T.C.
ALTINBAS UNIVERSITY
Institute of Graduate Studies
Information Technologies

THE ABILITY OF NEURAL NETWORKS TO SOLVE BIOMEDICAL IMAGE SEGMENTATION TASK

ZINAH ABDULLAH AHMED

Master of Science

Supervisor

Prof. Dr. Osman Nuri Ucan

Istanbul, 2020

THE ABILITY OF NEURAL NETWORKS TO SOLVE BIOMEDICAL IMAGE SEGMENTATION TASK

By

Zinah Abdullah Ahmed

Information Technologies

Submitted to the Institute of Graduate Studies in partial fulfillment of the requirements for the degree of Master of Science

ALTINBAŞ UNIVERSITY

2020

The thesis titled “THE ABILITY OF NEURAL NETWOK TO SOLVE BIOMEDICAL IMAGE SEGMENTATION TASKE” prepared and presented by Zinah Abdullah AHMED was accepted as a Master of Science Thesis in Information Technologies.

____

Prof. Dr. Osman Nuri Ucan

Supervisor

Thesis Defense Jury Members:

Prof. Dr. Osman Nuri Ucan	School of Engineering and Natural Sciences, Altinbas University	___________________
Asst. Prof. Dr. Abdullahi Abdu Ibrahim	School of Engineering and Natural Sciences, Altinbas University	___________________
Assoc. Prof. Dr. Adil Deniz Duru	Faculty of Sport Sciences Marmara University	___________________

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Approval Date of Institute of Graduate Studies: ____/____/____

DEDICATION

First I would like to Allah Almighty for the power of mind, health, strength, guidance knowledge and skills to complete this study .This thesis is wholeheartedly dedicated to my husband, who supported me and stood beside me and to my beloved father, who have been my source of inspiration, he tells me that" every success in your life will be best gift for me". To my mother, who have been supporting me with the kind and pure love.

ACKNOWLEDGEMENTS

I would like to thank my supervisor professor Prof. Dr. Osman Ucan for all support during my study. Its great pleasure to express my deepest gratitude to my friends who have shared with me best moments during my study for the Master degree.

ABSTRACT

THE ABILITY OF NEURAL NETWORKS TO SOLVE BIOMEDICAL IMAGE SEGMENTATION TASK

AHMED, Zinah Abdullah Ahmed,

M.Sc, Information Technologies, Altınbaş University,

Supervisor: Prof. Dr. Osman Nuri Ucan

Date: 23/09/2020

Pages: 51

Modern medical imaging techniques lead to the growth of image data   being captured .It allows the investigation of dynamic processes in living cells, e.g., membrane dynamics, viral infection and gene transport. In all these problems, the key part is an ability to detect cells or their organelles in a microscopy image and follow them in time-lapse sequences of images. Manual processing of all these data is not feasible .Automation can help to process them more effectively and extract quantitative information. There are several reasons why deep learning methods and especially neural networks are suitable for image segmentation. Deep learning is one of them a chine learning techniques with a high ability to adapt and generalize. In this thesis, we designed and trained own convolutional neural network model on a Cell Tracking Challenge dataset, which provides image segmentation.

The aim of the work is not just to reach a good result in the challenge but also to explore this method and to get experienced with it.

Keywords: Deep Learning, Image Segmentation, Convolutional Neural Network (CNN)

ÖZET

SİNİR AĞLARININ YETENEĞİBİYOMEDİKAL GÖRÜNTÜLERİ ÇÖZMEK İÇİNSEGMENTASYON GÖREVİ

Zinah Abdullah Ahmed, AHMED

Yüksek Lisans, Bilgi Teknolojileri, Altınbaş Üniversitesi,

Danışman: Prof. Dr. Osman Nuri Ucan

Tarih: 23/09/2020

Sayfalar: 51

Modern tıbbi görüntüleme teknikleri, yakalanan görüntü verilerinin büyümesine yol açar. Canlı hücrelerdeki dinamik süreçlerin, örneğin membran dinamikleri, _viral enfeksiyon ve gen transportunun araştırılmasına izin verir. Tüm bu problemlerde, anahtar kısım, mikroskopi görüntüsünde hücreleri veya organellerini tespit etme ve bunları zaman atlamalı görüntü dizilerinde takip etme yeteneğidir. Tüm bu verilerin manuel olarak işlenmesi mümkün değildir. Otomasyon onları daha etkili bir şekilde işlemeye ve başka bir nicel bilgi çıkarmaya yardımcı olabilir. Derin öğrenme yöntemlerinin ve özellikle sinir ağlarının görüntü segmentasyonu için uygun olmasının birkaç nedeni vardırderin öğrenme uyum ve genelleme yeteneği yüksek makine öğrenim tekniklerinden birisidir. Bu tezde, acelltracking veri seti üzerinde görüntü segmentasyonu sağlayan kendi evrişimli sinir ağı modelini tasarladık ve eğittik. çalışmanın amacı, sadece meydan okumada iyi bir sonuca ulaşmak değil, aynı zamanda yöntemi araştırmak ve onunla deneyimlemektir.

Anahtar Kelimeler: Derin Öğrenme, Görüntü Segmentasyonu,Evrişimli Sinir Ağı.

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

LIST OF ABBREVIATIONS

CNN	:	Convolutional Neural Network.
DIC	:	Differential Interference Contrast.
PHC	:	Phase Contrast
FCN	:	Fully Convolutional Network
G T	:	Ground truth

LIST OF SYMBOLS

Ø	:	Empty set.
$\sum$	:	Sum of.
λ	:	Wave length
Δ	:	Triangle or delta
Θ	:	Theta
∂	:	Partial derivative

1. INTRODUCTION

Neural networks and deep learning are popular buzzwords today. These methods have already solved many problems that seemed to be too hard or even impossible for a computer and the research in this field is not over yet. This thesis explores the ability of neural networks to solve biomedical image segmentation task.

Modern medical imaging techniques lead to the growth of image data being captured. It allows the investigation of dynamic processes in living cells, e.g., membrane dynamics [1], viral infection [2] and gene transport [3]. In all these problems, the key part is an ability to detect cells or their organelles in a microscopy image and follow them in time-lapse sequences of images. Manual processing of all these data is not feasible. Automation can help to process them more effectively and extract quantitative information.

In 2012, international bio image processing community organized a challenge called ISBI Cell Tracking Challenge [4] [5]. The competition task is to segment the given microscope video and to track cell migration and proliferation over time. This task can be divided into two parts. In the first part, all the images from the given sequence are segmented in the purpose of cell detection. In the second part, segmentation masks of individual cells are linked from frame to frame. Authors of the challenge also introduce their methodology how to compare the performance of different cell tracking algorithms. After three editions of the challenge, they compared all the participating algorithms [6]. Machine learning methods appeared in top-three places in all the datasets. These methods were successful especially in contrast enhancement microscopy datasets where their performance overcomes all the expectations. On the other hand, the performance of published methods is still behind the human-level ability to solve this task, so there is a wide space for improvement.

