Detection and segmentation of macrophages in Quantitative Phase Images by Deep Learning using a Mask Region-based Convolutional Neural Network

Tobias Kutscher

Dissertation presented as the partial requirement for obtaining a Master's degree in Data Science and Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

DETECTION AND SEGMENTATION OF MACROPHAGES IN QUANTITATIVE PHASE IMAGES BY DEEP LEARNING USING A MASK REGION-BASED CONVOLUTIONAL NEURAL NETWORK

by

Tobias Kutscher

Dissertation presented as the partial requirement for obtaining a Master's degree in Data Science and Advanced Analytics, specialization in Data Science

Advisor: Mauro Castelli

Co-Advisor: Kai Eder

March 2021

ABSTRACT

Quantitative Phase Imaging (QPI) has been demonstrated to be a versatile tool for minimally invasive label-free imaging of biological specimens and time-resolved cellular analysis. RAW 264.7 mouse macrophages were imaged by Digital Holographic Microscopy (DHM), an interferometry-based variant of QPI, in toxicological studies and cellular growth experiments. Robust detection and segmentation of cells in QPI images by Deep Learning facilitates automated data evaluation of images in high throughput microscopy. Detection, segmentation and the subsequent analysis of single cellular specimens in QPI images yields essential toxicity related physical parameters like the dry mass on the single-cell level. Deep Learning models, such as the Mask Region-based Convolutional Neural Network (Mask R-CNN), were proven to achieve robust results for object detection in fluorescence microscopy images. Thus, a Mask R-CNN was applied with the aim to obtain deeper cellular knowledge from DHM QPI images. This work shows that the combination of label-free DHM and a state-of-the-art Deep Learning model achieves reliable machine-generated data on the single-cell level and prospects to enhance the information as well as the quality of physical data that can be extracted from QPI images of biomedical experiments and label-free high throughput microscopy.

KEYWORDS

Deep Learning; Mask R-CNN; Digital Holographic Microscopy; Quantitative Phase Imaging; Biomedical Imaging; Cell Detection and Segmentation

INDEX

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

Mask R-CNN	Mask Region-based Convolutional Neuronal Network
QPI	Quantitative Phase Imaging
DHM	Digital Holographic Microscopy
MLP	Multilayer perceptron
SGD	Stochastic Gradient Descent
CNN	Convolutional Neural Network
SVM	Support Vector Machine
ROI	Region of Interest
RPN	Region Proposal Network
IoU	Intersection over Union
NMS	Non-Maximum Suppression
N cells	Normal macrophage cells
M cells	Multi-shaped macrophage cells
FPN	Feature Pyramid Network
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
MAE	Mean Absolut Error
MAPE	Mean Absolute Percentage Error

1. INTRODUCTION

Deep Learning is becoming increasingly popular, as it has been able to solve a large variety of problems from different research areas. Nowadays, Deep Learning is used for simple classification and regression, but also for tasks like natural language processing, speech recognition or image processing (Goodfellow et al., 2016). As modern computer systems generate an increasing amount of data, Deep Learning can be used to handle the analysis of these rising amounts. The amount of data that is generated with state-of-the-art analytical techniques, especially in the field of cell biology, exceeds the limits for manual analysis, since imaging systems can easily create terabytes of primary research data. The automated analysis of microscopy images is fundamentally important for drug discovery, medical diagnosis and disease research. Cell detection, segmentation and quantification are important aspects for the research of diseases, like cancer (Xing et al., 2018). One of the most common imaging techniques in biological laboratories is fluorescence microscopy. For cell counting in fluorescence microscopy images, the nucleus of each cell is commonly stained with a DNA stain. Simple pixel-value threshold filters are then used to segment the fluorescence signals from the nuclei from the remaining background. Such processes with limited reliability can bias downstream analysis (Caicedo et al., 2018). Fortunately, Deep Learning algorithms, such as the Mask Region-based Convolutional Neuronal Network (Mask R-CNN), were proven to be able to solve the cell detection and segmentation task in fluorescence microscopy in a reliable manner (Caicedo et al., 2019). However, fluorescence microscopy has limitations, as fluorescent proteins can have various effects on cell cultures. Fluorescent proteins can impair the viability or growth of cell cultures, which again can bias the analysis (Jensen, 2012). In contrast, label-free microscopy enables imaging without employing exogenous contrast agents, as opposed to imaging techniques such as fluorescence or histological staining (Pavillon et al., 2014). Quantitative Phase Imaging (QPI) by using Digital Holographic Microscopy (DHM) is a technique for label-free imaging of biological specimens and cellular analysis with minimal impact on cellular structures as it is applied in transmission with low laser light intensities (Park et al., 2018) (Kemper & Bally, 2008). This thesis aims to clarify if Deep Learning can be used for a reliable automated analysis process of DHM image data on a single-cell level. In particular, this work evaluates the analysis process in terms of cell detection, segmentation, classification and quantification in DHM QPI images, which has the potential to accelerate cell biology research.

1.1. Thesis Objective

This thesis aims to answer several research questions by developing an automated Deep Learning system that is able to detect, segment, classify and quantify two different phenotypes of RAW 264.7 mouse macrophages in QPI images, generated by DHM. The detection and subsequent segmentation of single cellular specimens in QPI images yields essential toxicity related physical parameters like the dry mass on the single-cell level. These parameters highly enrich the information available from biomedical experiments. In this work, a Mask R-CNN Deep Learning model is investigated to achieve the desired goal and answer the research questions.

1.2. Research Questions

To evaluate if Deep Learning techniques can be used for the reliable detection, segmentation, quantification and classification of different phenotypes of macrophage cells in QPI images, generated by DHM, multiple research questions will be answered:

1. **Is it possible to distinguish between two phenotypes of macrophage cells using Deep Learning approaches?**
2. **Is it possible to segment the macrophage cells on a single cell level using Deep Learning approaches?**
3. **Can a Deep Learning approach be used to detect and segment individual cells, even when cells are growing in clusters and overlapping?**
4. **Is it possible to achieve the detection, segmentation and classification process with only a single Deep Learning algorithm or is a combination of multiple Deep Learning algorithms required?**
5. **How well does the detection and segmentation process perform on label-free DHM QPI images?**

2. THEORETICAL BACKGROUND

2.1. MACHINE LEARNING

Machine Learning is the process of computer programs and algorithms learning patterns in data (Mitchell, 1997). The algorithms are designed in a way that they generalize patterns from the data and that their performance increases with the increasing amount of data they are exposed to. Especially due to the increasing amount of data, which is generated by modern technology, Machine Learning has become an important tool to analyze these rising amounts of data. Once a Machine Learning algorithm is able to generalize the underlying patterns, these can be used to predict certain outcomes of new and unseen data. Machine Learning has proven to be able to predict recovery rates of pneumonia patients, to detect fraudulent use of credit cards, to drive autonomous vehicles and to outperform humans in games, like backgammon (Mitchell, 1997). Machine Learning can be separated into three different fields: Supervised Learning, Unsupervised Learning and Reinforcement.

2.1.1. Supervised Learning

In supervised learning, the algorithms try to generalize patterns in the data based on a target variable (James et al., 2013). The learned patterns can then be used for predicting the value of the target variable for observations where the value of the target variable is unknown. In other words, for each observation of the predictor measurement, there is an associated response measurement. A machine learning model, which is fitted on data, relates the response to the predictors, intending to accurately predict the response for future observations or better understanding the relationship between the response and the predictors. Moreover, supervised learning can be separated into regression and classification. In a regression machine learning model, the algorithm learns and predicts the value of a numeric target variable. An example would be the regression of the age of a person, based on variables describing the person. In classification, the target value is categorical, e.g. the gender.

2.1.2. Unsupervised Learning

In contrast to supervised learning, in unsupervised learning, a target variable does not exist (James et al., 2013). Unsupervised learning algorithms seek to understand relations between variables or observations by observing the values of these variables so that they can be separated into groups. Grouping similar variables or grouping similar observations is possible. This can be used for grouping customers who have similar buying behaviors.

2.1.3. Reinforcement Learning

Reinforcement Learning is a reward- and penalty-based learning system (Mitchell, 1997). The learning algorithm tries to solve a specific task and receives positive and negative feedback for respective solutions. Regarding the feedback the algorithm receives, it can figure out ways to solve the desired task by itself, inside the controlled environment.

2.2. Deep Learning

Deep Learning is a subset of Machine Learning. Deep Learning algorithms are combinations of neural networks (Goodfellow et al., 2016). The idea behind that is to mimic the human brain. The combination of neurons can be so big and “deep” that Deep Learning systems are mainly black-box algorithms. This means that it is not fully comprehensible what patterns the algorithm exactly learns. Deep Learning can be used for a large number of tasks: from regression and classification to natural language processing and computer vision.

2.2.1. Artificial Neural Networks

The perceptron is the most basic neural network (Goodfellow et al., 2016). It is a mathematical function mapping a set of input values to an output value. A combination of multiple neurons is called a multilayer perceptron (MLP). The neuron weights the different input values and calculates a combined value out of them. Then a so-called activation function is used in every neuron to transform the values into a linear scale. The output value is then compared to the true value and the error is calculated. For calculating an error, an error function (also called loss function) needs to be designed specifically, depending on the goal of the neural network. The weights of the neurons are updated based on the chosen learning rate. This process is called backpropagation. The learning rate is a parameter which is responsible for deciding how much the error should influence the new weights for the neurons. Consequently, a Deep Learning system is a complex function formed by many simpler functions, by providing a new representation of the input. The idea is to learn a generalized function, which describes the input data in, for example, different categories and is then able to classify new data into one of these categories.

2.2.2. Stochastic Gradient Descent

Gradient descent is the process used for the optimization of the neural network (Goodfellow et al., 2016). In particular, it is responsible for the neuron weight updating process. The weights are updated in a way that the error of the loss function will decrease. This is done by calculating the error of the current neural network prediction of an input and the ground truth label of this input. The weights are then updated based on the learning rate. Stochastic Gradient Descent (SGD) is an adaption of Gradient Descent. In SGD, not all possible inputs are used to update the weights in one training iteration, only a small subset of inputs, called mini-batch, is used. The idea behind SGD is that the time and computational complexity for a single gradient step greatly increases with an increasing amount of training data. Thus, SGD helps to decrease the computational complexity.

2.2.3. Momentum

Momentum is a method to accelerate the learning process (Goodfellow et al., 2016). Especially when using SGD, the training process can be time-consuming as the gradients are updated in very small steps with respect to the mini-batches. Momentum helps to overcome this by accumulating a moving average of the past gradients to move in their direction. Averaging out big steps of the gradients when using mini-batches reduces the oscillation of the learning part and can increase the time to find an optimum. This is useful when the gradients face high curvature, small but consistent gradients, or noisy gradients. *Figure 1* shows the path of the gradients without momentum (black arrows) and the path when using momentum (red path). It can be observed that the path is smoother towards the optimum. The momentum parameter α is important, as it is the smoothing parameter. It can be seen as averaging over last 1 / (1 - α) points of steps in the gradient descent process. A common value of α is 0.9, meaning that it averages over the last 10 (1/1-0.9) steps.

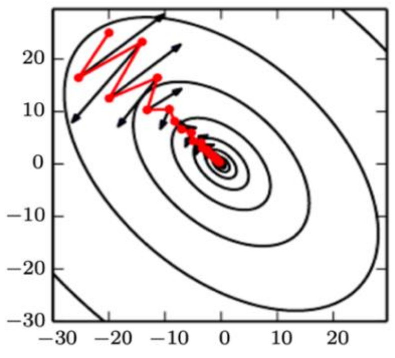

Figure 1: Momentum

2D Illustration of the momentum process within a SGD optimization (Goodfellow et al., 2016).

2.2.4. Weight Decay

Weight decay is a regularization method, which is also called L2 regularization or ride regression (Goodfellow et al., 2016). It reduces complexity to better generalize the learning process from the data by penalizing large weights with a regularization term in the loss function. The addition of the regularization term modifies the learning rule to multiplicatively shrink the weight vector by a constant factor on each step, before performing the usual gradient update. Weight decay functions in a way, that weights corresponding to unimportant directions of the gradient and do not contribute to reducing the loss function are decayed away throughout the training process.

2.3. CONVOLUTIONAL NEURONAL NETWORKS

Convolutional Neural Networks (CNN) are a specific kind of neural networks and visualized in *Figure 2* (Goodfellow et al., 2016). They can be used when the input data has a grid-like topology. For example, time-series data can be interpreted as a 1D grid with values of regular time intervals. However, CNN's are mainly used for image processing, as image data can be thought of as 2D grids of pixel-values. CNN's are very popular, as they have been tremendously successful in various applications. In a traditional fully connected neural network all neurons of one layer are connected to the previous and subsequent layer. In other words, every output unit interacts with every input unit. In fully connected neural networks every pixel-value would be processed individually. This would result in many parameters for high-resolution images and would make it impossible to capture the sparse and local information of connectivity of pixels. CNN's overcome these problems with a mathematical matrix operation called "convolution". Being responsible for the naming of these types of neural networks, this operation is performed in order to process data in a grid-like structure (Goodfellow et al., 2016).

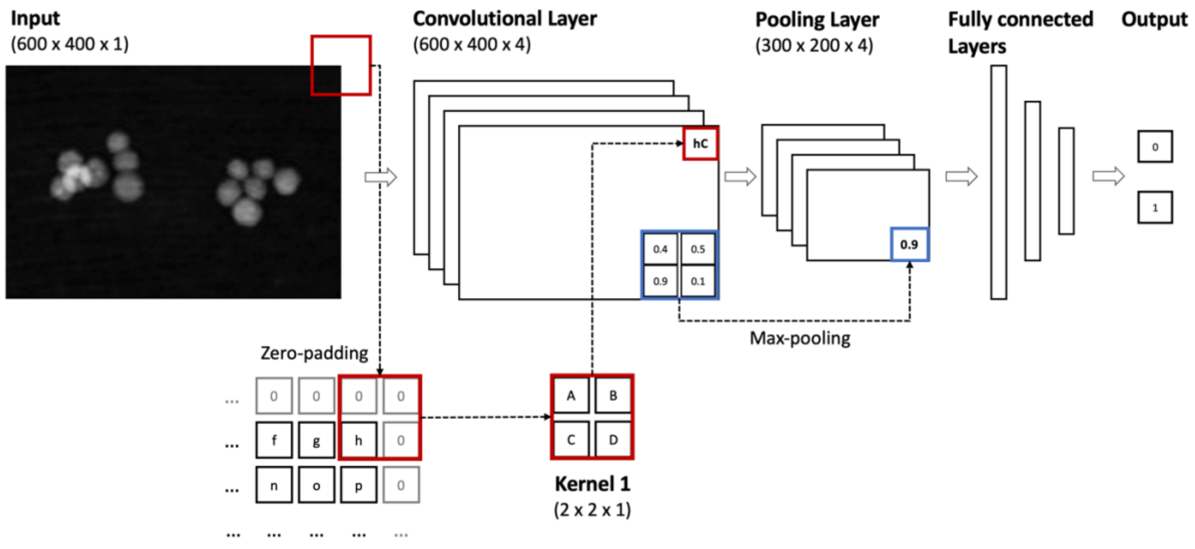

Figure 2: CNN example for image classification

In total, four convolutional kernels of size (2 x 2 x 1) are applied to the input image of size (600 x 400 x 1). Zero-padding is applied to the input pixel-values for the convolutional operation in the bottom left. One example sliding step of the first Kernel with the shape (2 x 2 x 1) is visualized. The element-wise products of the kernel weights and the overlapping pixel-values of the input image are calculated and the sum of the products (hC) is outputted to the resulting feature map, followed by the application of an activation-function. The third dimension of the feature map is four, as four kernels are applied in this example. Zero-padding is responsible for the equal shape of the first two dimensions of the feature map (600 x 400 x 4) and the input image. Max-pooling is applied in the subsequent pooling layer. The pooling window converts an (2 x 2) input into an (1 x 1) output by outputting the maximum value of 0.9. Pooling is performed on all feature map channels. The subsequent fully connected layers convert the three-dimensional feature map of the pooling layer into a one-dimensional output for a classification output (Goodfellow et al., 2016) (Modified figure version based on: Schilling, 2020).

