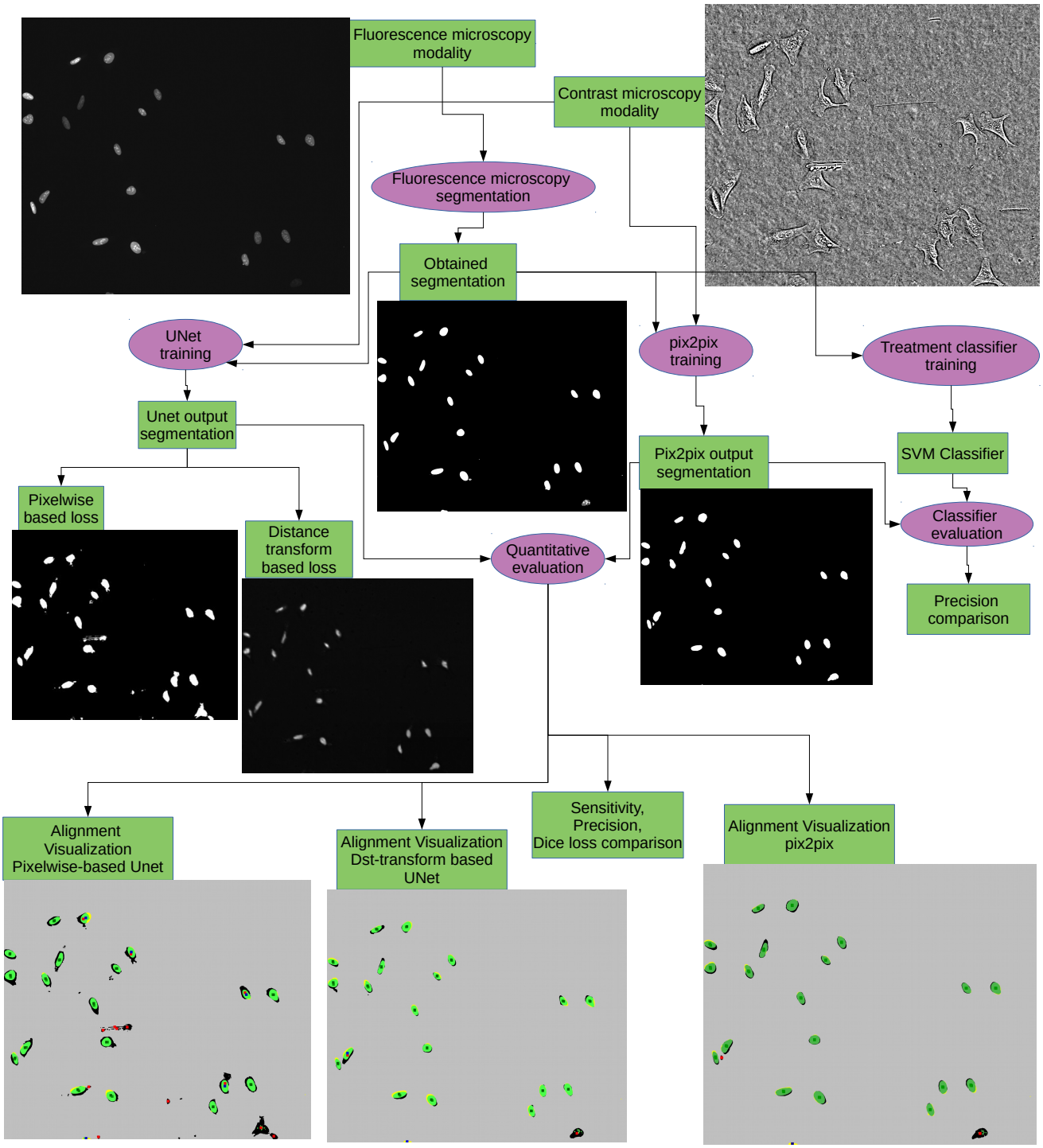


FIGURE A.1: The figure shows the diagram of the distinct steps of our pipeline. Purple ellipses represent the steps we did, while green rectangles represent the inputs/outputs.

B Output Data

We attach the visualization of results, as well as partial outputs and inputs of the designed pipeline in the directory on the cmp-grid (due to the limited upload size in KOS):
`/datagrid/Medical/temporary/microscopy/identifikace_leciv/hana_mertanova/diploma_output/`

README_data contains the description of individual archives.

The folder `diploma_output` on the cmp-grid contains the following results:

1. `results_segmentation.zip`: Different types of segmentations presented in Section 5.

2. `visdstcol.zip`: Output segmentations of pixel-wise-based UNet.

3. `visdstpix.zip`: Output segmentations of the distance-transform-based UNet. Both visdstcol and visdstpix contain images consisting of the six subimages originating from the same sample: In the first row, there is the output from the neural network overlayed over corresponding contrast microscopy (left) versus the reference segmentation overlayed over the same contrast microscopy (right). The second row represents the original input images, i.e., the contrast microscopy (left) and the fluorescence microscopy (right). In the third row, we can see the results from the neural network combined with the contrast microscopy (left) with distinguished FPs, FNs, and TPs versus the original fluorescence combined with the contrast microscopy (right).