There are several reasons why deep learning methods and especially neural networks are suitable for image segmentation. Deep learning is one of the machine learning techniques with a high ability to adapt and generalize. Convolutional neural network is a type of deep learning model that is designed to process image data. In a machine learning approach, it is not necessary to explicitly define the task solving process.Itis only required to determine the expected behavior of the solver precisely. For images in difficult modalities, we do not have sufficient knowledge of how to segment them, but there is a natural definition of expected results. In this thesis,

We designed and trained own convolutional neural network model on a Cell Tracking Challenge dataset, which provides image segmentation. The aim of the work is not just to reach a good result in the challenge, but also to explore this method and to get experienced with it. Image segmentation is not a standard task for neural networks. We had to study particular techniques how to adapt a neural network to this kind of task. To verify the suitability of convolutional networks for image segmentation in difficult imaging conditions, we have selected the most challenging dataset, namely DIC-C2DH-HeLa. At the end of the work, we compare our final model performance with other competitors in the Cell Tracking Challenge

2. THE BASICS OF NEURALNETWORKS

When somebody hears about neural networks for the first time, he naturally expects something closely connected with a human brain. At the very beginning of this concept, there was a direct inspiration from neuroscience. It has changed, and through the years deep learning grew to an independent field of study. It adapts techniques from fields of machine learning, linear algebra and probability theory. Artificial neural networks can be most adequately characterized as computational models with particular properties such as the ability to adapt, to generalize, or to cluster or organize data, and whose operation is based on parallel processing [7]. In this thesis, the term neural network or network always denotes artificial neural network. This chapter offers an excursion to the basics of the neural networks. We begin with the short history of the concept. Then we introduce a terminology of neural networks and some important principles.

2.1 NEURAL NETWORK HISTORY

The history of neural networks goes back to 1943 when the first artificial neuron was designed as a mathematical model of a neuron cell. This model was presented by McCulloch and Pitts [8] and represents a binary classifier. The behavior of the model was controlled by weights, which had to be set manually by a human operator. As a part of the research, they also built a physical model of the neuron using electrical circuits. It had no practical usage but inspired other scientists to follow their research. In 1957 F. Rosenblatt [9] published a neuron model called Perceptron that could learn weights to represent given data. Rosenblatt also proved, that his learning algorithm always found the optimal weight vector if it exists. His research group built a computer named Mark I Perceptron, which was trained to recognize letters. Letters were projected on a wall and perceived by 20x20 agrid of light sensitive sensors. Two years later it was published another learnable network called ADALINE (ADAptive LInear Element) [10]. It was built to illustrate adaptive behavior and artificial learning. The network was trained to recognize binary patterns. ADALINE introduced a special case of a learning algorithm called stochastic gradient descent. Perceptron and ADALINE belong to linear models, which are still used in the field of machine learning. In 1969 an article by Minsky and Papert [11] was published that be- came famous due to the rejection of neural network research. Authors discussed two well-known facts. First, one layer perceptron is not able to compute the logical XOR function. Second, no learning algorithm for multi-layer perceptron exists. They deduced that further research in the field of neural networks is not reasonable at all. This statement had a big impact, and researches focused on other fields of artificial intelligence in the next decade.

Another renaissance of neural networks came with research of physicist John Hopfield, who in two articles from 1982 [12] and 1984 [13] showed a relation between a neural network model and a physical behaviour of magnetic materials. In 1986 G. Hinton and R. Williams

[14] published an article where they described an algorithm called backpropagation, which allows learning of multilayer perceptrons. This algorithm remains dominant for a neural network training until today. facts Kunihiko Fukushima in his research Neocognitron [15] provides an interesting connection between neural networks and neuroscience. He introduced an architecture for image processing that was inspired by a mammalian visual system. This architecture became the basis for the modern convolutional networks. He also expected that many computational units connected can simulate intelligent behaviour.In 1989, the first autonomous car called ALVINN (Autonomous Land Vehicle In a Neural Network) was built at Carnegie Mellon University [16]. It used a front camera with resolution 32x30 pixels to predict a driving direction. The network had only five neurons in the hidden layer and one output neuron. The computational power of this vehicle was limited due to the hardware weight. The experiment showed that even a simple neural network could deal with practical tasks. In the 90s, many new machine learning methods were discovered and used. There were some promising results of neural network techniques, but he research remained a theoretical field of study. The situation changed rapidly in 2012 when a speech recognition neural model achieved a state-of-the-art performance for the first time. In the same year, Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton won with the convolutional neural network ImageNet Large Scale VisualRecognition Challenge (ILSVRC)[17] by a large margin. They reduced the ImageNet dataset error rate from 26% to 16%. Their model called AlexNet introduced several components that have become mainstays in the

convolutional networks.As we can see, neural networks are not a new concept at all. To finish this section, we introduce a context that supports the growth of this field today.

Firstly, researchers in this field can use new hardware supporting parallel computations. In the past, training of a small network was computationally demanding. A neural network has a parallel computational scheme. Many similar and independent computations are made at the same time. The parallel computational scheme.Many similar and independent computations are made at the same time.The architecture of modern GPUs is suitable for this kind of task. The usage of a GPU allows training big and complex neural networks faster than before.

Secondly, the size of learnable data is rapidly growing as well as the size of datasets. The larger dataset leads to lower generalization error, and it raises the model accuracy on unseen data.

Finally, the current success of neural networks attracts future development. In the last ten years, big companies such as Google,IBM, NVIDIA as well as many universities published new methods and software that make their search of neural networks even more accessible.

2.2 DEEP FORWARD NETWORKS

Deep Feed forward Networks (Multilayer Perceptron, MLP) are one of the basic classes of neural networks. We are going to focus on them because they’re suitable for demonstration of the main principles of deep learning. A deep feed forward network can be described as a universal function approximator [18]. It is formed by a network of many processing units, which connections can be drawn as a directed acyclic graph. This section contains a description of an artificial neuron model how are these neurons connected to a neural network. It describes learning and back propagation principles. There is also a discussion about the behavior of the network during the training and how the training can be controlled by a hyper parameter setting.