2.3.1. Convolution

Aiming at capturing the sparse connectivity of pixels, the convolutional matrix operation is processed with a kernel matrix, which is smaller than the input matrix (Goodfellow et al., 2016). This matrix is sliding over the entire input matrix and multiplies it with the pixel-values element wise and then outputs the sum of these multiplications. With this process, the convolution operation transforms the input feature map to a reduced output feature map. In contrast to fully connected neural networks, in CNN's a pixel of the output feature map is only connected to a subset of pixels of the input feature map and is not processed individually. This process results in fewer parameters and lower computational complexity. In addition, this process enables the model to extract local features. An input image can have thousands or millions of pixels and with the convolution operation, it is possible to detect local and important features that occupy only tens or hundreds of pixels, like edges or curves. Typically, CNN's consist of multiple convolutional layers, which use the output features of the previous convolutional operation as input features. As a result, subsequent extracted features are created from previous features, which allows the CNN to learn more complex shapes (Patterson & Gibson, 2017) (Goodfellow et al., 2016). Zero-padding is an essential feature for the convolutional step in CNN's and it is used to pad zeros to the input to enlarge the size. This process is visualized in *Figure 2.* Zero-padding is essential to not shrink the width of the representation by one pixel less than the kernel width at each convolutional layer. Finally, it is utilized to fully control the size of the output feature map by setting the window size.

2.3.2. Pooling

Besides the convolution operation, CNN's also use so-called pooling operations (Goodfellow et al., 2016). A pooling operation reduces the complexity by outputting a summary statistic of the nearby values. The window-size of this operation determines how many values are considered for complexity reduction. Further, the pooling window slides through the whole feature map and applies a pooling function. Max-pooling outputs the highest value of the currently observed window, average-pooling outputs the mean value of the values within the current window. Pooling layers are typically inserted between convolutional layers and aim to make the CNN more invariant to small changes of the input (Patterson & Gibson, 2017). For example, the CNN models, which are generally used for image classification, do not need to know the exact location of the eyes with pixel-perfect accuracy to detect a face. Pooling enables the CNN to learn generalized information from images. Furthermore, pooling is essential for handling varying input sizes. For instance, it is used to convert features into the same size for the final classification layer, which always have to be equally sized.

2.3.3. Transfer Learning

Transfer learning is the process of exploiting already trained neural networks for other training processes (Goodfellow et al., 2016). Since training a neural network, especially for image processing, can be very time and energy consuming, one can benefit from already trained models. For transfer learning, it is assumed that factors explaining the variations of an already learned dataset *P1* are also relevant for the explanation of variations of a different dataset *P2*. Transfer learning is typically used for supervised learning with the same input data format. This is especially useful for CNN's. For example, if a CNN is trained on a data set of visual categories, such as cats and dogs, the neuron weights can be used to train a CNN for a different set of categories, such as lions and tigers. As described in *2.3.1 Convolution*, early convolutional layers learn basic shapes of objects. The model does not need to learn these basic shapes anymore, as the neuron weights can already extract this information from the new images. Especially when the first dataset *P1* is significantly larger, the model for the smaller new dataset *P2* can generalize the information much quicker. For transfer learning, models can either be used as feature extractors, or an existing model can be fine-tuned.

If an already trained CNN is used as a feature extractor, only the last layer of the CNN is changed with respect to the new dataset categories. All weights in all layers are kept and will not be readjusted for the new dataset. As a consequence, there will not be a new training process, which has the advantage of being a cost and time-efficient process. The assumption for that process is, that the trained model is generalized enough to deliver desirable results.

The second possible transfer learning approach is based on fine-tuning an already trained model. In this process all, or a subset of layers, are being readjusted in respect to the new data and the respective classes during a training process. A commonly applied approach is only fine-tuning the last layers of pre-trained neural networks, as the early layers already extract generalized basic shapes sufficiently (Patterson & Gibson, 2017).

2.4. R-CNN

In 2014, the R-CNN (Regions with CNN features) algorithm was published by researchers of the Berkeley University of California (Girshick et al., 2014). It is able to detect objects in images, by using features created from a CNN. It marks objects with bounding boxes, by a process called selective search, which tries to group pixels, by zooming at the image with different window sizes. It groups the pixels by color, intensity or texture for each window size and creates object proposals. The proposals are then passed through a CNN, to generate features. Subsequently, a Support Vector Machine (SVM) classifier is used to classify the object in certain classes. In the final step, the algorithm is a linear regression, which is used to improve the bounding boxes. It takes sub-regions of the image corresponding to the objects as input and outputs new and better fitted bounding box coordinates for the object in the sub-region. *Figure 3* visualizes the architecture of the algorithm.

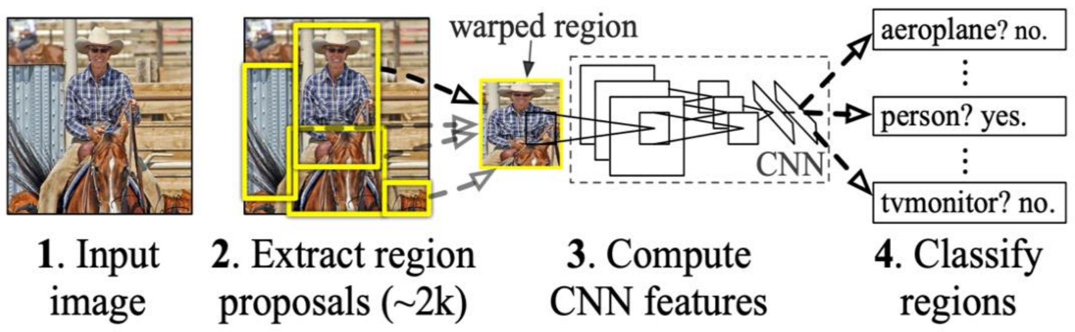

Figure 3: R-CNN

Illustration of the R-CNN architecture. (1): An image is the input of the R-CNN (2): Selective search is applied to extract region proposals (3): the region proposals are warped to the same size and inputted to the CNN to extract CNN features, then a regression is used to adjust and improve the bounding box coordinates (4): A SVM classifier is used to predict the object class (Girshick et al., 2014).

2.5. Fast R-CNN

In 2015 an improvement of the R-CNN was released, due to the suboptimal time efficiency of R-CNN (Girshick, 2015). R-CNN requires one pass through the CNN for every proposal region and it needs to train three different algorithms: the CNN, the SVM and the linear regression. To overcome these problems, Fast R-CNN was created. Region of Interest (ROI) Pooling is the core implementation of reducing the training time. With ROI pooling, each image is passed through the CNN only once and the created features are then shared and pooled for every proposal region. Furthermore, the classifier is integrated into the CNN and the final classification is no longer an extra algorithm like it was in R-CNN. *Figure 4* visualizes the change from an SVM classifier to the integration of the classification and regression layers in the CNN.

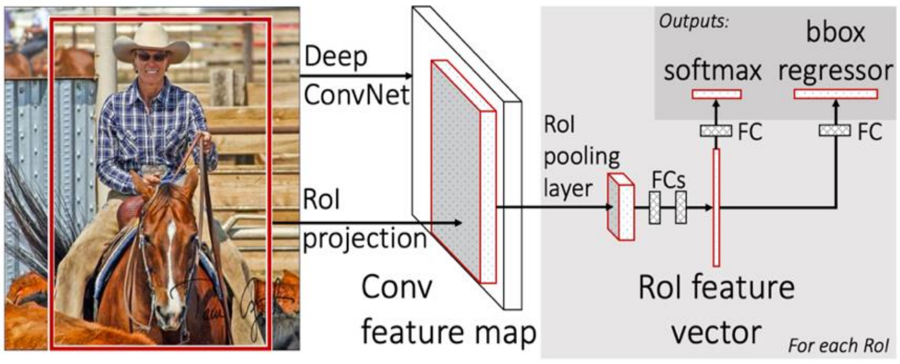

Figure 4: Fast R-CNN

Illustration of the Fast R-CNN architecture. The input image is passed into a CNN and the ROI's, generated by the selective search process, are projected to the CNN feature map, followed by a ROI pooling layer. This layer converts all ROI's into the same size. Subsequently, the ROIs's are classified using a classification CNN layer and the bounding boxes are adjusted using a regression layer (Girshick, 2015).

2.6. FASTER R-CNN

In 2016, the Fast R-CNN algorithm was further improved to Faster R-CNN (Ren et al., 2016). The selective search algorithm is replaced by a so-called Region Proposal Network (RPN), which is visualized in *Figure 5*. The RPN also uses the features created by the CNN to generate proposals. The RPN is a fully connected neural network which receives the output features of the CNN. A window slides through the feature map and the middle of this window is called an anchor point. For every anchor point, 9 proposal anchor boxes are created, based on the scales and aspect ratios. An aspect ratio is the width of an image divided by the height of an image. The scale defines the pixel length of the boxes.

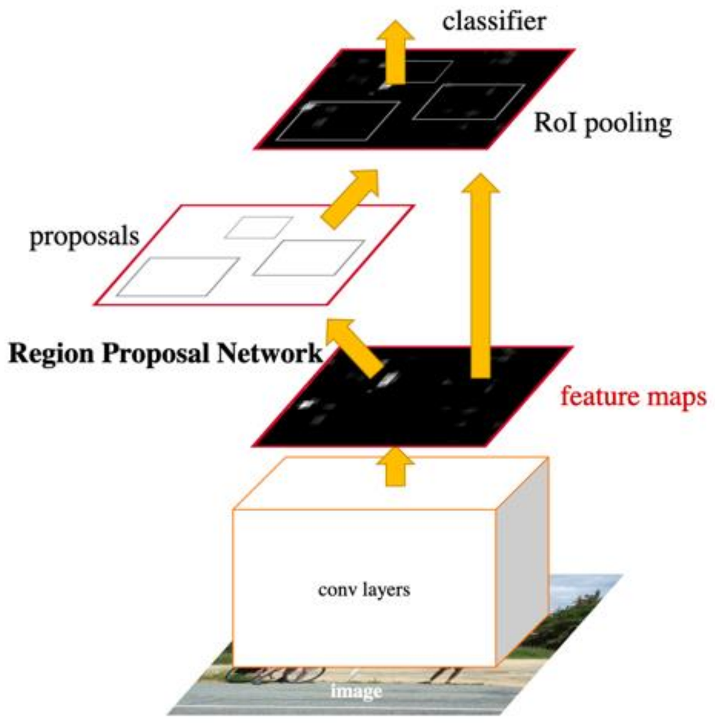

Figure 5: Faster R-CNN

Illustration of the Faster-RCNN architecture. The input image is passed into a convolutional neuronal network to generate a feature map. This process is followed by a RPN, which is responsible for ROI's by using anchor generation. The number of ROI's is then reduced with a process called non-maximum suppression. Subsequently, the ROI's are scaled into the same size, by ROI pooling and then classified with a classification layer and the bounding boxes are adjusted by using a regression layer (Ren et al., 2016).

For every anchor point, 9 proposal boxes are created, because 3 scales and 3 aspect ratios need to be chosen for the anchor generation. *Figure 6* visualizes the anchor box generation process. The RPN also has a classification layer, which learns to predict if there is an object in an anchor box. In addition, it also has a regression layer, which is trained to learn the offset of the anchor boxes for adjusting and fitting them to the object. For the adjustment process, the Intersection over Union (IoU) needs to be calculated (Ren et al., 2016).

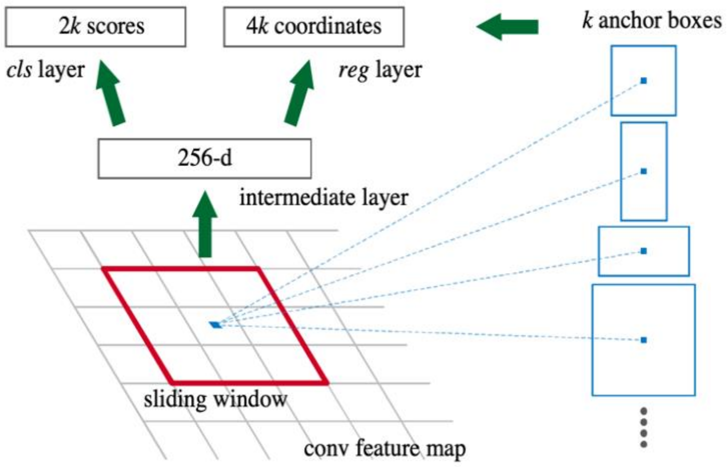

Figure 6: Anchor Generation

Illustration of the anchor generation process. For every anchor point, k anchor boxes are generated from the feature map. The number of anchor points is dependent on the number of aspect ratios (Ren et al., 2016).

2.6.1. Intersection over Union

The IoU is calculated by dividing the overlapping area of the ground truth box by the union area of both boxes (Ren et al., 2016). *Figure 7* visualizes the calculation. The offset of the anchor box and the ground truth box is learned by calculating the IoU of the anchor boxes and the ground truth boxes and then calculating the difference in the coordinates. These differences are the targets for the regression and the existence of an overlap of the ground truth box and the anchor box is the target for the classification. It does not predict the class of the object, instead, it only predicts if it is an object or background. The regressed offsets are applied to the anchor boxes to get more accurate positions.

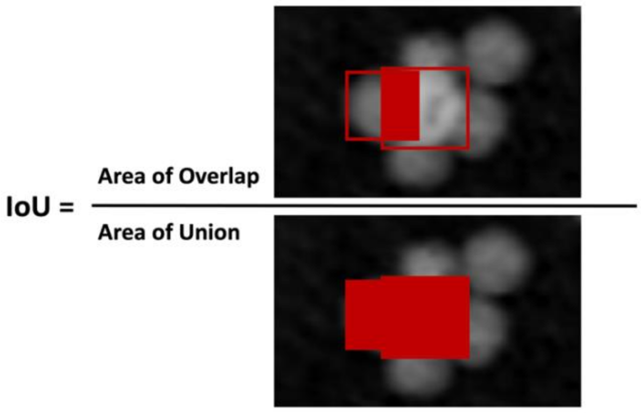

Figure 7: IoU

Illustration of the intersection over union. It is calculated by dividing the area of overlap by the area of union.

2.6.2. Non-Maximum Suppression

After the anchor box generation, Non-Maximum Suppression (NMS) is applied to reduce the number of potential boxes (Wang et al., 2020). NMS starts by sorting the candidate boxes by their confidence scores in a decreasing manner. The box which has the highest confidence score of containing an object is used to calculate the IoU for all remaining boxes. The candidate boxes which have a higher IoU value than the chosen threshold are removed. This process is then repeated with the remaining boxes until there is no box in the candidate box list left. NSM is a crucial parameter when objects are crowded or overlapping, as boxes result in a high degree of overlap. If a fixed NSM threshold is defined, some boxes might be removed even though they have a high confidence score for another object, as shown in *Figure 8*. Thus, the NSM threshold always needs to be adjusted specifically for each dataset (Schmidt et al., 2018).