4. `vispix2pix.zip`: Folder contains the output segmentation of the pix2pix architecture.
   Each image represents the output segmentation overlapped with the reference obtained from the fluorescence microscopy.
   The matched nuclei pixels are green, while false positive pixels are black and false-negative pixels are yellow. The matched centers are green, while the false-positive centers are red, and false-negative centers are marked blue.

5. `inpall.zip` Input for NN: contains the segmentations obtained from the fluorescence images, as well as paths to the contrast input images.

6. `ouputNN.zip` Output images of the particular NN.

7. `stats.zip` Comparison statistics of NN output and fluorescence segmentations.

8. `extracted_features.zip` Extracted features from fluorescence segmentation images and pix2pix output images.

Bibliography

[1] Pascal Bamford and Brian Lovell. "Unsupervised cell nucleus segmentation with active contours". In: *Signal processing* 71.2 (1998), pp. 203–213.

[2] Christopher M Bishop. "Regularization and complexity control in feed-forward networks". In: *Proceedings International Conference on Artificial Neural Networks ICANN'95* 1 (1995), pp. 141–148.

[3] Pierre Blanchard, Desmond J. Higham, and Nicholas J. Higham. *Accurate Computation of the Log-Sum-Exp and Softmax Functions*. 2019. arXiv: 1909.03469 [math.NA].

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

[5] Jierong Cheng, Jagath C Rajapakse, et al. "Segmentation of clustered nuclei with shape markers and marking function". In: *IEEE Transactions on Biomedical Engineering* 56.3 (2008), pp. 741–748.

[6] Travers Ching et al. "Opportunities and obstacles for deep learning in biology and medicine". In: *Journal of The Royal Society Interface* 15.141 (2018), p. 20170387.

[7] Yunjey Choi et al. "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8789–8797.

[8] Dan Ciresan et al. "Deep neural networks segment neuronal membranes in electron microscopy images". In: *Advances in neural information processing systems* 25 (2012), pp. 2843–2851.

[9] Frederic Commandeur et al. "Deep learning for quantification of epicardial and thoracic adipose tissue from non-contrast CT". In: *IEEE transactions on medical imaging* 37.8 (2018), pp. 1835–1846.

[10] Michal Drozdzal et al. "The importance of skip connections in biomedical image segmentation". In: *Deep learning and data labeling for medical applications*. Springer, 2016, pp. 179–187.

[11] Kenneth W Dunn et al. "Applications of ratio fluorescence microscopy in the study of cell physiology". In: *The FASEB journal* 8.9 (1994), pp. 573–582.

[12] Xinyang Feng et al. "Discriminative localization in CNNs for weakly-supervised segmentation of pulmonary nodules". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2017, pp. 568–576.

[13] Paweł Filipczuk et al. "Computer-aided breast cancer diagnosis based on the analysis of cytological images of fine needle biopsies". In: *IEEE transactions on medical imaging* 32.12 (2013), pp. 2169–2178.

[14] Alberto Garcia-Garcia et al. "A review on deep learning techniques applied to semantic segmentation". In: *arXiv preprint arXiv:1704.06857* (2017).

[15] Yasmeen Mourice George et al. "Remote Computer-Aided Breast Cancer Detection and Diagnosis System Based on Cytological Images". In: *IEEE Systems Journal* 8.3 (2014), pp. 949–964. DOI: 10.1109/JSYST.2013.2279415.

[16] Swarnendu Ghosh et al. "Understanding deep learning techniques for image segmentation". In: *ACM Computing Surveys (CSUR)* 52.4 (2019), pp. 1–35.

[17] Ian Goodfellow et al. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016.

[18] Ian Goodfellow et al. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016, pp. 241–249.

[19] A Ardeshir Goshtasby. *Image registration: Principles, tools and methods*. Springer Science & Business Media, 2012.

[20] Hayit Greenspan, Bram Van Ginneken, and Ronald M Summers. "Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique". In: *IEEE Transactions on Medical Imaging* 35.5 (2016), pp. 1153–1159.

[21] Hua He, Chao Xie, and Jicun Ren. "Nonbleaching fluorescence of gold nanoparticles and its applications in cancer cell imaging". In: *Analytical chemistry* 80.15 (2008), pp. 5951–5957.

[22] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[23] Mohammad Hesam Hesamian et al. "Deep learning techniques for medical image segmentation: achievements and challenges". In: *Journal of digital imaging* 32.4 (2019), pp. 582–596.

[24] Haigen Hu et al. "MC-Unet: Multi-scale Convolution Unet for Bladder Cancer Cell Segmentation in Phase-Contrast Microscopy Images". In: *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE. 2019, pp. 1197–1199.

[25] Phillip Isola et al. "Image-to-image translation with conditional adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1125–1134.