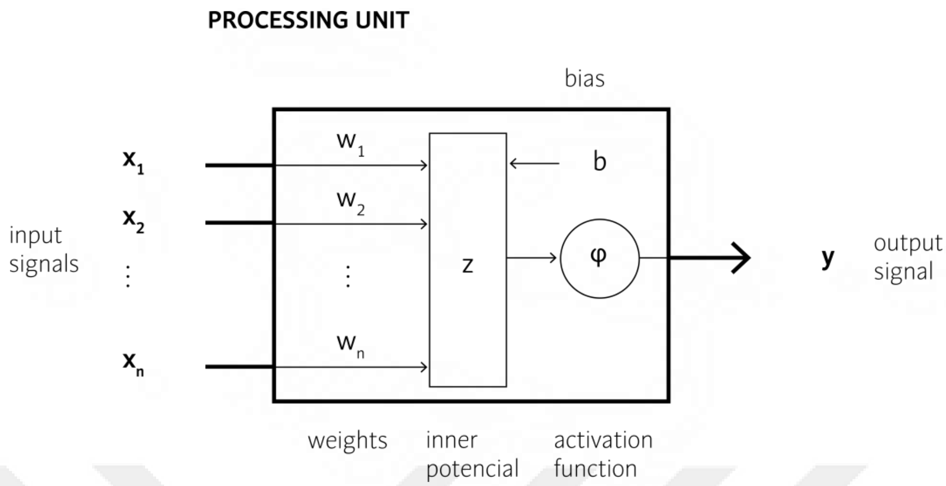

Figure 2.1: Processing unit model

2.2.1 Neuron Model

The artificial neuron (processing unit) is a basic component of each neural network. The common network consists of at least a thousand of neurons. These units provide all the network's computation power. The structure of one artificial neuron is shown in Figure 2.1. Each neuron receives input signals from neighbor neurons together with one extra input signal called bias or threshold. All the input signals transform into one value called the inner potential. Output signal is a function of the inner potential called activation function. The output signal propagates to the other neurons. There is a dense structure of connections between the neurons. Each connection has its own weight or throughput. Neurons communicate to each other only by sending signals. Processing unit can be defined as multivariate function. Let the unit has $n \in N$ input connections. Formally it receives input signals $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ and a bias signal $b \in \mathbb{R}^n$. Input signals are weighted by vector $\mathbf{w} = (w_1, w_2, \ldots, w_n) \in \mathbb{R}^n$. Weight of the bias is 1, by definition. The inner potential z is a function of input vector, which is parametrized by vector $\mathbf{w}$ and scalar b. A unit the output a is a function $\phi : \mathbb{R} \rightarrow \mathbb{R}$ of the inner potential. The following

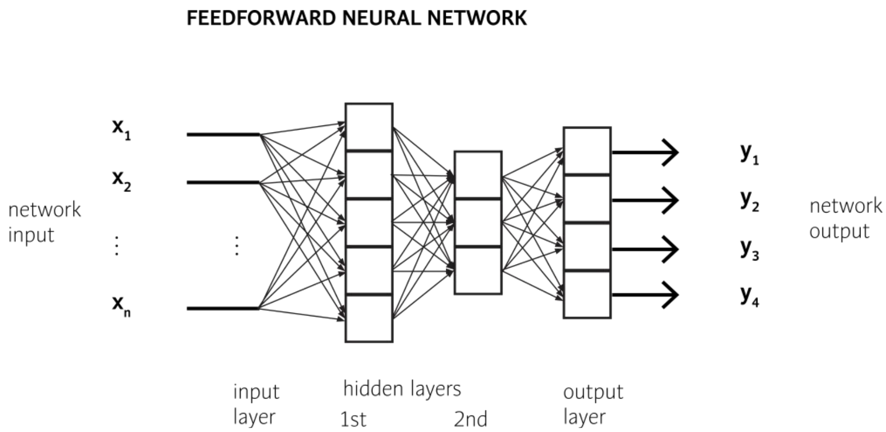

Figure 2.2: Formulas hold

$$Z(X) \coloneqq \sum_{i=1}^{n} wibi + b,$$

$$a \coloneqq \emptyset\big(z(x)\big).$$

We can change the behavior of the processing unit by modifying **w** and b , which are also called learnableparameters of the given unit.

2.2.2 Network Topology

A neural network consists of many computational units connected to each other. A schema of connections between the units is called the network topology. A feedforward network topology is based on layers. It simplifies network behaviour by a strict definition of all unit connections. A layer is a subset of neural network processing units. Each unit belongs to just one layer. Layers are stacked one after another. In case of feed forward neural networks, each unit from one layer sends signals to all units in the next layer. A topology of small feed forward network is shown in Figure 1.2. The first layer is called input layer. It does not provide any computation and only passes data to the next layer. The last layer is called output layer. The other layers are called hidden layers. At the schema, we can also see all the connections.

Let suppose a feed forward neural network has a depth $k \in \mathbb{N}$.

Variable $n_i \in \mathbb{N}$, $i \in \{1, 2, \dots, k\}$ denotes the number of neurons in i-th hidden layer. Variable $n \in \mathbb{N}$ is a size of the input vector. The hidden layers as well the output layer can be described as functions of the previous layer output signals. The hidden layer number $i \in \{1, \dots, k-1\}$ can be described by a mapping $h^{(i)}: \mathbb{R}^{n_{i-1}} \longrightarrow \mathbb{R}^{n_i}$. The output layer is a mapping $o^{(k)}: \mathbb{R}^{n_{k-1}} \longrightarrow \mathbb{R}^{n_k}$ The whole network can be then seen as function $f(x): \mathbb{R}^{n_0} \longrightarrow \mathbb{R}^{n_k}$, where **x** is the input vector. Function f can be also written as a function concatenation

$$f := {}^{(k)} \circ h^{(k-1)} \circ h^{(k-2)} \circ \dots \circ h^{(1)} \circ \mathbf{x}.$$

In practise, we are only interested in the function f value and not in values of the layer functions. The function f does not uniquely determine neither hidden layer functions h nor output layer function o.

2.2.3 Activation Functions

In this subsection, we discuss the choice of activation functions. Let's suppose, that all the neurons in one layer have the same activation function. The usage of activation function is to introduce a non-linearity to the model. If there were no such a function, a neural network would be capable of representing only linear functions, no matter how many processing units or layers it has. Adding new layers with non-linear activation function exponentially increases the complexity of the function represented by the network. That is why deep networks are considered to be a powerful tool. The activation function is usually non-decreasing and some threshold function. It should be differentiable due to the learning process, but in practice, the derivative could also be approximated. Let us now list some possible choices of activation functions Signum function (Binary step function) sets a positive input to 1and negative input to−1.It is common in historical example ssuch as Perceptron

$$\mathrm{sgn}(z)\begin{cases}-1 \text{ if } z<0\\ 0 \text{ if z} = 0\\ 1 \text{ if z} > 1\end{cases} \tag{2.1}$$

.

.