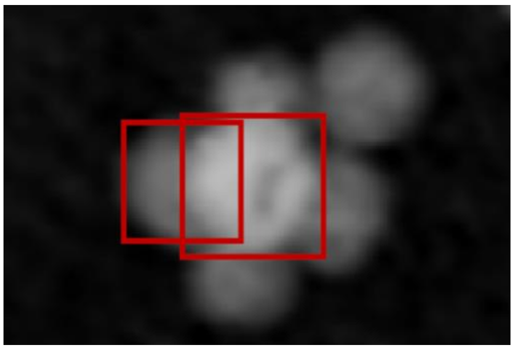

Figure 8: High IoU for overlapping cells

Illustration of a high IoU score for overlapping cells, which can complicate NMS.

.

2.6.3. Positive Fraction Ratio

After filtering the proposals, the remaining ones are sampled in a chosen positive fraction ratio (Ren et al., 2016). This is done to control the balance between background and foreground boxes, to avoid biasing the learning process towards the more prominent type. Usually, more background than foreground boxes exist, but this can vary a lot depending on the used dataset.

2.6.4. ROI Pooling

The remaining proposals after NMS are called Regions of Interest (ROI) (Ren et al., 2016). These ROI’s are then max pooled to convert all ROI’s to the same size for the final classification and bounding box regression layers of the Faster R-CNN. The ROI Pooling process takes the ROI part of the feature map and pools this part in a predefined number of grids. This is how all ROI's are converted into the same size. The output is then fed into the classification layer to predict the class and to the regression, layer to predict the coordinates of the bounding box.

2.7. Mask R-CNN

In 2017, the Faster R-CNN algorithm was extended by adding a segmentation process to it. The result is called Mask R-CNN (He et al., 2018). Besides the Regression and the Classification head layers of the model, another layer was added, the Mask Segmentation layer. This layer segments the image by considering the feature map. The Mask R-CNN architecture is visualized in *Figure 9*.

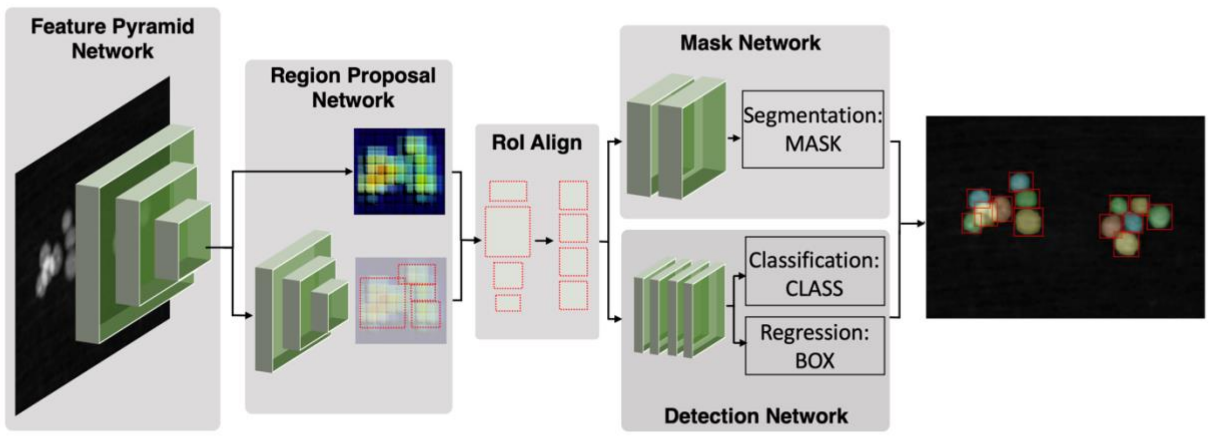

Figure 9: Mask R-CNN

Illustration of Mask R-CNN architecture. The input image is passed into the feature pyramid network to generate a feature map. The subsequent RPN generates regions of interest by applying anchor generation. The generated regions of interest are scaled into the same size using region of interest align, a more pixel-precise variant of region of interest pooling. The adjusted regions of interests are passed into classification and bounding box regression layers. In parallel, the regions of interest are also passed to a segmentation layer, which is a binary pixel-wise classification of the object inside a ROI (He et al., 2018).

It learns and outputs a binary matrix, in which the ones in the matrix represent pixels containing an object and the opposite for the zeros in the matrix. As the predicted mask is encoded in a spatial layout, it differs from the label or bounding box encodings, which are represented in short vectors. A $m \times m$ matrix is predicted for each ROI. The model pixel-wise predicts the binary information of being a pixel belonging to an object or being a pixel belonging to the background (He et al., 2018).

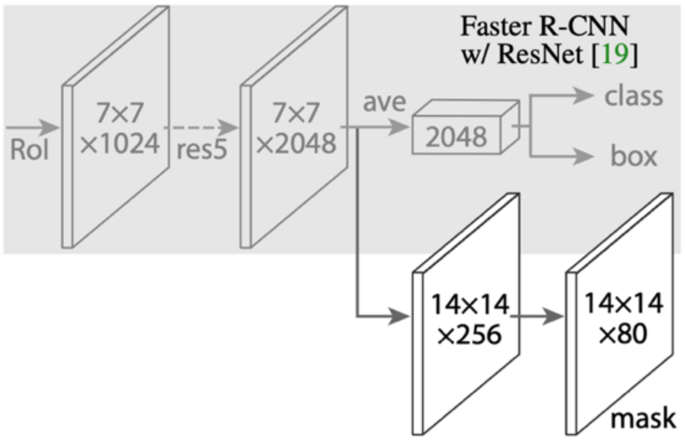

Figure 10: Mask Head

Illustration of the mask head. It is represented as $m \times m$ matrix. The classification and bounding box regression layers are represented as vectors (He et al., 2018).

2.7.1. ROI Align

For segmenting an object pixel-wise, ROI Pooling is not suitable anymore, as the output segmentation masks would be slightly misaligned. This is due to the fact that the chosen grid for ROI Pooling will not always symmetrically fit into the ROI and the pixel grid sizes are unequal. This is not a problem for the bounding box predictions of the Faster R-CNN, as they are not dependent on pixel-level specificity, unlike for the segmentation head of the Mask R-CNN. Due to the misalignment of ROI Pooling, ROI Align was implemented (He et al., 2018). If an image for example has the size 256 x 256 x 3(RGB) and a VGG16 backbone is used, the convolved feature map would be 16x16x256 (Simonyan & Zisserman, 2015). So, the scale factor is 16 (256/16=16). If a ROI has the size 150x200, the dimensions cannot be divided clearly by 16. The coordinates in the grid would be 9.375x12.5. The grids of the pooling do not fit perfectly in the feature map, without the grid overlaying multiple pixels, or without having unequal grid sizes. ROI Align solves that problem by creating equal size grids and using bilinear interpolation of all pixels inside a grid for the max-pooling operation. *Figure 11* shows the bilinear interpolation of pixels from an anchor which is not aligned to the feature map. This results in a possibility of pixel-perfect segmentation.

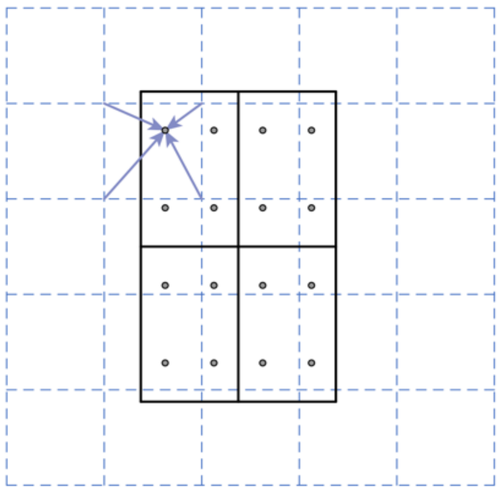

Figure 11: ROI Align

Illustration of ROI Alin process. Bilinear interpolation is used for region of interest align to create regions of interest with an equal number of pixels in each grid of the region of interest (He et al., 2018).

2.7.2. Loss Function

The loss function of the Mask R-CNN is a complex combination of several sub-loss functions which is performed on each ROI (He et al., 2018). The loss function can be observed in *equation (2.1).* The RPN loss is a combination of a RPN objectness loss and a RPN localization loss as defined in *equation (2.2)*.

$$Loss = Loss_{RPN} + Loss_{Box} + Loss_{class} + Loss_{Mask} \qquad (2.1)$$

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*) \qquad (2.2)$$

The input of the RPN is an image of any size and the output is a set of object proposals in a rectangular shape with a corresponding objectness score for each proposal. In this equation, i is the index of an anchor in the mini-batch. The predicted probability of an anchor i being an object is presented as p_i. The ground truth label p_i^* is 0 if the anchor is negative (does not contain an object) and 1 if it is positive (the anchor contains an object).

A vector of the coordinates of the predicted bounding box is represented as t_i and the corresponding ground truth as t_i^*. The classification loss L_{cls} is a binary cross-entropy loss and is calculated at the grid points which are either foreground or background in the ground truth labels. It is used to find the presence or absence of objects in a ROI. $L_{reg}(t_i, t_i^*)$ is the regression loss $R(t_i - t_i^*)$. R represents the smooth L_1 loss function. $p_i^* L_{reg}$ means that the regression loss is only calculated when a positive anchor $(p_i = 1)$ is present. This means the loss is only calculated when the anchor represents an object and is not calculated if the anchor represents background. The N_{cls} and N_{reg} represent a normalization process. Typically, the N_{cls} is normalized by the number of mini-batches of the SGD and the N_{reg} is normalized by the number of anchor locations. The term λ is an absolute balancing parameter for the L_1 smoothing, which is an absolute instead of a quadratic version of the weight decay described in *2.2.4 Weight Decay.*

For the regression of the bounding box, four coordinates are estimated, as presented in *equation (2.3).* The center coordinates are represented as x and y. The width and height are represented as w and h respectively. The letters x, y, w, h represent a value for a predicted box. If the letter is followed by a small a (e.g., x_a) it represents a value for the anchor box and if it is followed by a star-symbol (e.g., x^*) it represents a value of the ground truth. This can be described as a bounding box regression from an anchor box to a nearby ground truth box, to later calculate the offset between the anchor and the ground truth box to better align the bounding box (Ren et al., 2016).

$$\begin{aligned} t_x &= \frac{(x - x_a)}{w_a} & t_y &= \frac{(y - y_a)}{h_a} \\ t_w &= \log\left(\frac{w}{w_a}\right) & t_h &= \log\left(\frac{h}{h_a}\right) \\ t_x^* &= \frac{(x^* - x_a)}{w_a} & t_y^* &= \frac{(y^* - y_a)}{h_a} \\ t_w^* &= \log\left(\frac{w^*}{w_a}\right) & t_h^* &= \log\left(\frac{h^*}{h_a}\right) \end{aligned} \tag{2.3}$$

The Mask R-CNN follows the procedure of the Fast R-CNN approach for the $Loss_{Box}$ and $Loss_{class}$ which are trained and predicted in parallel and therefore are independent of each other. (He et al., 2018) Each ROI has a ground truth label class u and a bounding box regression target v. A multi-task loss is used on each labeled ROI to jointly train the bounding box regression and the classification.

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1] L_{loc}(t^u, v) \tag{2.4}$$

$$L_{cls}(p,u) = -\log p_u \tag{2.5}$$

The log loss for the true class u is represented by *equation (2.5)*. The discrete probability distribution for the categories is defined by the term p. Moreover, t^u stands for the bounding box regression of the RPN localization loss.

The L_{loc} is defined over a tuple of bounding box regression targets, which are true for class $u, v = (v_x, v_y, v_w, v_h)$ and a predicted tuple $t^u = (t_x^u, t_y^u, t_w^u, t_h^u)$, and the same for class u. The Iverson bracket indicator function $[u \geq 1]$ evaluates to 1 when $u \geq 1$ and 0 otherwise, which would mean it is labeled as background. The term λ is again the balancing parameter of the loss functions.

For the box regression loss ($Loss_{Box}$) a smooth L_1 loss function is used, as it was used for the RPN localization. The loss function is stated in *equation (2.6)* where the smooth L_1 loss is formulated like in *equation (2.7).* The term x corresponds to the difference of prediction and ground truth. (Girshick, 2015)

$$L_{loc}(t^u, v) = \sum_{i \in \{x,y,w,h\}} smooth_{L_1}(t_i^u - v_i) \tag{2.6}$$

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & if\ |x| < 1 \\ |x| - 0.5. & otherwise \end{cases} \tag{2.7}$$

Previous loss function components were proposed in reports and publications about the Fast R-CNN and Faster R-CNN and were adopted to the Mask R-CNN architecture, but the $Loss_{Mask}$ from *equation (2.1)* was added to the previous loss functions. The mask loss is also trained independently from the other loss equations and has a Km^2- dimensional output for every ROI, where $m \times m$ represent the size of the mask and K stands for the number of classes. In total K binary masks are generated. Then a pixel-wise sigmoid is applied, to classify if the pixel belongs to the background or an object. These values are then the input for the $Loss_{Mask}$, which is then the average binary cross-entropy loss. Like for the $Loss_{Box}$ only positive ROI's are used for the loss calculation. Consequently, this means that only ROI's which can be mapped with a ground truth object of the same class are considered for the $Loss_{Mask}$ calculation. This gives the neural network the advantage of being able to generate masks for all classes without competition among classes, as the classification process is handled in the classification branch of the neural network (Shu et al., 2020).

2.7.3. Optimizer

The standard optimizer for Fast R-CNN, Faster R-CNN and Mask R-CNN is the SGD optimizer. The creators of Mask R-CNN recommend SGD over other common optimizers like Adam for example, as mini-batches are required during training (He et al., 2018). The mini-batches are sampled hierarchically, first by sampling N images and then by sampling R/N ROI's from each image (Girshick, 2015). As described in *2.6.3 Positive Fraction Ratio,* one image contains positive and negative example anchors, foreground and background respectively. As there are normally a lot more background proposals it would bias the loss calculation towards the negative samples if the loss would be calculated over all anchors. So, instead, the loss is only calculated over the chosen number anchors per image (batch size per image), where positive and negative samples are distributed to the chosen positive sample fraction (Ren et al., 2016).

2.8. U-NET

In 2015, the same year when Fast-RCNN was developed, another neural network architecture was created, which aimed to solve the segmentation problem. This neural network is called U-Net and aims to segment images into binary masks, instead of individually detecting objects in images (Ronneberger et al., 2015). In other words, U-Net outputs a matrix of zeros and ones. The matrix has the same size as the input image and represents the presence of an object with a "one" and the absence of an object with a "zero". This is called semantic segmentation. It only outputs pixel specific if an object is present or not. The architecture differs a lot from the R-CNN like algorithms and is visualized in *Figure 12*.