[26] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. [Online; accessed 3-May-2019]. 2001–. URL: http://www.scipy.org/.

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[28] Mineichi Kudo and Jack Sklansky. "Comparison of algorithms that select features for pattern classifiers". In: *Pattern recognition* 33.1 (2000), pp. 25–41.

[29] Harold W Kuhn. "The Hungarian method for the assignment problem". In: *Naval research logistics quarterly* 2.1-2 (1955), pp. 83–97.

[30] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[31] Xiaomeng Li et al. "H-DenseUNet: hybrid densely connected UNet for liver and tumor segmentation from CT volumes". In: *IEEE transactions on medical imaging* 37.12 (2018), pp. 2663–2674.

[32] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. "Unsupervised image-to-image translation networks". In: *arXiv preprint arXiv:1703.00848* (2017).

[33] Katerina Lomanov et al. "Cell detection with deep convolutional networks trained with minimal annotations". In: *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*. IEEE. 2019, pp. 943–947.

[34] Le Lu et al. *Deep Learning and Convolutional Neural Networks for Medical Imaging and Clinical Informatics*. Springer, 2019.

[35] Jihene Malek et al. "Automated breast cancer diagnosis based on gvf-snake segmentation, wavelet features extraction and fuzzy classification". In: *Journal of Signal Processing Systems* 55.1 (2009), pp. 49–66.

[36] Jiri Matas et al. "Robust wide-baseline stereo from maximally stable extremal regions". In: *Image and vision computing* 22.10 (2004), pp. 761–767.

[37] Damian J Matuszewski and Ida-Maria Sintorn. "Minimal annotation training for segmentation of microscopy images". In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. IEEE. 2018, pp. 387–390.

[38] Peter Naylor et al. "Segmentation of nuclei in histopathology images by deep regression of the distance map". In: *IEEE transactions on medical imaging* 38.2 (2018), pp. 448–459.

[39] Nobuyuki Otsu. "A threshold selection method from gray-level histograms". In: *IEEE transactions on systems, man, and cybernetics* 9.1 (1979), pp. 62–66.

[40] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[41] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[42] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

[43] Leila Shafarenko, Maria Petrou, and Josef Kittler. "Automatic watershed segmentation of randomly textured color images". In: *IEEE transactions on Image Processing* 6.11 (1997), pp. 1530–1544.

[44] Dinggang Shen, Guorong Wu, and Heung-Il Suk. "Deep learning in medical image analysis". In: *Annual review of biomedical engineering* 19 (2017), pp. 221–248.

[45] Chuen-Kai Shie et al. "Transfer representation learning for medical image analysis". In: *2015 37th annual international conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE. 2015, pp. 711–714.

[46] Jonas Sjöberg and Lennart Ljung. "Overtraining, regularization and searching for a minimum, with application to neural networks". In: *International Journal of Control* 62.6 (1995), pp. 1391–1407.

[47] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image processing, analysis, and machine vision*. Cengage Learning, 2014, pp. 175–320.

[48] Jan Švihlík, Jan Kybic, and David Habart. "Automated separation of merged Langerhans islets". In: *Medical Imaging 2016: Image Processing*. Vol. 9784. International Society for Optics and Photonics. 2016, p. 978438.

[49] Bernard Valeur and Jean-Claude Brochon. *New trends in fluorescence spectroscopy: applications to chemical and life sciences*. Vol. 1. Springer Science & Business Media, 2012.

[50] Tomas Vicar et al. "Cell segmentation methods for label-free contrast microscopy: review and comprehensive comparison". In: *BMC bioinformatics* 20.1 (2019), pp. 1–25.

[51] Francesco Visin et al. "Renet: A recurrent neural network based alternative to convolutional networks". In: *arXiv preprint arXiv:1505.00393* (2015).

[52] Stéfan van der Walt et al. "scikit-image: image processing in Python". In: *PeerJ* 2 (June 2014), e453. ISSN: 2167-8359. DOI: 10.7717/peerj.453. URL: https://doi.org/10.7717/peerj.453.

[53] Pin Wang et al. "Automatic cell nuclei segmentation and classification of breast cancer histopathology images". In: *Signal Processing* 122 (2016), pp. 1–13.

[54] Zhuo Wang et al. "Label-free intracellular transport measured by spatial light interference microscopy". In: *Journal of biomedical optics* 16.2 (2011), p. 026019.

[55] Xiangchun Xiong et al. "Analysis of breast cancer using data mining & statistical techniques". In: *Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Network*. IEEE. 2005, pp. 82–87.

[56] Bolei Zhou et al. "Learning deep features for discriminative localization". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2921–2929.

[57] Zongwei Zhou et al. "Unet++: A nested u-net architecture for medical image segmentation". In: *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer, 2018, pp. 3–11.

[58] Joviša Žunić, Kaoru Hirota, and Paul L Rosin. "A Hu moment invariant as a shape circularity measure". In: *Pattern Recognition* 43.1 (2010), pp. 47–57.