Sigmoid function gives smooth activations in rang (0 ,1). It is used in hidden layers as well as in the output layer to approximate the unit excitation probability

$$\sigma(z) = \frac{1}{1+e^{-1}} \tag{2.2}$$

Hyperbolic tangent has a similar shape as the sigmoid function shape, only itsrangeis(-1,1)

$$\tanh(z) = \frac{1-e^{-2z}}{1+e^{-2z}} \tag{2.3}$$

Rectifed linear function (ReLu) is a common hidden layer activation function in contemporary neural networks. It was published in 2000 by Hahn loser et al. [19]. It is widely used in computer vision and speech recognition deep learning applications. There are many theoretical variants: LeakyReLu penalises negative input and SoftPlus that approximate ReLu function by a smooth curve, though these functions are not very common in practice. Such as

$$ReLu(z) = \max\{0, z\}$$
$$LakeRelu(z) = \max\{0{,}0.01z\} \tag{2.4}$$
$$SoftPlus(z) = \log(1+e^{z})$$

2.2 LEARNING PROCESS

According to the task, we can distinguish two types of learning processes supervised and unsupervised learning. In supervised learning, the network is trained on a dataset of input/output pairs. In unsupervised learning the network is trained to recognize some clusters or patterns in the dataset without knowing the expected output. This text focuses just on supervised learning. We are going to train our network on a labelled dataset.

Before the start of learning, available labelled data has to be split into several subsets, which allow us to control the learning process. We distinguish three of thesubsets.The training data set is used for the model training. It contains the most of the labelled data. The dataset size depends on the number of available samples.The second sub set is called validation dataset and it is used for controlling the model behavior during the learning and for hyper parameters optimization. The third subset called testing dataset isused for the model evaluation. All the datasets should contain the same distribution of the training samples.

2.3.1 Error Function

In the next part, we will define two key terms of the learning: loss function and cost function. They are the functions that measure the model performance. Both functions describe a quality of the model. The loss function measure show the model performs on just one training sample. The cost function measures an ability of the network to represent well the whole training dataset. To explain these two terms better, let us supposed to have a training dataset $D = \{(\mathbf{x}(1), \mathbf{y}(1)), (\mathbf{x}(2), \mathbf{y}(2)), \dots, (\mathbf{x}(m), \mathbf{y}(m))\}$ containing $m \in \mathbb{N}$ training samples, where the pair $(\mathbf{x}(i), \mathbf{y}(i)) \in D$, $i \in \{1, 2, \dots, m\}$ is an input and expected output of the i-th training sample. We are going to train our model on this dataset. The model is represented by a function $f(\mathbf{x})$.

For a given model f ($\mathbf{x}$) and a training sample ($\mathbf{x}$, $\mathbf{y}$), the loss function Emeasures a distance between the vector f ($\mathbf{x}$) = $\hat{\mathbf{y}}$ = ($\hat{y1}$, $\hat{y2}$,…, $\hat{yn}$)and an expected output $\mathbf{y}$ = (y1, y2, … , yn). In the case of two distant

Vectors, a proper loss function returns a high value. If the vectors are similar or almost the same, the loss function returns a value close to the zero. There are several common variants of per-example loss functions.

The most important ones are mean squared error, mean absolute error and categorical cross-entropy. Their formulas are listed in the Table 2.1.

Table 2.1: Loss functions

Name	Formula
Mean squared error	$E_{mes}(\grave{y},y)= \frac{1}{2} \sum_I (y_i-\grave{y}_i)^2$
Mean absolute error	$Emae (\grave{y},y) = \sum_I \backslash y_i - \grave{y}_i \backslash$
Categorical cross- entropy	$E_{cce} (\grave{y},y) = - \sum_I y_i \log \grave{y}_i$

Table 2.2: Regularization terms

Name	Formula
Lasso regularization	$L_1(\theta)= \lambda \sum_i \lvert\theta_i\rvert$
Ridgeregularization	$L_2(\theta)= \lambda \sum_i \theta_i^2$
ElasticNet	$L(\theta)= \lambda \sum_i(\alpha \cdot \lvert\theta_i\rvert + (1-\alpha) \cdot \theta_i^2)$

The regularization is a machine learning technique adapted to neural networks, which controls the ability of the model to generalize. Not all the input vector values are equally important for the correct behavior of the model. If we set the high weight to some of the unimportant ones, it can the performance of the previously unseen data. For that reason, the all weights should be as small as possible. The regularization term penalizes high model weights and thus reduces the weight of all the unnecessary inputs. Commonly used regularization terms are called Lasso regularization, Ridge regularization or Elastic-net regularization and they are listed in the table 2.2. A hyper parameter gives the weight of the penalization. The regularization can reduce the model variance and thus prevent overfitting, which we will discuss later.

The Cost function is a function (θ) used to measure a quality of the vector θ to represent given dataset. The value of the function is close to zero if the model is performing well and it grows up in case of some errors. Cost function consists of the sum of averaged losses of all the m samples in the dataset and the regularisation term (θ).

$$J(\emptyset) = \frac{1}{m}\sum_{i=1}^{m} E(y^i, y^i) + L(\emptyset) \tag{2.5}$$

2.3.2 Learning Algorithm

A learning algorithm defines the process of learning. For each machine learning task, there is an unknown mapping $f*(\mathbf{x})$ that optimally solves the given task. The learning process aim is to approximate $f^*$by the function $f(\mathbf{x})$, which is represented by our network. This function is achieved by a proper setting of the vector of learnable parameters θ.It can be done explicitly using some prior knowledge, but it is not a suitable approach for challenging tasks. Another way is to compute appropriate algorithmically.

The principle of the learning process is to minimize the cost function by modifying vectorθ. Learning algorithm work site natively. Each learning step (epoch) has a simple schema. In the step number $t \in \mathbb{N}$, the algorithm computes the cost function$(\theta^{[t]})$ using the actual parametersθ. It computes updates$\Delta^{[t]}$ and$\Delta b_j^{[t]}$ for all the learnable parameters and defines new vector$\theta^{[t+1]}$.At time t, the new weights and biases values are computed according to the equations

$$W_i^{[t+1]} = W_i^{[t]} + \Delta W_I^{[T]}$$
$$b_j^{[t+1]} = b_j^{[t]} + \Delta b_j^{[t]} \tag{2.6}$$

The learning algorithm stops after the given number of iterations. Initial values of all the weights should be set by a random number close to zero. If two neurons in one layer have the same weight values at the beginning, their behavior will remain the same through the whole learning process, and they will not provide any extra generalization power [20].