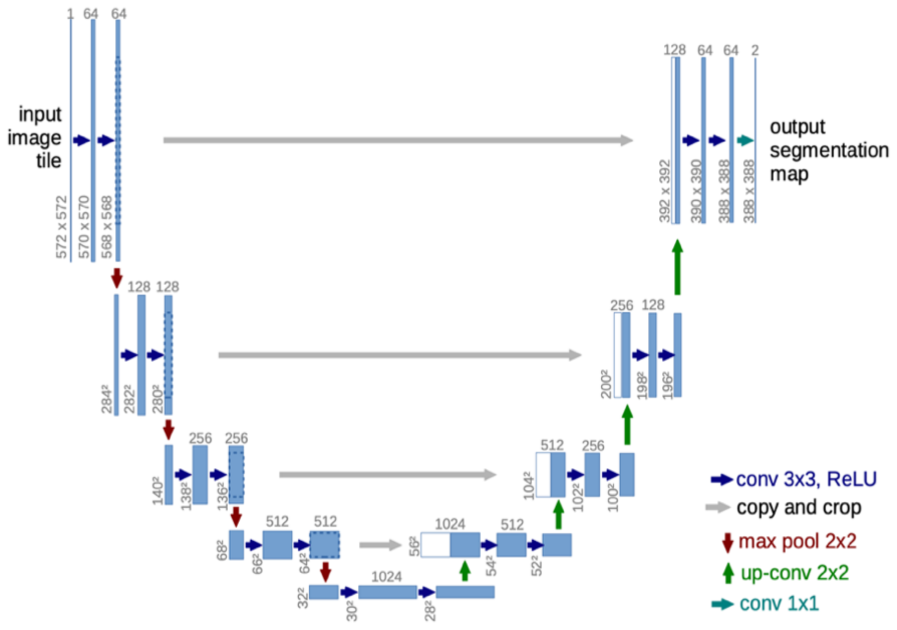

Figure 12: U-Net

Illustration of the U-Net architecture. The input image is passed through convolutional contracting layers for feature generation and expensing layers for increasing the resolution of the output. The output is a binary matrix of pixel values belonging to an object (Ronneberger et al., 2015).

The network consists of a contracting and expensing path, also called encoder and decoder respectively. The encoder path is used to learn features from the image. While performing typical convolutional and pooling operations the image is being downsampled. Then the decoder is used to expense the feature map to increase the resolution of the output, by exchanging the pooling operator with upsampling operators. As the U-Net architecture only outputs binary masks for all objects, the network has huge drawbacks when objects are very crowded. Semantic segmentation is not performant when instance segmentation is required. To solve this problem the U-Net architecture was expanded. Besides that, an extra output channel was added to predict borders of crowded objects, which is used to separate touching objects in the binary masks (Iglovikov et al., 2018). Furthermore, U-Net can perform multi-class classification, by giving each class a different pixel value (Novikov et al., 2018).

3. LITERATURE REVIEW

In the biomedical field, cell segmentation was traditionally conducted by using thresholding methods and watershed (Caicedo et al., 2018). For these methods, pixel value thresholds need to be found to separate cells from the background. These methods are widely used, as they are simple and computationally efficient, but this simplicity has drawbacks.

Generally, the amount of segmentation errors is non-trivial, especially with complex cell structures. Segmentation errors bias downstream analyses, yielding unreliable measures or systemic noise that is difficult to quantify and factor out. The causes of segmentation errors can vary. Classical algorithms have limitations and assumptions that do not always hold. For example, thresholding methods assume bimodal intensity distributions of pixel-values for cells or background. Moreover, these methods assume clearly separable boundaries between individual cells. These limitations were suitable for image analysis in the early stages of bioimaging, but they do not hold for many bioimaging processes nowadays. The amount and complexity of generated data have tremendously increased over the last decades (Caicedo et al., 2018). Regarding this, new and more reliable methods were approached.

Object detection and object segmentation are tasks, which were able to be solved with Deep Learning algorithms for several years and the algorithms constantly improved, as R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN and U-Net and its variations evolved. Even though U-Net was especially designed for biomedical image segmentation and Mask R-CNN was mainly designed to detect day-to-day objects, Mask R-CNN achieves good and sometimes better performances in biomedical images. The Data Science Bowl 2018 revealed this, as the challenge was to detect nuclei of cells in a variety of biomedical images. Many top scoring solutions were based on a Mask R-CNN architecture or were at least following the ranking strategy of the Mask R-CNN to select the best segmentation masks from several candidates, predicted by the base models. The data set contained 37,333 manually annotated nuclei in 841 2D images from more than 30 different experiments with different samples, cell lines, imaging conditions and microscopy instruments, research facilities and staining protocols. Hence, all cell cultures were treated with staining agents, which were used to make the cell nuclei fluoresce (Caicedo et al., 2019). In a direct comparison, Mask-RCNN seems to better detect individual nuclei from a cluster while U-Net has a tendency to clump them into one big nucleus. Moreover, the amount of misclassifications of the Mask R-CNN is lower than for the U-Net model (Vuola et al., 2019).

In addition to that, Mask R-CNN was proven to be a performant Deep Learning approach to detect nuclei in stain and marker-free phase time images (Yuan et al., 2019).

4. DATA

The data, analyzed in this thesis, are images provided by the Biomedical Technology Center of the Medical Faculty of the University of Muenster, Germany. More specifically, the images are QPI images generated by DHM.

4.1. DATA GENERATION

An inverted Nikon Ts2R microscope equipped with an automated microscope stage and an attached DHM module was applied for quantitative phase contrast imaging of living RAW 264.7 macrophages. (Lenz et al., 2016). The RAW 264.7 mouse macrophage cell line was obtained from ATCC (American Type Culture Collection, Manassas, Virginia) and subcultured routinely twice a week in standard cell culture medium supplemented with FCS (Baczewska et al., 2020). The DHM imaging experiments were performed on subconfluent cells on µ-well plates with glass lids (ibidi GmbH, Munich, Germany). An incubator chamber (ibidi GmbH, Munich, Germany) allowed the investigation of living cells in a 5% CO_2 atmosphere at a temperature of 37°C. The coherent light source for the recording of digital holograms was a fiber coupled solid-state laser (Cobolt 06-DPL, λ=532 nm, Cobolt AB, Solna, Sweden). Digital off-axis holograms of the specimens were recorded with a complementary metal-oxide-semiconductor (CMOS) sensor using a 20x microscope lens (Nikon Plan 20x/0.4, Nikon, Japan). The reconstruction of QPI images from digitally captured holograms was performed numerically (Min et al., 2017).

4.2. DATA DESCRIPTION

The images were provided as graylevel images (8-bit) and contain two different phenotypes of RAW 264.7 mouse macrophages. The phenotypes can be classified as “normal macrophage cells” (N cells) and “multi-shaped macrophage cells” (M cells). *Figure 13* visualizes close-up examples of these phenotypes. In contrast to N cells, which have a stable round shape, the shape of M cells can vary significantly*. Figure 14* shows an image of both N and M cells. In total 1,420 images were provided. The images were primarily generated for biomedical experiments, toxicological studies and cellular growth experiments. In these experiments, a set of cells was treated with chemical substances and observed for around 24 hours, in which the observed set of cells was imaged every 30 minutes. This process of observing one set of cells can be called one sequence.

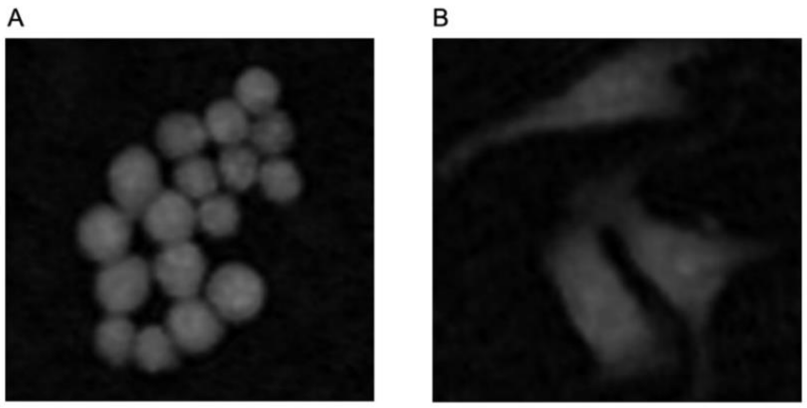

Figure 13: Macrophage cell phenotypes

*(**A**): Illustration of 17 normal macrophage cells (**B**): Illustration of three multi-shaped macrophage cells.*

Depending on the treatment the cells received, the number of cells is increasing, decreasing, or the ratio between the phenotypes of the cells are changing. This fact makes it important to analyze images within one sequence over time. The 1,420 images are subdivided into 29 sequences. In *Figure 15* a sequence of an increasing number of macrophages (upper sequence) and a sequence of a decreasing number of macrophages (lower sequence) in the microscopy images can be observed. The first, the 25th and the 48th image of the respective sequence that were recorded as distinct time points (0h, 12h, 24h) of the according experiment, are presented as an example.

In total 97 images were manually labeled by experts. Each cell was segmented and classified into one of the mentioned phenotypes. Around three to four images of each sequence were sampled and then labeled. To have a diverse set of images for each sequence, images from the start, from the middle and the end of a sequence were considered in the sampling process.

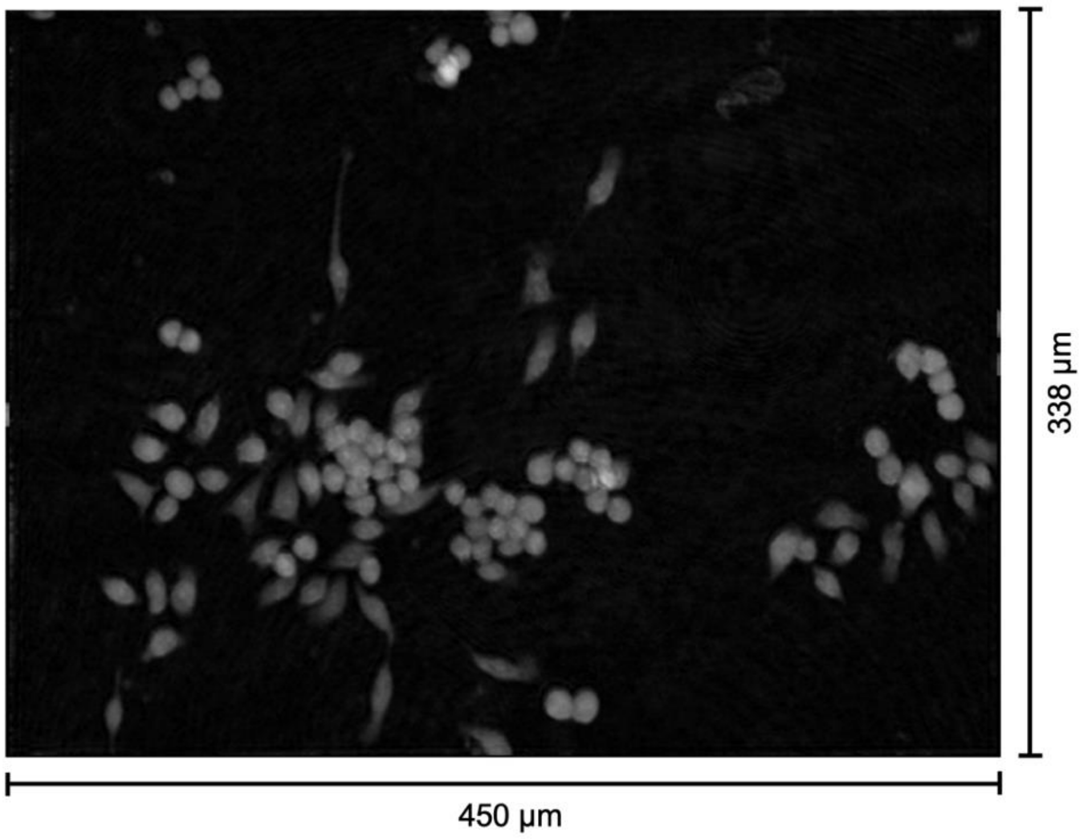

Figure 14: Example quantitative phase image of macrophage cells

Illustration of an example quantitative phase image, generated by digital holographic microscopy, containing both normal and multi-shaped macrophage cells.

The number of cells of the two different phenotypes is imbalanced. In total, the labeled images contain 7,600 cells, of which 6,451 are N Cells, with an average of 66.5 N cells per image. Only 1,149 are M cells, with an average of 11.9 M Cells per image. The histogram in *Figure 16* illustrates the unequal distribution for the phenotypes. Images can contain more than 250 N cells. In contrast, there are only a few images which contain more than 50 M cells.

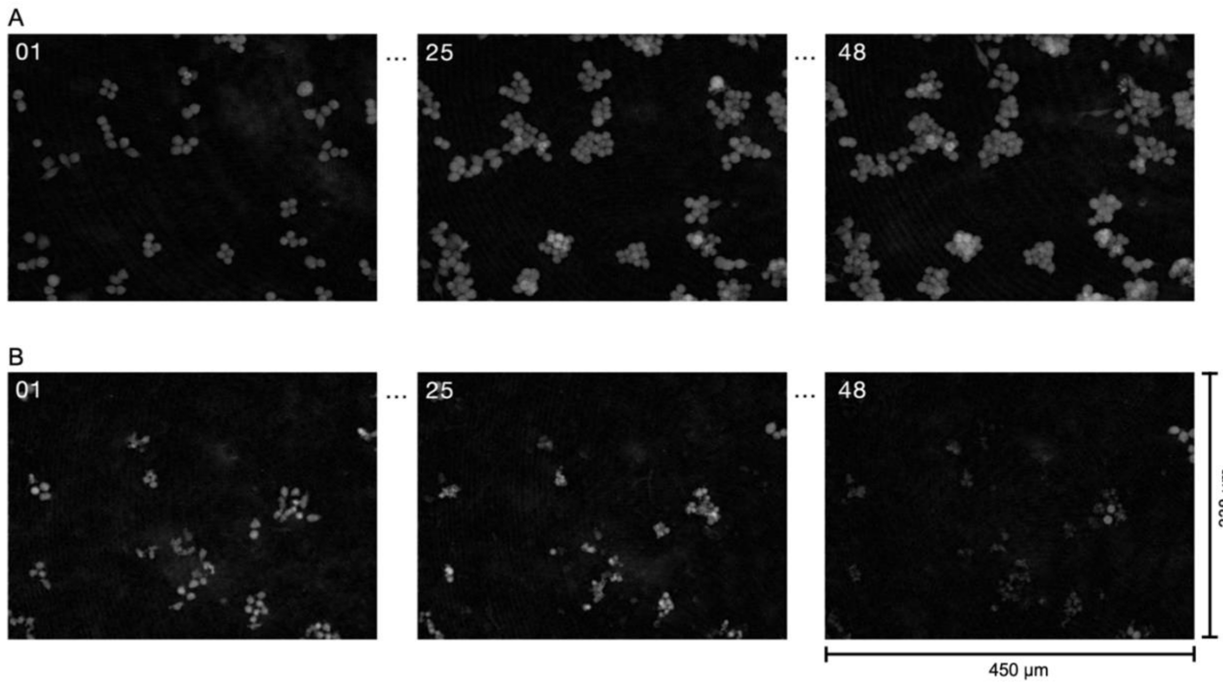

Figure 15: Example sequences

*(**A**): Illustration of three exemplary images from a sequence of a toxicology experiment, where the number of both normal and multi-shaped macrophage cells are increasing over time. (**B**): Illustration of three sample images from a sequence of a toxicology experiment, where the number of both normal and multi-shaped macrophage cells are decreasing over time.*

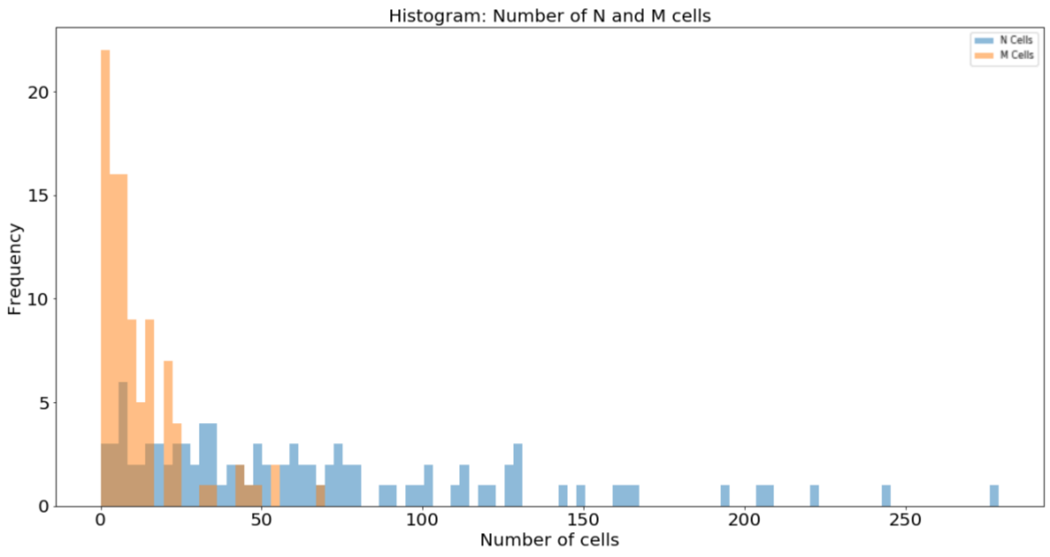

Figure 16: Histogram of the number of N and M cells

Illustration of the number of N and M cells in the analyzed images, represented in a histogram.