The Gradient descent is an optimization method defining the update value $\Delta^{[t]}$ and $\Delta bj^{[t]}$ in a time $t$as a negative partial derivative of the cost function with respect to the wior bj

$$\Delta W_i^{[t]} = -\epsilon \frac{\partial J(\theta^{[t]})}{\partial W_i}$$

$$\Delta b_j^{[t]} = -\epsilon \frac{\partial J(\theta^{[t]})}{\partial b_j} \tag{2.7}$$

where $\epsilon$is a hyperparameter called learning rate that controls an update magnitude. The method is based on the observation that any multi-variable function $g$in a point $a$decreases the fastest in the direction of the negative gradient of $g$at a. Gradient descent is the most common optimization method used in deep learning and it works well in various tasks. Let mention also two drawbacks of this method. First, the minimum size of a step is not guaranteed, so the descent can be unlimitedly slow. Second, the optimization can get stuck in the local optimum. The term backpropagation denotes a schema of how all the gradients in multi-layer networks are computed. The process starts in the output layer. Then the error is propagated in a backward direction through the whole network. In the end, all learnable parameters update. Gradients of weights in one layer are computed in parallel. The back propagation schema reuses already calculated inner potentials of all the neurons. Every gradient should be computed from the wholeavailable dataset.

On the other hand, a common dataset contains thousands of samples. It would be computationally demanding to do just one learning step and progress would be too slow. In the Mini-Batch Gradient m Descent approach, the cost is computed from randomly chosen dataset subsets called mini-batches. It updates weights more often, and the learning algorithm converges faster. A typical size of one mini-batch is about 32 to 128 samples. It is limited also by the available hardware memory. The mini-batch approach introduces a random noise to the data. It leads to a better generalization of the model and prevents overfitting. The mini-batch size is one of the hyper parameters.

2.3.3 Network Tuning

In this section, we are going to discuss methods that can be used measure if he model is well trained. We focus on techniques used.Apart of this section is dedicated to hyper parameter setting.

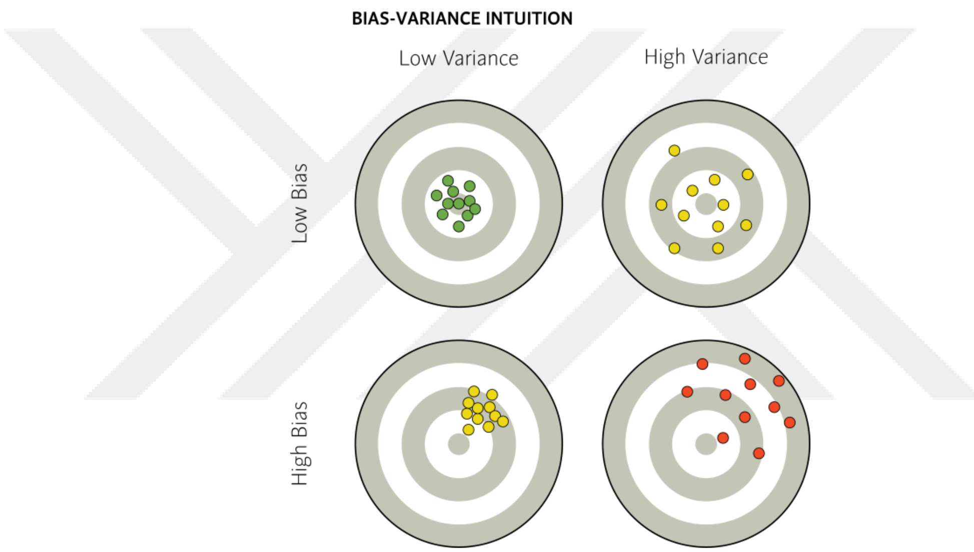

Figure 2.3: Bias and variance intuition

During the designing and training a new network. A part of this section is dedicated to hyper parameter setting.are illustrated in the Figure2.3.

The bias-variance trade-off offers an important view on the model error rate. We can identify two main sources of the generalization error: the error due to bias and the error due to variance. There is a trade-off between them, and it is crucial to differ which one struggles with the model. Both are illustrated in the Figure 2.3.
The bias is an error caused by lack of training samples that are relevant for the given task. We can also say, that the dataset does not represent the task well. In this case, providing more learning data from the same distribution does not improve accuracy. It is necessary to improve the training dataset and learn the model from the new data which is closer to

unseen ones. A similar problem is caused by a model that is not capable enough to solve the task even on the training dataset. It misses the target because it is unable to perform the learned function. This kind of error is also called model under fitting.

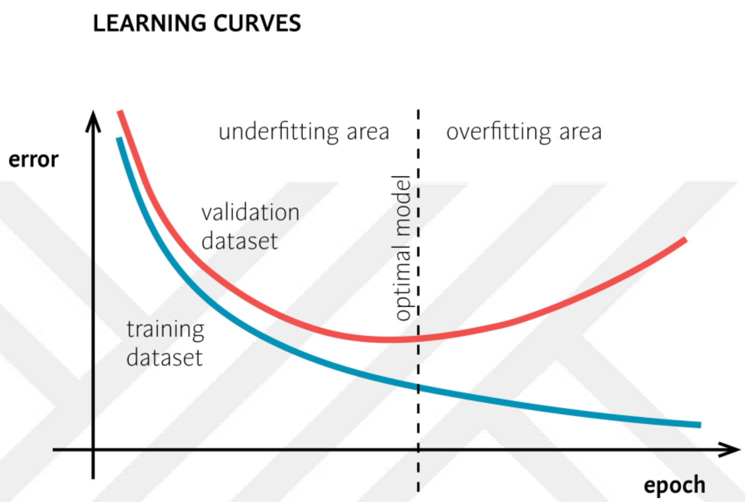

Figure 2.4: Learning curves

A high variability of predictions causes the variance error . Neural networks are highly capable of representing complex functions. If we learn from a small dataset, the network will be able to remember all the training samples, and it will not generalize the knowledge well. This model performs well on a training dataset, but much worse on a testing dataset. This problem is called is called overfitting. There are several possibilities how to improve the generalisation of the model with a high variance. The first one is to add some restrictions to the network. For example, we can increase the weight of the regularisation term. Another way is to train on a bigger dataset or stop training earlier to avoid increasing the variance too much.

The learning curve is used to control the process of training. It is a simple plot of errors in training and validation dataset during different learning epochs. Using the learning curve plot, we can recognise several network training properties. First, we can check whether the training error is decreasing during the training. Second, it helps us to determine the error that the model suffers from, whether it is due to the high bias or due to the high variance. If the training error goes to zero quickly and the validation error is high, then the model will be overfitting and will suffer from the high variance. When the training error is as high as the validation one, then the model is biased. Third, we can deduce a right time for stopping the learning. The Figure 1.4 shows the typical shape of learning curves.

Hyperparameters are variables that set up a training process. In the text,we already mentioned the learning rate, mini-batch size and regularisation oefficient . Other hyperparameters are momentum and optimiser settings. By proper hyperparameter setting, we can significantly improve the trained model. We can speed up the learning or avoid the bias and variance errors.

These parameters do not have a generally correct value. The values are task specific. There are several approaches of how to find proper values. The basic hyperparameters optimization methods are grid search and random search. Both of them test several different settings and compare them to each other. Hyperparameters can be changed during the training. For example, it is common to reduce the learning rate to reach the optimum more precisely in later epochs.

3. DEEP LEARNING AND IMAGESEGMENTATION

This chapter connects neural networks with image processing. It introduces the specific class of neural networks called convolutional neural networks. It contains a definition of a segmentation task, and it shows how the task can be solved using this neural network type.

3.1 CONVOLUTIONAL NEURAL NETWORKS

Convolutional neural networks (CNN) is the class of feed forward neural networks designed for image processing. It was first described in 1989 by LeCun [21] who originally developed it for a hand-written digits recognition. CNN was one of the first neural networks used in commercial applications. Already in 1990, AT&T used a convolutional network in a real bank system that reads handwritten digits in checks [22]. Nowadays, they are used in a majority of computer vision tasks. They are capable enough to solve a wide range of object recognition problems as well as some more difficult tasks. [reference] The advantage of CNN is that it can be applied onto structured data. The input of the network is a multi-dimensional array. A two dimensional array represents a grayscale input image where each number describes one pixel. A three-dimensional array represents a colour image where a vector of colour values describes each pixel. A convolutional neural network can also be trained on one-dimensional sequences, e.g., written text or sound signal. The neurons in a convolutional network are arranged in a grid, and they can take into account relationship between neighbouring data samples.

In this section is an introduction to convolutional neural networks. We explain the term convolution as well as the typical topology of a convolutional neural network. There are examples of successful CNN architectures illustrating important milestones in this field.

3.1.1 Convolution Operation

Convolutional neural networks are named after a linear mathematical operation called convolution. In the image processing, convolution is an operation of two matrices(1*k)where is an input vector and K. is a kernel (convolutional matrix), which product is also a matrix. For each matrix position, the result value of this operation is a weighted sum of input matrix values in a small neighborhood. The kernel defines weights and size of the neighborhood. Let's define convolution for two-dimensional data. The result value at the position is computed by the following formula

$$z_{ij} = (I * K)_{IJ} = \sum_m \sum_n I(m,n).K(i-m, j-n) \qquad (3.1)$$

Where I(m,n) , K(m,n) denotes the pixel value at the position[m,n]. In the formula we assume that both the input image and the kernel are infinite. In practise, we can imagine a finite image surrounded by zeros.

In the context of convolutional neural networks, the same operation is used to compute neuron 'sinner potential. The kernel is a weight matrix of an odd size. The common kernel sizes are from 3x3 to 9x9 pixels. Compared to the convolutional matrix used in the image processing, CNN kernel is fully learnable. It can adapt to patterns in training data. The convolution is used as a pattern detector. Different kernels define different patterns. They are recognized in several levels. Each network layer represents one level of detection. The early layers detect simple patterns, e.g., edges in different orientations, shades or gradients. More complex patterns are recognized in later layers as acombination of simple ones [23].

3.1.2 Convolutional Layer

The convolutional layer is the main part of a CNNs architecture. It is a layer of neurons, which compute their inner potential by a convolution. The schema of the layer is shown in Figure 3.1. Neurons in a convolutional layer have a fixed position in a grid. The grid of these neurons is called eature map. Weights of their connection are given by a kernel. There can be several feature maps in one convolutional layer that are performing in parallel on the same data. Each feature map uses its kernel.The convolution layer can be modified by several settings, which influence the layer behaviour and feature maps size. The most common settings are called stride and padding. The stride is a number,

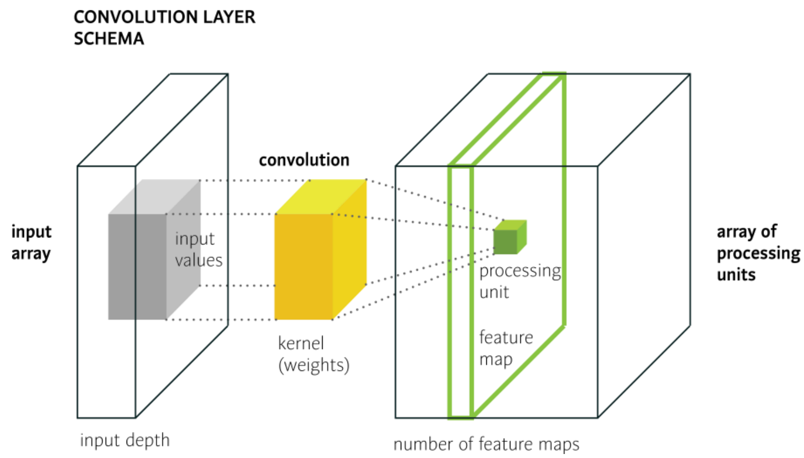

Figure 3.1: Convolutional layer schema

Defines the distance between two closest kernel positions. If stride equals to one, convolution will be computed for every position. If stride equals to two, the result is computed only in every second location. Thus a feature map is down-sampled by two. The padding sets computation in border positions. Convolution cannot be computed for pixels where the neighborhood is outside the image domain. The possible solution is to add a zero-padding around the input image. In convolutional neural network terminology, it is called same padding. To not contaminate calculation by zeros, we can compute convolution only at valid locations. Then the feature map size is reduced by $2k - 1$ pixels where **k** is a kernel dimension. This setting is called valid padding. There are two main reasons why the convolutional networks perform better on image data than MLP.

The sparse connectivity relates to the fact that the kernel dimension is smaller than the image size. An input can have millions of pixels in total, but a standard kernel has at most hundred values. We can produce a feature map with the same dimension as the input using only a few weights. If we want to produce an output of the same size as input by MLP, the number of weights is square of the input size. The convolution layer significantly reduces the memory requirements as well as runtime of learning. It allows processing images in a high resolution without any down-sampling.

The parameter sharing describes the fact that one kernel is used several times at different image positions. For example, if the kernel is trained to recognize a vertical edge, it will recognise it at all other picture locations. It also means that every kernel is learned in all of these locations and thus on more data than the number of training samples. It increases the probability that the kernel is learned well.

3.1.3 Pooling Layer

A pooling layer is a type of neural network layer, which has no learn- able parameters. It down-samples the input by performing the given statistics of the nearby units, e.g., maximum, mean. The pooling layers are then called max pooling or average pooling. The pooling layer makes network performance invariant to the small translations. For example, in face detection, it can eliminate small differences in face movements. Together with a convolutional layer, they are basic components of convolutional neural networks.