The train and test sets were divided by taking the sequences into consideration. Each set consists of different sequences. This was conducted to limit the potential bias, as images from the same sequence might be similar. The train and validation sets were sampled out of the same 15 sequences with 55 images. For the validation set, 6 images (around 10%) were sampled. For the test set, 49 images were sampled from 14 different sequences. This was done to limit the potential bias when testing the algorithm with the test set. It secures that the algorithm never sees images from a sequence of the test images for training the algorithm, as images within one sequence can be similar. Hence, the train set consists of 49 images and the validation set of 6 images, whereas the test set consists of 42 images. *Table 1* gives an overview of the datasets and the distributions of M and N Cells.

Table 1: Dataset overview

Dataset	**Number of images**	**Number of cells**	**Number of N Cells**	**Number of M Cells**
Train	49	4483	3855	628
Validation	6	370	264	106
Test	42	2747	2332	415
Total	97	7600	6451	1149

5. METHODOLOGY

As described in *4. Data*, 97 labeled images are available to answer the proposed research questions. This chapter focuses on the presentation of the method used to analyze the given data in order to answer the research questions. The proposed algorithm with its specific parameter settings and training configurations is presented. The decisions made for configuring the algorithm and the training process are explained in detail. Moreover, to quantify the performance of the proposed method, the evaluation metrics are described. The complete overview of the chosen parameter settings can be observed in the Appendix.

5.1. MODEL

In order to reach the goal of this thesis, a Mask R-CNN architecture was chosen, as the literature review revealed that this is an appropriate Deep Learning architecture to answer the research questions. The main reason for that is the fact that the Mask R-CNN is able to solve the instance segmentation directly, which means it allows the segmentation and classification of individual objects in images in a trainable end-to-end manner. No further post-processing tasks of the prediction results are needed. Additionally, it was proven to deliver robust results in a biomedical context and to be a performant Deep Learning approach for detecting nuclei in stain and marker-free phase time images, as described in *3. Literature Review*. Furthermore, it was proven to be able to detect and segment objects in a performant way, even in densely crowded areas e.g., when cells are clumping together. The use of the Mask R-CNN architecture for detecting, segmenting, classifying and quantifying cells in QPI images, generated by DHM, is a logical conclusion.

5.1.1. Feature Pyramid Network

Numerous CNN architectures can be used as a feature extractor for the subsequent object detection and segmentation steps. These networks are called Feature Pyramid Networks (FPN) Two different Deep Residual Learning architectures, also called ResNet architectures, are tested to evaluate their performance (He et al., 2015). The ResNet50 is a CNN, which consists of 50 layers and 25.6 million trainable parameters. The ResNet101 consists of 101 layers and 44.5 million trainable parameters, increasing the complexity and training time (Zagoruyko & Komodakis, 2017). The ResNet50 and ResNet101 were pre-trained on the Common Objects In Context (COCO) dataset (Lin et al., 2015).

5.1.2. Anchor Generation

As the Mask R-CNN is trained to detect macrophages, which are relatively small, the anchor generation sizes are reduced. The default anchor sizes are 32, 64, 128, 256 and 512. For the given macrophage dataset, the anchor sizes are set to 16, 32 and 64 pixels. The corresponding aspect ratios are set to 0.5, 1 and 2. Consequently, 9 proposal anchor boxes are created in total for every anchor point.

5.1.3. Region Proposal Network

The RPN has numerous parameter options, which have a great impact on the learning process of the algorithm. The IoU for being background is set to 0.3 and the IoU for foreground is set to 0.7. Meaning, that all anchors are matched to the ground truth. If the threshold is higher than the defined threshold for being background or foreground, the anchor is assigned to the corresponding label.

To reduce computational complexity, the number of generated anchor proposals is limited before applying NSM. Only the 12,000 top scoring proposals are kept during training and 6,000 during testing, which is the standard setting of the implementation. Subsequently, NMS is applied and the NMS threshold is set to 0.4. After filtering the proposals with NSM, only 4,000 top scoring anchor boxes are kept during training and 2,000 during testing. These numbers are increased from the default values of 2,000 and 1,000 for training and testing respectively, due to the high number of macrophage cells in the images.

Furthermore, the loss of the RPN was weighted 1.5 times higher than the other parts of the complete loss function. This is done because the main challenge of the algorithm is to localize and detect all cells even in densely crowded areas, where cells can also heavily overlap.

5.1.4. ROI

The IoU for the remaining ROI boxes is set to the default value of 0.5, meaning that if an ROI has an IoU greater than 0.5 the pixels inside of that ROI are considered as a foreground object and below 0.5 as background. The batch size per image is set to 512, which means that 512 ROI's are sampled randomly from one image for the training process. The sampling process is dependent on the positive fraction ratio, which is set to 0.25. Meaning, that 25% of the sampled anchors contain foreground objects. For the testing process, an additional NMS is added to filter the predicted boxes. The NSM test threshold is set to 0.4. Moreover, a confidence score filter is set to 0.6, allowing only boxes with a probability of containing an object higher than 60% will be passed on into the classification and box regression layer of the network.

5.1.5. Solver

SGD was chosen as the optimizer algorithm, due to the reasons described in *2.7.3 Optimizer.* A momentum of 0.9 is the default value used by the authors of Faster R-CNN and Mask R-CNN (Ren et al., 2016). Hence, it is also set to 0.9. Moreover, weight decay is also used and set to the Mask R-CNN author's default value of 0.0001 (He et al., 2018).

5.2. TRAINING

Besides the algorithmic related parameter settings described in *5.1 Model,* several options for the training process need to be chosen in order to generate performant and comparable results. Training related parameters do not infer with the Deep Learning architecture itself. These parameters are responsible for the learning process of the algorithm.

5.2.1. Epochs

The used Mask R-CNN implementation is iteration based, which implies that the epochs are depended on the batch size and number of iterations. One epoch is equal to the number of images divided by the number of batches. Each training process was trained for 600 epochs, with a batch size of two, to generate comparable results. As the training set consists of 55 images and the validation set is sampled by 10% from the training set, 49 images are used for training and 6 images for validation. With a batch size of 2, 14.700 iterations are needed to train the network for 600 epochs.

5.2.2. Learning Rate

Multiple learning rates are tested to evaluate the performance with the same number of epochs during training. In particular 0.01, 0.001 and 0.00025 are tested as learning rates. The chosen learning rates are relatively small, as the model is used with transfer learning and the object structures are rather simple than complex.

The learning rate has 1.000 linear warm-up iterations with a warm-up factor of 1/1000, implying that the initial learning rate is decreased in the beginning and linearly increasing in every step of the first 1.000 warm-up iterations until it reaches the initial learning rate (Li et al., 2020). This is conducted as SDG with momentum is used to prevent remembering a suboptimal step at the beginning of the learning process, which would have a high impact, if the learning rate would be high already at the very beginning of the training process (Loshchilov & Hutter, 2017).

5.2.3. Data Augmentation

Since the dataset is not very large, data augmentation is used to expand the variance in the dataset artificially. *Table 2* lists the used augmentation processes and their probabilities of applying them during training. Random brightness, random saturation, random contrast are applied every time an image is sampled for a training iteration, whereby each augmentation process is applied with a different value. The relative values represent the intensity of applying them. A random brightness of 0.8 means that the brightness is reduced by 20% and a value of 1.8 means that the brightness is increased by 80% of the original brightness, whereas a value of 1 means the brightness is not changed. Random rotation of either zero or 90° is applied at every iteration step. Random horizontal and vertical flip is applied with a 40% chance. The images are randomly cropped at every iteration step to 40% or 60% of the original image. After applying augmentation, all images are rescaled to the same size, namely to 1333x800.

Table 2: Augmentations

Augmentation	**Value**	**Probability**
Random Brightness	0.8 – 1.8	100%
Random Saturation	0.6 – 1.4	100%
Random Contrast	0.7 – 1.3	100%
Random Rotation	0°, 90°	100%
Random Flip	Horizontal	40%
Random Flip	Vertical	40%
Random Crop	0.4, 0.6	100%

5.3. Evaluation Metric

Object detection and segmentation problems can be evaluated with numerous evaluation metrics. Often it is not possible to only observe one metric during the whole detection and segmentation process, as different metrics focus on different aspects of the performance. Besides classifying objects into the correct category, the alignments of the bounding boxes or the segmentation masks are also very important aspects of the performance.

5.3.1. Detection Accuracy

The accuracy metric for standard classification problems is defined in *equation (5.1).* It measures the ratio of correct predictions over the total number of instances evaluated. Thus, it can be interpreted as the percentage of correctly predicted objects (Hossin & M.N, 2015).

$$Accuracy = \frac{True\ Positives + True\ Negatives}{True\ Positives + False\ Positives + True\ Negatives + False\ Negatives} \tag{5.1}$$

A $True\ Positive$ (TP) is a correct detection which contains an object, which is the case if the IoU of the predicted box and the ground truth box of an object is above the defined IoU threshold. A $True\ Negative$ (TN) is a correct detection which does not contain an object, meaning the IoU threshold is below the defined threshold or it fully contains background. A $False\ Positive$ (FP) is a detection where the model predicts an object, which does not contain an object. A $False\ Negative$ (FN) is a not detected object (Caicedo et al., 2018). The accuracy metric does not take the segmentation performance into account and can be suboptimal when classes are highly imbalanced but both categories have the same importance.

5.3.2. Mean Absolute Error & Mean Absolute Percentage Error

The Mean Absolute Error (MAE) and the Mean Absolute Percentage Error (MAPE) are two very trivial performance metrics, which are normally used for regression or time-series performance evaluation. The MAE, presented in *equation (5.2)*, is the absolute difference between the predicted number of cells and the actual number of cells within one image (Willmott & Matsuura, 2005).

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |Actual\ number\ of\ cells - Predicted\ number\ of\ cells| \tag{5.2}$$

$$MAPE = \frac{1}{N}\sum_{i=1}^{N} \frac{|Actual\ number\ of\ cells - Predicted\ number\ of\ cells|}{Actual\ number\ of\ cells} \tag{5.3}$$

The MAPE *(5.3)* is very similar to the MAE and is just the percentual difference between the actual number of cells and the predicted number of cells (De Myttenaere et al., 2016). These metrics cannot present any information about the alignment of the bounding box or the segmentation mask. Moreover, they do not consider false predictions. So, these metrics should be considered with caution and scepsis, but they can still deliver some additional information when it is important to consider the number of missed detections.

5.3.3. Pascal VOC Mean Average Precision

The Pascal VOC Mean Average Precision (mAP) is a more reliable and meaningful performance metric for object detection and segmentation (Everingham et al., 2010). It was developed for the Pascal Visual Object Classes Challenge (VOC) 2012 challenge. It is the mean of the interpolated areas of the Precision-Recall-Curves for each category. The Precision is also called positive predicted value and it measures the ratio of true-positive predictions and total positive predictions and can be observed in *equation (5.4)*. Specifically, the precision indicates the percentage of correctly predicted cells within all predicted cells.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positve} \tag{5.4}$$

The Recall is also called sensitivity and measures the ratio of true-positive predictions and all correct predictions. So, it emphasizes the percentage of correctly predicted cells within all correct predictions (Hossin & M.N, 2015).

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{5.5}$$

A high recall value with a corresponding low precision value means that all ground truth objects have been detected, but a lot of predicted detections are incorrect. In other words, the model predicts many FP’s. In contrast, a low recall value with a corresponding high precision value means that all predicted objects are correct, but a lot of ground truth objects have not been detected. This indicates that the model suffers from a lot of FN predictions. An important note is that for the Pascal VOC implementation of the mAP the IoU is always set to 0.5 to classify predictions into FP, FN, TP or FP (Everingham et al., 2010).

For the mAP, the average precision (AP) needs to be calculated. As mentioned earlier, the interpolated area under the curve (AUC) of the precision-recall curve. This curve can be visualized by plotting the recall on the x-axis and the precision on the y-axis. Further, the curve represents the recall value for each precision value and wise versa. The precision and recall are used, as they do not take TN into account and are therefore less biased to imbalanced datasets (Saito & Rehmsmeier, 2015). An example of a precision-recall curve can be observed in *Figure 17(A).*

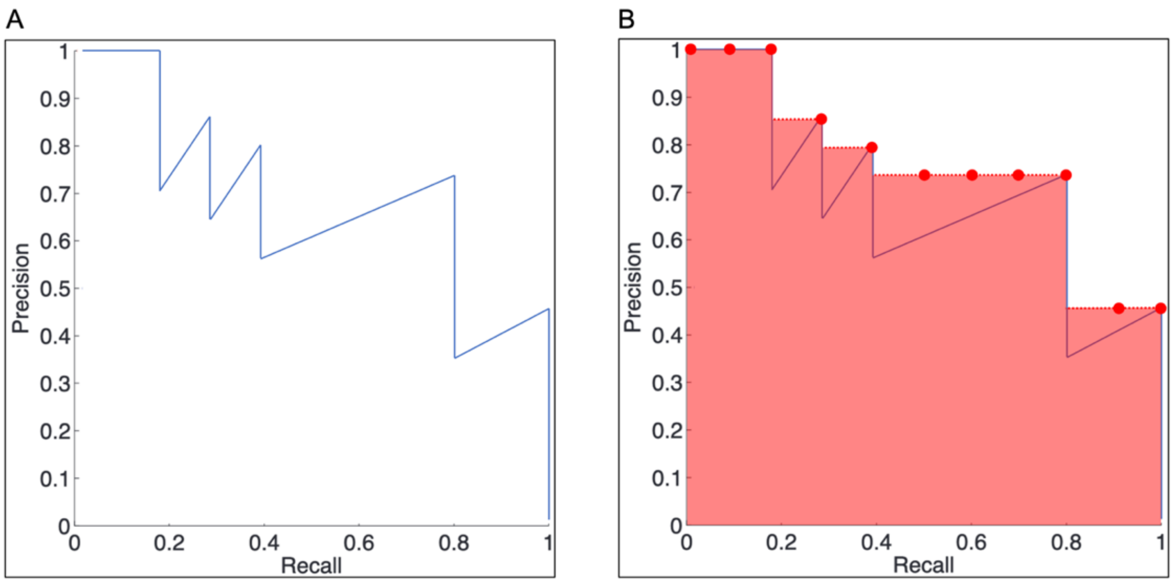

Figure 17: Mean average precision

*(**A**): Illustration of a precision-recall curve (**B**): Illustration of an interpolated area under the precision-recall curve (red).*

For the AP calculation *(5.6)*, the precision is interpolated at each recall level r by taking the maximum precision measured for which the corresponding recall exceeds r. Where $p(\tilde{r})$ is the measured precision at recall $\tilde{r}$. (Everingham et al., 2010) This is done for 11 recall points (0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0), which are represented as the red dots in *Figure 17(B)*. The red area represents the AUC of this interpolated precision-recall curve.

$$AP = \frac{1}{11} \sum_{r \epsilon \{0,0.1,0.2,\ldots,1\}} p_{interpolated}(r) \quad (5.6)$$

$$p_{interpolated}(r) = \max_{\tilde{r} > r} p(\tilde{r}) \quad (5.7)$$

The intention of interpolating the precision-recall curve is to reduce the variations in the precision-recall curve. To obtain a high score, a model should have high precision at all levels of recall. The AP is calculated for each class and then the mean is calculated for every class AP. The resulting measure is the mAP. This procedure secures that every class is weighted equally, even when the classes are imbalanced (Everingham et al., 2015) (Everingham et al., 2010). In this work, the mAP is calculated for the bounding box and the segmentation mask individually. This means, that the fixed IoU threshold of 0.5 is used for the TP, TN, FP, FN calculation (which is needed for the mAP calculation) on bounding box overlapping and segmentation mask overlapping separately and therefore can result in different mAP values.