3.1.4 Case Studies

In this subsection, we would like to introduce some known examples of contemporary convolutional networks shortly. They all took part in the Large Scale Visual Recognition Competition (ILSVRC) [24], which has held since 2010. It is one of the most important challenges in this field. The competition task is to classify digital photos into thousands of different classes. All the introduced architectures remain in use in another computer vision tasks.

This network won ILSVRC competition in 2012 [17]. It was the first network, which has shown that deep learning can compete with traditional image analysis methods.

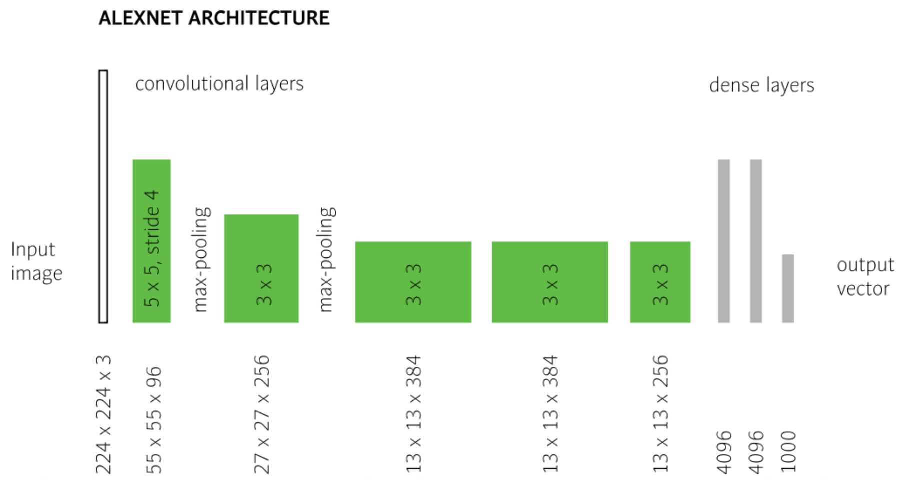

Figure 3.2: The architecture of AlexNet

The network has a quite simple architecture, shown in Figure 3.2. There are five convolutional layers with max-pooling followed by three fully connected layers. The combination of convolutional layers and pooling layers has been typical for CNNs until today.

This network was developed by Google [25], and it won ILSVRC-2014 with accuracy 93.3%. This network architecture is characteristic of high complexity. It introduced new building block called Inception Module, which is shown in Figure 3.3. The module consists of several small convolutional networks, which are computed in parallel. Due to sharing some computational phases, the computation is effective. The final answer of the module is a concatenation of the outputs. The authors demonstrated that this approach has great results. Inception module allows recognizing a complex features without increasing the network depth.

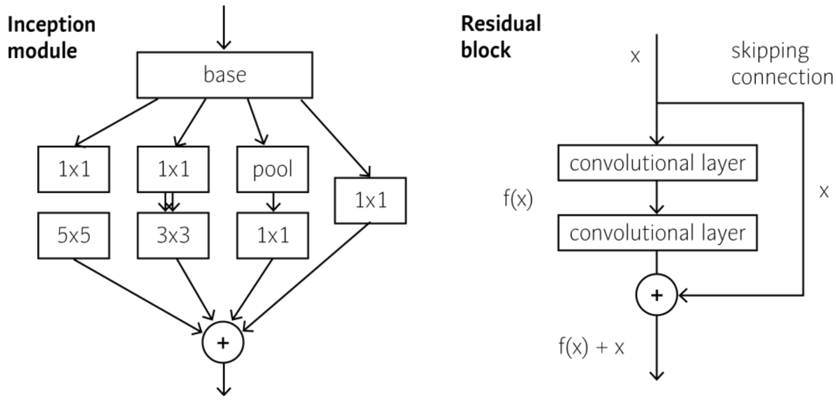

Figure 3.3: Inception module and Residual block schema

The ResNet has succeeded in ILSVRC in 2016 with 96.4% accuracy. It is a product of Microsoft [26], and it has 152 layers stacked in residual blocks. The Figure 2.3 shows the residual block schema. This block is a group of layers bridged by a skipping connection. This connection helps to the layers of the block to learn an identity function. Thus layers in the block can be skipped if they do not offer any new information. This approach allows training very deep networks. It prevents some problems that are typical for deep networks, e.g., high variance and vanishing gradient.

3.2 IMAGE SEGMENTATION

Image segmentation is one of the key problems in the field of computer vision. It is one of the steps that lead to an automatic image understanding. We are going to present how it can be solved using convolutional neural networks. At the end of the section, there is a detailed description of the network called U-Net. This network architecture represents the state-of-the-art on biomedical images and also inspired our research.

3.2.1 Computer Vision Tasks

Image segmentation is not an isolated task in the field of computer vision. It is one of the several tasks that leads to automatic image understanding. Let’s describe some of the basic ones.

In the beginning, there is image classification that can be described as an assigning the image to the class. In this context, classes differ images according to the content. The goal of object detection task is to recognise if the given object is in the picture. A task object localisation provides additional information about spatial location of the recognized object. A centroid or a bounding box can define the object location.

Image segmentation is another branch of computer vision tasks. The segmentation aim is to recognise all the given classes in the image at the pixel level. It can be described as a pixel-wise classification. Segmentation mask defines all the pixels of one class. The combination of object detection and segmentation is called instance segmentation that recognises different objects from the same class and provide their segmentation masks. To further refine an image hierarchical segmentation can be used. It analyses the input image in several scales in parallel.

Let's define the term segmentation. Let is Ω the set of all input image pixels and $\mathbb{P} = \{1, 2, \dots, k\}$, $k \in \mathbb{N}$ is a set of recognized classes. Segmentation is a mapping $S : \Omega \rightarrow \mathbb{P}$ that assigns to each pixel exactly one value from the set $\mathbf{P}$. The output of the segmentation can be a labelled image where each pixel has a value according to the class it represents. In the case of binary segmentation, the output is a binary mask.

Image segmentation is a useful tool in various fields of study including the medical research. In this field, the precision of used methods is essential and human resources are very expensive. Modern medical imagining techniques lead to the growth of image data being captured, and automation can help to process them more effectively. Examples of segmentation task in biomedical field are mitosis detection [27] or tracking neuronal processes in EM images [28].

3.2.2 CNN Architectures for Image Segmentation

There is no direct way to use the CNN for image segmentation. Typical usage of CNNs is the image classification task. There are several approaches how to modify the network to be suitable also for segmentation. To the successful ones belong sliding window, fully convolutional network and SegNet.