6. TOOLS

For creating the Mask R-CNN and to analyze the provided data, a various set of hard- and software was used. This chapter aims to present the utilized tools.

6.1. SOFTWARE

For the entire project the programming language Python 3.6 was used. The Detectron2 framework was utilized for creating the Mask R-CNN (Wu et al., 2019). The framework provides pre-build functions like data loaders and pre-implemented FPN networks. The Mask R-CNN models can be configurated and customized within the framework. Detectron2 is built with PyTorch, which is a framework for creating artificial neural networks with Python (Paszke et al., 2019).

For creating the ground truth labeled dataset, the software labelme was utilized (Wada, 2016). With labelme, labeled images can be saved as json files in which coordinates of the annotated objects are stored.

6.2. HARDWARE

The models were trained on Google Collaboratory, which is a platform providing hosted Jupyter Notebooks with GPU access (Bisong, 2019). Python code can be written and executed with these notebooks. The Mask R-CNN models were trained using an Nvidia Tesla K80 GPU.

7. RESULTS

This chapter provides an overview of the performance of the evaluated models with different backbone and learning rate settings. In addition, a detailed analysis of the best performing model is conducted.

7.1. MODEL COMPARISON

In total, six different configurations are tested and evaluated. The results are presented in *Table 3*. With all parameters fixed, as described in *5. Methodology,* it can be observed, that the best performing model was trained with a ResNet101 backbone and a learning rate of 0.001. The mAP@50 for the bounding box is 82.69 and for the segmentation it is 82.83 for the test set. For both ResNet50 and ResNet101, the best performing learning rate is 0.001, where the model using the ResNet101 outperforms the model using ResNet50.

Table 3: Model performance for parameter settings

Backbone FPN Network	**Learning Rate**	**Box mAP@50**	**Seg mAP@50**
ResNet50	0.01	80.08	80.15
	0.001	80.27	80.33
	0.00025	78.08	78.69
ResNet101	0.01	78.25	78.99
	0.001	**82.69**	**82.83**
	0.00025	80.83	80.85

7.2. MODEL TRAINING

When analyzing the best performing model, it can be observed that the loss function of both training and validation set converges relatively fast. As visualized in *Figure 18* the training total loss is steadily and slowly decreasing after a fast-decreasing process at the beginning of the training. The validation loss has similar behavior, but it heavily saturates after around 12,000 iterations and does not decrease significantly anymore. When observing the validation loss function, it can be seen that the function is not smoothly decreasing. This is due to the fact that the validation loss is only calculated every 500 iteration steps, due to computational reasons.

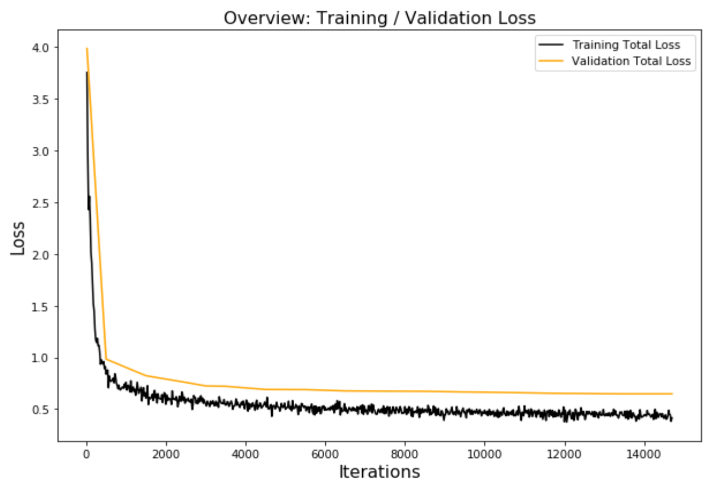

Figure 18: Overview training and validation loss

Illustration of the performance of the training and the validation loss function. The validation loss function is evaluated every 500th iteration.

As the total loss function is a combination of several loss functions, as described in *2.7.2 Loss Function*, it is important to validate the individual loss function to ensure that the training process is not biased towards one of the loss functions. As it can be observed in *Figure 19* and *Figure 20,* all loss functions are converging, even though some loss functions converge faster than others. The Mask loss saturates a lot faster than the PRN Location loss, which is steadily decreasing for example. Despite the fact that there is some inequality in the converging process, in general, all loss functions behave as expected and demonstrate that the learning process is balanced towards all loss functions.

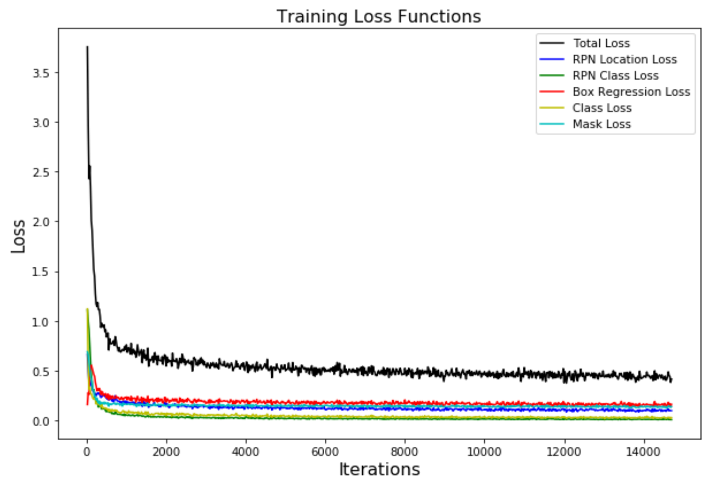

Figure 19: Total training loss functions and underlying sub-loss functions

Illustration of the performance of the total training loss function and the respective underlying sub-loss functions.

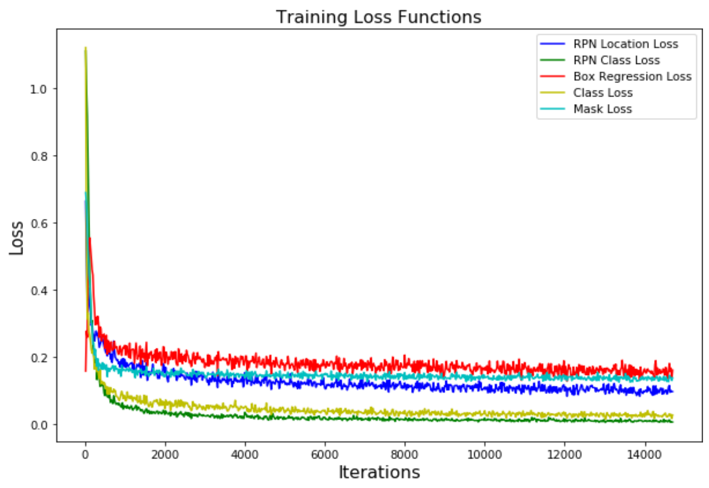

Figure 20: Training loss functions

Illustration of the performance of the sub-loss functions of the total training loss, for a better overview of the individual performances.

Besides validating the training performance of the loss function, the accuracy of the classification layer of the Mask R-CNN can be observed. The accuracy value is steadily increasing, starting from 0.7 to around 0.93. In detail, the first steps show the greatest gains in accuracy, as *Figure 21* shows. After approximately 500 iterations the accuracy increases very slowly with small spikes.

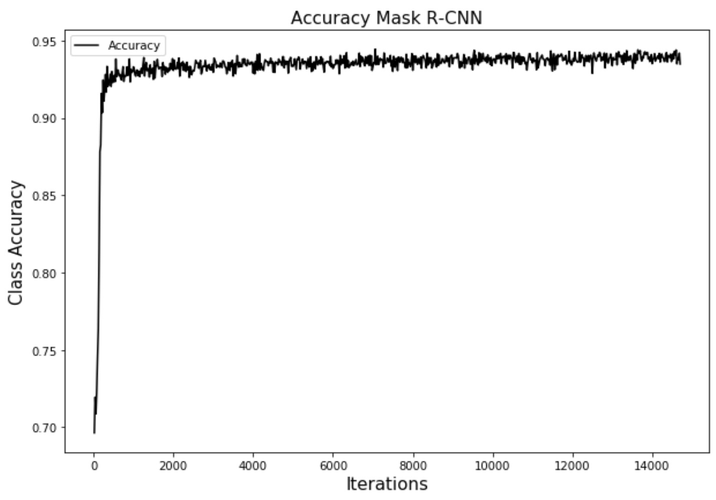

Figure 21: Accuracy Mask R-CNN

Illustration of the performance of the accuracy for the train set while training the Mask R-CNN model.

The False-Positive-Error behaves similar to the accuracy. In the first few hundred steps, the error is strongly decreasing and shows the biggest reductions of this error metric, which can be observed in *Figure 22*. After the strong plumbing, the error is saturating below the value of 0.1 and shows spikes in the saturating process again.

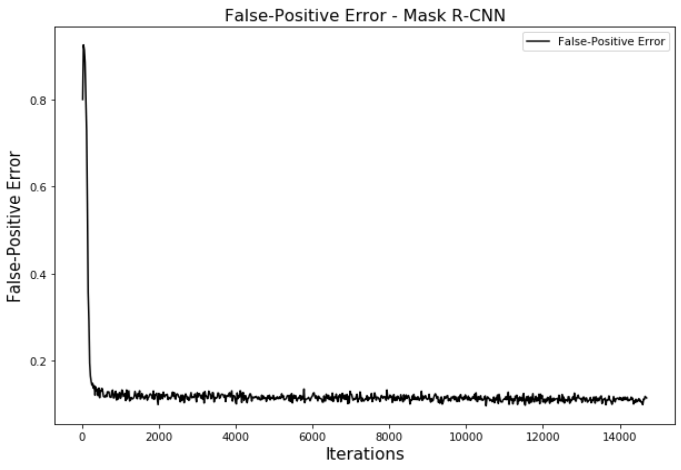

Figure 22: False-Positive error

Illustration of the performance of the False-Positive Error during the training process.

The False-Negative error behaves differently. The error metric starts already with a relatively small value of around 0.065 and then fluctuates around values of 0.055 to 0.035. The error slowly decreases again to a final value of around 0.045. As *Figure 23* shows, the spikes are present as well. Spikes are relatively small, even though they seem heavier than for the False-Positive Error, because of a smaller range on the x-axis.

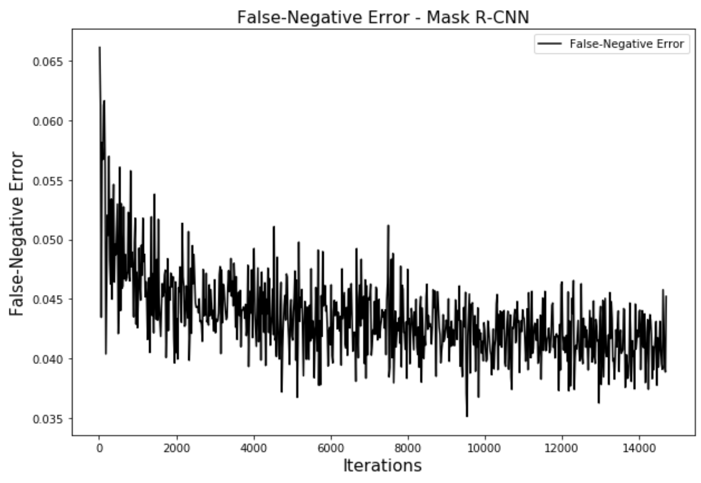

Figure 23: False-Negative error

Illustration of the performance of the False-Negative Error during the training process.

Another important measure tracked during the training is the mAP@50. Every 500th iteration the mAP is calculated for the bounding box and the segmentation of the validation set. Both, bounding box and segmentation mAP, strongly increase in the first 2,000 steps to around 74, then drop slightly to approximate 70 and finally rise again and fluctuate around values of 73 to 76. In *Figure 24* it can be seen that the segmentation and bounding box mAP behave very similarly, but the segmentation mAP is slightly higher than for the bounding box.

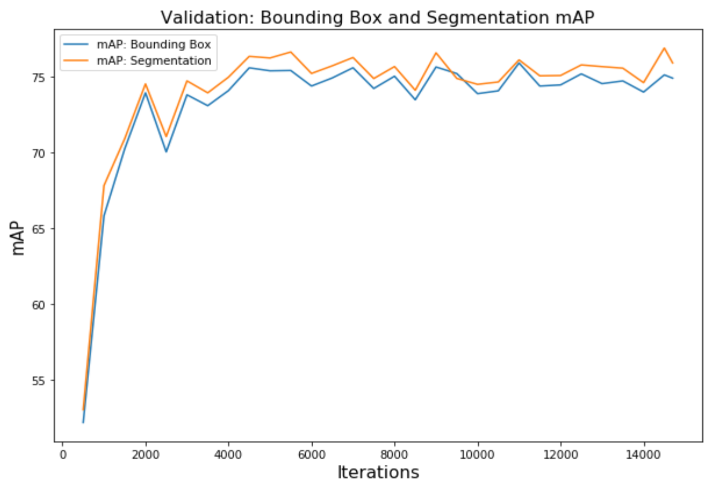

Figure 24: Validation bounding box mAP

Illustration of the performance of the bounding box and segmentation mAP of the validation set. The mAP is measured every 500th iteration step during the training process.

7.3. Quantitative Results

As mentioned in *7.1 Model Comparison* and *7.2 Model Training* the best performing model achieves a mAP of 82.69 for the bounding box and 82.83 for the segmentation mask. The detection accuracy of the final model is 0.935. For the test set, the MAE and MAPE are 1.6 cells and 4.4% respectively. For the N Cells, the MAE is slightly higher than on a total perspective. On average, the difference between the actual number of cells and the predicted number of cells is 1.7 cells per image or 6.1%. For the M Cells, the MAE is lower with 1.2 cells than for the N Cells. In contrast, the MAPE is higher with a value of 18.9%. All performance measures of the final model are listed in *Table 4***.** In total, the test set contains 2,747 cells and in total 2,777 cells are predicted, so the relative difference would be around 1.1% instead of 4.4%. These differences can be observed in *Figure 25.*

Table 4: Quantitative results of the best performing Mask R-CNN model on test set

Measure	**Performance**
Bounding Box MAP@50	82.69
Segmentation MAP@50	82.83
Accuracy	0.935
MAE – Total	1.6
MAPE – Total	4.4 %
Absolute Difference – Total	30
Relative Difference – Total	1.1%
MAE – N Cells	1.7
MAPE – N Cells	6.1 %
Absolute Difference – N Cell	1
Relative Difference – N Cell	0.04%
MAE – M Cells	1.2
MAPE – M Cells	18.9%
Absolute Difference – M Cells	31
Relative Difference – M Cells	7.5%

Overall, the test set contains 2,332 N cells and the model predicts 2,331 N cells and the difference between the actual and the predicted number of N cells is only 1 cell. The relative difference is thereby with 0.04% close to zero. Besides that, the differences in the total level for the M Cells are higher. The actual number of M Cells is 415, but the model predicts 446, so in total 31 cells are too much predicted. On the total perspective, the relative difference is around 7.5%.

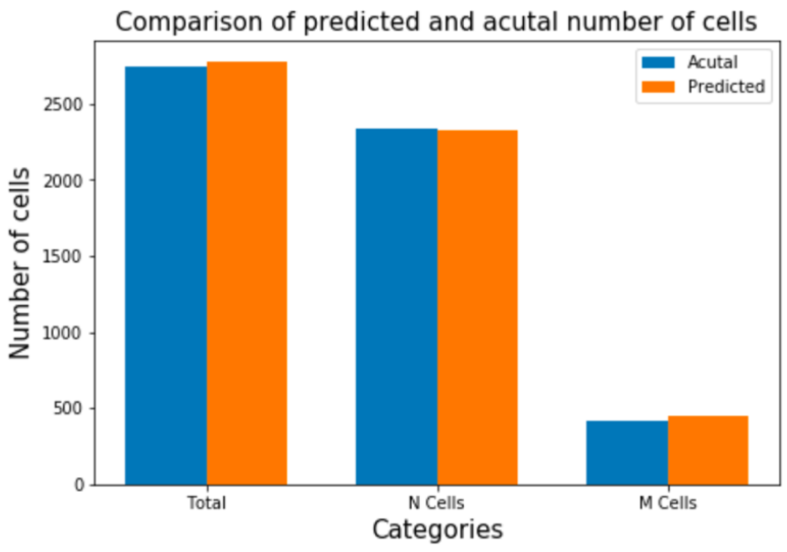

Figure 25: Comparison of the predicted and actual number of cells

Illustration of the number of actual and predicted cells on a total perfective and for both and within the two phenotype categories.

7.4. Qualitative Results

Besides evaluating the model with quantitative measures, a qualitative review of predictions should be conducted. In *Figure 26(A)* the ground truth image is compared with the predicted labels of the image *Figure 26(B)*. Yellow cells represent predicted N Cells and purple cells represent predicted M Cells. The percentage values in the corners of the bounding boxes for each cell represent the confidence of the model detecting and classifying a cell into one of the categories. By patiently comparing both images, differences can be observed. These range from a difference in classification of the macrophage cells, segmentation errors or differences between the ground truth annotation label and the predicted labels, as well as the difference of the actual detection process of the algorithm and the manual annotation. It can be used to gain a broad overview of the differences, but to perceive errors in detail this comparison is not detailed enough.

In *Figure 26(C),* the original image can be compared to the same image, in which the pixels of the predicted cells have been removed *Figure 26(D).* Therefore, the image is filtered for the pixels which are not part of a predicted object. The removed pixels are represented as black pixels. This can be used to observe if the algorithm missed cells in the prediction process. Moreover, it is useful to verify the segmentation process. Due to the bright color of the macrophage cells in the QPI DHM images, the cells are easily separable from the background for the human eye. With the pixels of the cells filtered out, it can easily be observed if large bright spots appear around a filtered cell. This would be a sign of bad segmentation, as all belonging pixels to a cell should be removed in that image.

Figure 26(E) represents the original image filtered by the pixels of only M Cells and *Figure 26(F)* shows only N Cells. These images are filtered the other way around than *Figure 26(D).* Here only pixels corresponding to a predicted cell are visualized and all other pixels are filtered out and represented as black pixels. This filtering technique does not help to observe if cells have not been detected, but it can be utilized to reveal the misclassifications of detected cells. When filtering for only one cell phenotype it can be visually perceived if only cells of one class are in the same image. If more than one class can be detected in an image, it is a sign of misclassification of the algorithm.

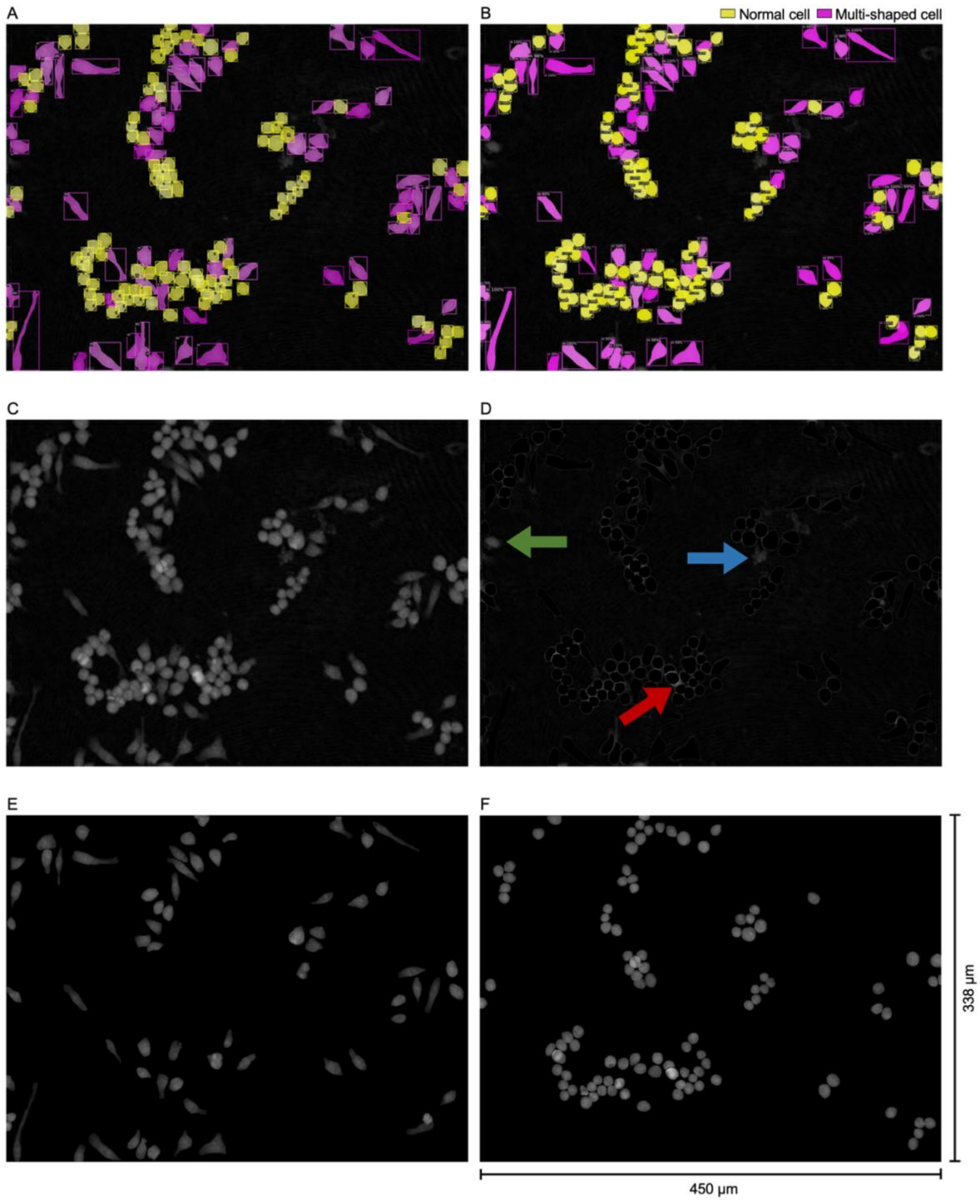

Figure 26: Visual result observation I

*(**A**): Illustration of the ground truth masks for normal macrophage cells (yellow) and multi-shaped cells (purple). (**B**): Illustration of predicted masks of the best performing model. (**C**): Sample image of macrophages generated by DHM. (**D**): Illustration of the presented microscopy image where pixels of the masks are removed, the green arrow points to a cell which was not detected by the algorithm, the blue arrow shows cell garbage, which was correctly not classified as a cell, the red arrow points to segmentation errors. (**E**): Illustration of the sample image where only pixels corresponding to a multi-shaped cell are presented. (**F**): Illustration of the sample image where only pixels corresponding to a normal cell are presented.*

In *Figure 27* it can be observed that the best performing model is also able to detect individual cells when they are clumping together, or even overlap completely.

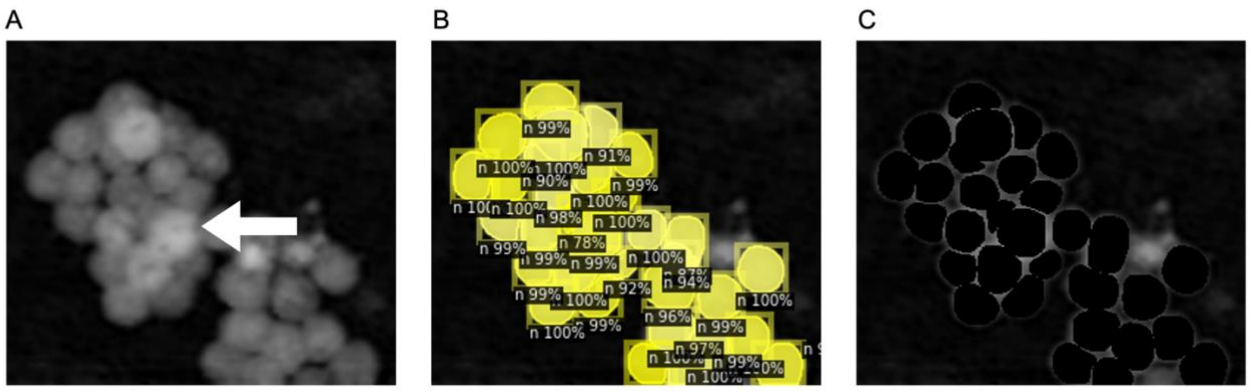

Figure 27: Visual result observation II

*(**A**): Section of an image, which shows clumping cells. The white arrow highlights an area of the section where cells are highly overlapping. (**B**): Illustration of the predicted masks of the cells with the corresponding model confidence. (**C**): Illustration of the presented section of an image where pixels of the masks are removed.*

The white arrow in Figure 27*(A)* points to an area where cells are overlapping. Section *(C)* of *Figure 27* shows that the model was able to separate between multiple cells in that area. However, the model suffers from some segmentation errors which can be observed as white circles around the black and round pixel-removed spots. In *Figure 27(B)*, it can be observed that the model confidence is rather high. Most of the times the model outputs a confidence level higher than 90%, sometimes even 100%. Only in the middle of the group of clumping cells the model outputs a confidence of 78% for one N Cell.

8. DISCUSSION

In this chapter, the previously presented results and findings of the model comparison, model training, quantitative results and qualitative results are analyzed and interpreted.

Model Comparison

When comparing results in *Table 4*, it can be observed that a high learning rate of 0.01 performs better when using a simpler backbone e.g., ResNet50. This leads to the conclusion that the model might be slightly overfitted to the train data set, due to the high learning rate and complex structure of the ResNet101 FPN. For a smaller learning rate of 0.001, it seems to be the opposite. The conclusion could be that the ResNet50 might not be able to capture the whole complexity of the dataset. The model using the ResNet50 performs more than 2 mAP points worse for both box and segmentation than the model trained with the ResNet101. The same can be witnessed when comparing the results for the smallest tested learning rate of 0.00025. The ResNet101 FPN Network outperforms the ResNet50 with more than 2 mAP points, for again both box and segmentation. This underlines that the ResNet50 is not complex enough to achieve optimal results. Nevertheless, the differences are relatively small. Even though the ResNet101 performs better, the ResNet50 backbone achieves acceptable results and can be considered if a less complex and less computational expensive model is needed e.g., for using a model in production.

Furthermore, it can be mentioned that in both ResNet50 and ResNet101 models, the performances in terms of mAP for the box and segmentation are close to equal, where the segmentation mAP is always slightly higher than the box mAP. This shows that once an object was detected, the cell structures seem not to be too complex for the algorithm to perform the segmentation.

Model Training

When observing the training loss in *Figure 18*, it can be recognized that the function has many spikes, which is an effect of having a small batch size (in this case, a batch size of two). As a result, the gradient update and loss value are highly dependent on only the two batches and therefore, the loss is sometimes slightly increasing or decreasing in respect to the batches. However, the global trend of the curve is decreasing. As mentioned under *7.2 Model Training,* the global trend of the validation loss has similar behavior. The loss is not increasing during the whole training process. It is always decreasing and is saturating after around 12,000 iterations. This is a sign of not overfitting on the training data, which is a highly desirable situation. All sub-loss functions for the training data are behaving as expected, steadily decreasing with small spikes, due to the small batch size.

In general, both validation and training loss curves and belonging sub-loss curves decrease very fast and only improve slightly for most of the training time. This reveals that the model is able to learn the greatest part of the object variance relatively fast and takes most of the training time for learning smaller details to better localize, detect, segment and classify the cells in the images. *Figure 21* also underlines this, as the accuracy spikes to over 90% in the first 100 iterations. The model seems to learn the cell detection process very quickly. The false-positive and false-negative error in *Figure 21* and *Figure 21* respectively indicate that the model more often falsely predicts something as an object when it is not an object, rather than not detecting an object, which is an object. This assumption is based on the fact that the False-Positive-Error is saturating below 0.1 and the False-Negative error decreases to around 0.045.

As mentioned in *7.2 Model Training*, both bounding box and segmentation mAP fluctuate around values of 73 to 76 after a strong increase in the approximately first 5,000 iterations. Due to this fact, it can be suspected that further training would not significantly increase the mAP. The fact that the mAP is not decreasing steadily in the last iterations of the training process is additional evidence that the model does not overfit on the training data.

Another interesting aspect is that the mAP for the validation set is lower than for the test set. The mAP for the validation set after 600 epochs or 14,700 iterations is 74.87 and 75.87, while for the test set it is 82.69 and 82.83 for the bounding box and segmentation, respectively. One assumption for this phenomenon is that this happens due to the unequal size of the validation and test set and the corresponding variance of the training and the test set.

Quantitative Results

When comparing the MAPE with the relative difference, it can be observed that the relative difference is always lower than the MAPE. This is due to the model, which sometimes overpredicts and sometimes underpredicts the number of cells in the images. Consequently, the differences are averaging each other out and on a whole-dataset-perspective, the differences are lower than on a per-image-perspective.

When observing the MAE, the model seems to better detect M Cells. The MAE of M Cells is 1.2. In comparison, the MAE for N Cells is 1.7 for this model. In contrast, when observing the MAPE the interpretation is the opposite. The MAPE for M Cells is, with 18.9%, around three times higher than the relative difference for the N Cells, which is 6.1%. This can be explained by the unequal distribution of N and M Cells in the microscopy images. As the histogram in *Figure 16* revealed that in the majority of cases only very few M Cells are present in the images. If the model misclassified a singles M Cell or did not detect it and there are only two M Cells, the relative error is already 50%. This makes the MAE, MAPE and absolute and relative difference suboptimal performance metrics, but it can reveal that on a total-dataset-perspective, the model rather overestimates the number of M Cells than underestimating it.

Qualitative Results

The visual observation of the prediction results of the final model leads to the conclusion, that the detection and segmentation task was successfully performed. *Figure 26* shows that it can clearly be distinguished between the two phenotypes and that the ground truth and predicted masks are very similar. Moreover, the model does not predict cell debris as cells and the detection, segmentation and separation of cells even deliver desirable results when cells are clumping together or are overlapping each other, which can be observed in *Figure 27*. In addition, the model was able to learn and generalize the various shapes of the M Cells, which can occur. However, sometimes individual cells are not being predicted as cells correctly and small segmentation errors emerge. These segmentation errors can be observed by the halo-like white spots in the images of *Figure 26(D)* and *Figure 27(C),* where the pixels of the predicted masks are being removed.

9. CONCLUSIONS, LIMITATIONS AND FUTURE WORK

Based on a literature review, a Mask R-CNN Deep Learning model was proposed to detect, segment, classify and quantify RAW 264.7 mouse macrophage cells in QPI images generated by DHM. Parameterization options of the algorithms were tested, evaluated, optimized and manually adjusted in order to enhance the performance of the algorithm for this analyzed dataset. The number of cells per image was one curtail aspect, as the number can highly vary between the images. Some images contain more than several hundred cells and parameters needed to be adjusted accordingly. The results achieved and described in *7 Results* show that the trained Mask R-CNN is indeed capable of achieving high-quality results. In addition, it was found that a complex CNN architecture, namely the ResNet101, performed best as a feature extractor when choosing a medium learning rate size of 0.001. By observing the achieved results, conclusions on the research questions can be drawn accordingly.

9.1. ANSWERING RESEARCH QUESTIONS

1. **Is it possible to distinguish between two phenotypes of macrophage cells using Deep Learning approaches?**

   The proposed, trained and evaluated Mask R-CNN model is capable of distinguishing N Cells and M Cells, which correspond to two phenotypes of the same cell type. The detection accuracy is with a value of 93.5% a satisfying result. The MAE of N cells is 1.6 and for M Cells it amounts to 1.2. This shows that the chosen approach can distinguish between two phenotypes of macrophage cells with small errors.

2. **Is it possible to segment the macrophage cells on a single cell level using Deep Learning approaches?**

   The Mask R-CNN is also able to segment detected objects. The mAP of the segmentation is 82.83. In addition, the successful segmentation process could be confirmed visually.

3. **Can a Deep Learning approach be used to detect and segment individual cells, even when cells are growing in clusters and overlapping?**

   The quantitative and visual results of the proposed approach show that the best performing model, based on quantitative measures, is also able to detect and segment individual cells, even when they are growing in clusters and overlapping.

4. **Is it possible to achieve the detection, segmentation and classification process with only a single Deep Learning algorithm or is a combination of multiple Deep Learning algorithms required?**

   The Mask R-CNN algorithm is able to solve the instance detection, segmentation and classification in an end-to-end manner. No further post-processing or combinations of multiple algorithms need to be used.

5. **How well does the detection and segmentation process perform on label-free DHM QPI images?**

   For this dataset of DHM QPI images, containing RAW 264.7 mouse macrophages, the proposed Mask R-CNN archives a bounding box mAP of 82.69, segmentation mAP of 82.83 and a classification accuracy of 93.5%.

In summary, it can be stated that the chosen approach was appropriate to successfully answer the research questions according to this case study.

9.2. Implications based on this work

This work shows that robust automated analysis of label-free microscopy images is indeed possible and fluorescence staining is not obligatory for a detection, segmentation, classification and quantification analysis process based on Deep Learning algorithms. Even though this work only evaluates the performance of a Mask R-CNN Deep Learning model for living RAW 264.7 macrophages in DHM QPI images, it shows great potential to accelerate image analysis in cell biology and medical research. A reliable automated analysis system for label-free microscopy images can improve the quality and time efficiency of cell biology research studies.

9.3. Limitations

While the results show that the proposed solution is a sufficient and good approach to answer the research questions, there are limitations, which need to be taken into account when making assumptions based on this work.

The literature review revealed that the Mask RCNN is a robust approach for microscopy image analysis, as it seems to outperform other neural network architectures, such as U-Net, when it is desired to detect overlaying cells. Even though this study proves that the Mask R-CNN is indeed able to detect cells in clusters, the U-Net was not evaluated, tested and compared to the Mask R-CNN. As described in *3. Literature Review,* the studies were not conducted on DHM QPI images and were not applied on RAW 264.7 mouse macrophages. The U-Net architecture might also be a good solution for this specific dataset, evaluated in this thesis. This work is only reflecting the performance of the Mask R-CNN and cannot be used for comparing it with other neural network architectures.

The Mask R-CNN has many configuration parameters, as described in *5.1 Model* and *5.2 Training.* The model was optimized based on only two parameters, namely the FPN backbone architecture and the learning rate. The reason for not optimizing all parameters was the limitation of GPU computation usage. All remaining parameters have been set, based on a qualitative evaluation and conclusion, based on the characteristics of the data. The configuration of the best performing model, showed in *Table 5* in the *Appendix*, is not comprehensively evaluated and optimized. This aspect needs to be taken into consideration, if conclusions or assumptions are being made based on this work.

As mentioned in *8. Discussion*, the mAP for the test set is higher than for the validation set. As described, this might be due to the unequal size and possible variance of the data sampled for the train, test and validation set. To fully answer and prove that assumption, the data should be cross-validated for making a more reliable conclusion about the different behavior of the mAP for the respective data sets. The cross-validation process is very computationally expensive, as a lot of training processes need to be undertaken for one specific set of parameter settings. The limited GPU usage capacity was again the main reason for not being able to evaluate cross-validation for the training processes.

9.4. Recommendations for future work

The recommendations for potential future works are strongly related to the limitations of this thesis. Future work can mainly focus on optimizing all configuration parameters of the Mask R-CNN model and training process. Various parameter settings can be evaluated based on the increase or decrease in the performance of the model. In addition, this optimizing process can and should be done using cross-validation, which delivers more reliable and generalizable results for conclusions and assumptions based on a set of parameters. Moreover, if a large GPU computation capacity is available, the model can be trained with a higher batch size. This could reveal if the model can generalize better and faster if the gradients are updated based on more data on a single updating step. Then it could also be observed if the loss function would still suffer from small spikes, which was assumed to be due to the small batch size in this work.

Besides optimizing the already proposed model, future work could also focus on comparing this model to other segmentation neural network approaches, such as the various U-Net based architectures. This might reveal if the Mask R-CNN also outperforms other approaches on DHM QPI images.

Since this work showed that the Mask R-CNN model can distinguish between multiple phenotypes of the same cell type, another interesting work could be to evaluate if a generalized model can be created, in order to detect and segment a various number of different cell types in images, imaged by different microscopy techniques. This would offer a great benefit for biologists around the world, who do not have the resources to create and deploy custom trained neural networks.

10. BIBLIOGRAPHY

Baczewska, M., Eder, K., Ketelhut, S., Kemper, B., & Kujawińska, M. (2020). Refractive Index Changes of Cells and Cellular Compartments Upon Paraformaldehyde Fixation Acquired by Tomographic Phase Microscopy. *Cytometry Part A*, *n/a*(n/a). https://doi.org/10.1002/cyto.a.24229

Bisong, E. (2019). Google Colaboratory. In E. Bisong (Hrsg.), *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners* (S. 59–64). Apress. https://doi.org/10.1007/978-1-4842-4470-8_7

Caicedo, J. C., Goodman, A., Karhohs, K. W., Cimini, B. A., Ackerman, J., Haghighi, M., Heng, C., Becker, T., Doan, M., McQuin, C., Rohban, M., Singh, S., & Carpenter, A. E. (2019). Nucleus segmentation across imaging experiments: The 2018 Data Science Bowl. *Nature Methods*, *16*(12), 1247–1253. https://doi.org/10.1038/s41592-019-0612-7

Caicedo, J. C., Roth, J., Goodman, A., Becker, T., Karhohs, K. W., Broisin, M., Csaba, M., McQuin, C., Singh, S., Theis, F., & Carpenter, A. E. (2018). *Evaluation of Deep Learning Strategies for Nucleus Segmentation in Fluorescence Images* [Preprint]. Bioinformatics. https://doi.org/10.1101/335216

De Myttenaere, A., Golden, B., Grand, B. L., & Rossi, F. (2016). Mean Absolute Percentage Error for regression models. *Neurocomputing*, *192*, 38–48. https://doi.org/10.1016/j.neucom.2015.12.114

Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2015). The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, *111*(1), 98–136. https://doi.org/10.1007/s11263-014-0733-5

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, *88*(2), 303–338. https://doi.org/10.1007/s11263-009-0275-4

Girshick, R. (2015). Fast R-CNN. *ArXiv:1504.08083 [Cs]*. http://arxiv.org/abs/1504.08083

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *ArXiv:1311.2524 [Cs]*. http://arxiv.org/abs/1311.2524

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*., 98 - 373

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2018). Mask R-CNN. *ArXiv:1703.06870 [Cs]*. http://arxiv.org/abs/1703.06870

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *ArXiv:1512.03385 [Cs]*. http://arxiv.org/abs/1512.03385

Hossin, M., & M.N, S. (2015). A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process*, *5*, 01–11. https://doi.org/10.5121/ijdkp.2015.5201

Iglovikov, V. I., Seferbekov, S., Buslaev, A. V., & Shvets, A. (2018). TernausNetV2: Fully Convolutional Network for Instance Segmentation. *ArXiv:1806.00844 [Cs]*. http://arxiv.org/abs/1806.00844

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning* (Bd. 103). Springer New York. https://doi.org/10.1007/978-1-4614-7138-7

Jensen, E. (2012). Use of Fluorescent Probes: Their Effect on Cell Biology and Limitations. *Anatomical record (Hoboken, N.J. : 2007)*, *295*. https://doi.org/10.1002/ar.22602

Kemper, B., & Bally, G. von. (2008). Digital holographic microscopy for live cell applications and technical inspection. *Applied Optics*, *47*(4), A52–A61. https://doi.org/10.1364/AO.47.000A52

Lenz, P., Brückner, M., Ketelhut, S., Heidemann, J., Kemper, B., & Bettenworth, D. (2016). Multimodal Quantitative Phase Imaging with Digital Holographic Microscopy Accurately Assesses Intestinal Inflammation and Epithelial Wound Healing. *Journal of Visualized Experiments*, *2016*. https://doi.org/10.3791/54460

Li, Y., Wei, C., & Ma, T. (2020). Towards Explaining the Regularization Effect of Initial Large Learning Rate in Training Neural Networks. *ArXiv:1907.04595 [Cs, Stat]*. http://arxiv.org/abs/1907.04595

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., & Dollár, P. (2015). Microsoft COCO: Common Objects in Context. *ArXiv:1405.0312 [Cs]*. http://arxiv.org/abs/1405.0312

Loshchilov, I., & Hutter, F. (2017). SGDR: Stochastic Gradient Descent with Warm Restarts. *ArXiv:1608.03983 [Cs, Math]*. http://arxiv.org/abs/1608.03983

Min, J., Yao, B., Ketelhut, S., Engwer, C., Greve, B., & Kemper, B. (2017). Simple and fast spectral domain algorithm for quantitative phase imaging of living cells with digital holographic microscopy. *Optics Letters*, *42*(2), 227–230. https://doi.org/10.1364/OL.42.000227

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill., 1 - 128

Novikov, A. A., Lenis, D., Major, D., Hladůvka, J., Wimmer, M., & Bühler, K. (2018). Fully Convolutional Architectures for Multi-Class Segmentation in Chest Radiographs. *ArXiv:1701.08816 [Cs]*. http://arxiv.org/abs/1701.08816

Park, Y. K., Depeursinge, C., & Popescu, G. (2018). Quantitative phase imaging in biomedicine. *Nature Photonics*, *12*(10), 578–589. https://doi.org/10.1038/s41566-018-0253-x

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *arXiv:1912.01703 [cs, stat]*. http://arxiv.org/abs/1912.01703

Patterson, J., & Gibson, A. (2017). *Deep learning: A practitioner's approach* (1. Aufl.). O'Reilly Media. https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=1564784

Pavillon, N., Fujita, K., & Smith, N. (2014). Multimodal Label-free Microscopy. *Journal of Innovative Optical Health Sciences*, *7*, 2014. https://doi.org/10.1142/S1793545813300097

Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with

Region Proposal Networks. *ArXiv:1506.01497 [Cs]*. http://arxiv.org/abs/1506.01497

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *ArXiv:1505.04597 [Cs]*. http://arxiv.org/abs/1505.04597

Saito, T., & Rehmsmeier, M. (2015). The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLoS ONE*, *10*(3). https://doi.org/10.1371/journal.pone.0118432

Schilling, L. (2020). *Generating synthetic brain MR images using a hybrid combination of Noise-to-Image and Image-to-Image GANs*. http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-166034

Schmidt, U., Weigert, M., Broaddus, C., & Myers, G. (2018). Cell Detection with Star-convex Polygons. *arXiv:1806.03535 [cs]*, *11071*, 265–273. https://doi.org/10.1007/978-3-030-00934-2_30

Shu, J.-H., Nian, F.-D., Yu, M.-H., & Li, X. (2020). An Improved Mask R-CNN Model for Multiorgan Segmentation. *Mathematical Problems in Engineering*, *2020*, 1–11. https://doi.org/10.1155/2020/8351725

Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv:1409.1556 [Cs]*. http://arxiv.org/abs/1409.1556

Vuola, A., Ullah Akram, S., & Kannala, J. (2019). *Mask-RCNN and U-net Ensembled for Nuclei Segmentation*.

Wada, K. (2016). *labelme: Image Polygonal Annotation with Python*. https://github.com/wkentaro/labelme

Wang, D., Li, C., Wen, S., Han, Q.-L., Nepal, S., Zhang, X., & Xiang, Y. (2020). Daedalus: Breaking Non-Maximum Suppression in Object Detection via Adversarial Examples. *ArXiv:1902.02067 [Cs]*. http://arxiv.org/abs/1902.02067

Willmott, C., & Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research*, *30*, 79–82. https://doi.org/10.3354/cr030079

Wu, Y., Kirillov, A., Francisco, M., Lo, W.-Y., & Girshick, R. (2019). *Detectron2*. https://github.com/facebookresearch/detectron2

Xing, F., Xie, Y., Su, H., Liu, F., & Yang, L. (2018). Deep Learning in Microscopy Image Analysis: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(10), 4550–4568. https://doi.org/10.1109/TNNLS.2017.2766168

Yuan, P., Rezvan, A., Li, X., Varadarajan, N., & Nguyen, H. (2019). Phasetime: Deep Learning Approach to Detect Nuclei in Time Lapse Phase Images. *Journal of Clinical Medicine*, *8*, 1159. https://doi.org/10.3390/jcm8081159

Zagoruyko, S., & Komodakis, N. (2017). Wide Residual Networks. *ArXiv:1605.07146 [Cs]*. http://arxiv.org/abs/1605.07146

11. APPENDIX

Table 5: Mask R-CNN parameterization

Network Part	Parameter	Configuration
General	Solver	SGD
Feature Pyramid Network	Backbone Architecture	ResNet50 ResNet101
Anchor Generation	Anchor Sizes	16 32 64
	Aspect Ratios	0.5 1 2
Region Proposal Network	IoU Background	0.3
	IoU Foreground	0.7
	Training Pre NSM-Proposals	12,000
	Testing Pre NSM-Proposals	6,000
	NMS Threshold	0.4
	Training Post NSM-Proposals	4,000
	Testing Post NSM-Proposals	2,000
	RPN Loss Weight	1.5
ROI	IoU	0.5
	Batch Size per Image	512
	Positive Fraction Ratio	0.25
	Test NMS	0.4
	Confidence Threshold	0.6
Training	Warm-up factor	1/1000
	Momentum	0.9
	Weight Decay	0.0001
	Epochs	600
	Learning Rate	0.01 0.001 0.00025