One of the first methods using CNNs for segmentation was the sliding window. Each pixel is classified by evaluating small cropped area that surrounds the pixel [29]. It is not easy to correctly determine the size of the cropped part. There is a trade-off between the computational requirements and amount of contextual information. If the cropped area is too large, the computation takes to much time. If the area is too small, it does not contain enough information. Compared to the current methods, the method of the sliding window is computationally expensive and inefficient.

The Fully convolutional network (FCN) introduces an end-to-end convolutional network architecture for segmentation [30]. There are no pooling layers is this architecture, so the network output has the same dimension as an input. The model produces the labelled image directly. In the last layer, there is a 1x1 convolution, that sets to each pixel the vector for one-hot classification.
End-to-end learning is a successful concept for building a neural network for segmentation. The advantage of this approach is the simplicity of the model. One method replaces entire segmentation pipeline. More computation steps lead to a higher probability of an error, and it amplifies. On the other hand, the end-to-end architecture is suitable only if there is enough data to train it properly.
A team of V. Badrinarayanan[31] introduced this architecture to reduce the computational costs of FCN and to improve the network ability of generalization. It belongs to a class of the end-to-end architectures. It uses a type of neural network layer called up-sampling layer that provides a reverse operation to the pooling. There are no learnable variables in this layer, and it can increase the feature map dimension. The SegNet topology recalls the topology of a neural network type called autoencoder. SegNet consists of two symmetric parts, an encoder and a decoder. The encoder part uses a standard CNN layer structure. There are convolution layers followed by pooling layers. In the decoder part, convolutional layers are interlaced by upsampling layers to increase a feature maps dimensions up to the size of the original image.

3.2.3 U-Net

The U-Net network was developed in 2015 by Olaf Ronnenberger, Phillip Fischer and Thomas Brox at the University of Freiburg. The au- thors designed the network for semantic segmentation of biomedical images. In that time, it was ranked the best segmentation method for the most challenging datasets in the ISBI Cell Tracking Challenge [32]. It performs well in other ISBI segmentation challenges, too.

The primary motivation of Olaf Ronnenberger research group was to improve the sliding window approach by a making it simpler and more effective. They developed a robust network that involves the whole context of the image in the computation and directly produces the segmentation masks.

The network has the SegNet architecture with several improvements. Figure 2.4 shows the architecture. In the encoder part, there are four down-sampling steps. At each step, there are two convolution layers with 3 × 3 kernel and ReLu activation, followed by 2 × 2 max-pooling layer. The number of feature maps in the first layer is 64, and it doubles after every down-sampling step. The decoder topology is symmetrical to the encoder. There are four up-sampling steps; each contains two convolutional layers. After every up-sampling step, the number of feature maps is halved. There are also the skipping connections between corresponding encoder and decoder steps. The output layer uses 1 x1 convolution, which for every pixel reduces information from all the feature maps into a two-element vector.

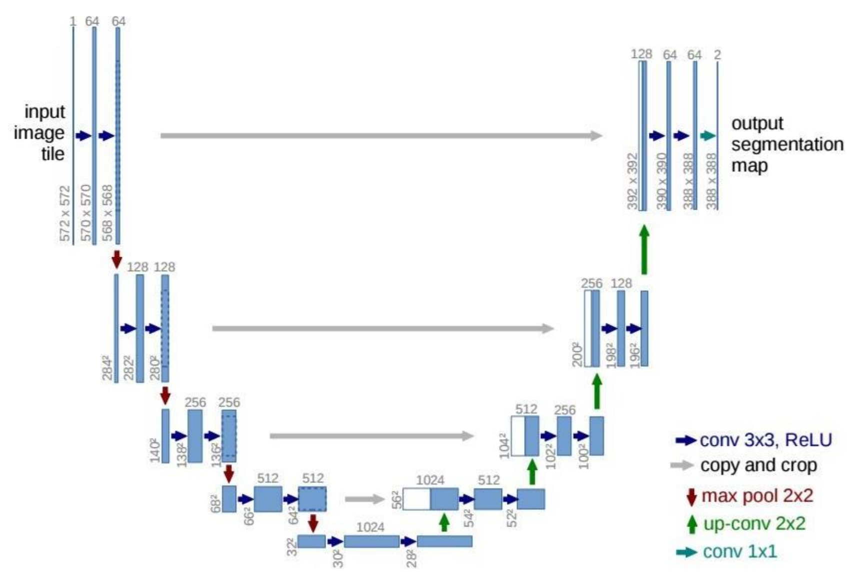

Figure 3.4: U-Net topology

The U-Net is possible to train also on the datasets with only a few available training samples. In the original article, the authors describe some augmentation methods on how to increase the dataset size. New images are generated by shifts and rotations. They also used elastic deformation of original samples to simulate the shape variability of biomedical objects. The deformation was made by a random displacement of vectors in 3x3 grid sampled from a Gaussian distribution with standard deviation of ten pixels. They also mention the problem of grey value variations of the input images.

The cost function is pixel-wise soft-max function combined with the cross-entropy loss function.To improve the network performance, authors supposed that pixels have different weights according to their importance in the final segmentation. Here is the formula of the cost function

$$J(\emptyset) = -\sum_x w(x)\log\left(|1 - p_l\,(x;\emptyset)|\right), \tag{3.2}$$

.

Where (x) is a weight of the pixel x, $l$is the expected class of the pixel $x$and $p(x;\,\theta)$ is the probability given by the model that $x$belongs to the class l. The weight of each pixel was pre-computed by the following formula:

$$w(x) = w_c\,(x) + w.\,exp\left(-\ \frac{(d_1(x)+d_2(x))^2}{2\sigma^2}\right. \tag{3.3}$$

Where d_1 and d_2 denotes the distance of the pixel x to the first and second closest object, respectively. The variables $w0$ and $\sigma$are constants The original network was implemented in Caffe software. All the computations were done on a graphical card. Due to the big size of the image samples and hardware limitations, it was not possible to train the mini-batch of size higher than one. As a learning algorithm, they used stochastic gradient descent with a momentum.

U-Net is successful in the segmentation of various biomedical images. It still occupies the first places in the ranking of the Cell Tracking Challenge. The method is clear to understand. In recent last years, several another approaches inspired by the U-Net topology was published. For these reasons, we decided to base our research on this network.

4. PRACTICALPART

This chapter presents step-by-step the process of a training a convolutional neural network suitable for a semantic segmentation task. We focus on the tasks given by the ISBI Cell Tracking Challenge. This com- petition provides the background for the evaluation of our methods, too.

One of this work goals is to get familiar with the technique, which is current state-of-the-art for the biomedical segmentation. We chose one of the challenging datasets to prove our ability to control this method. This dataset defines many of decisions that we made during the work. The crucial part of the whole work is a proper preparation of the dataset to be sufficient for convolutional neural network training. Another important part was a designing of the appropriate neural network model. We trained two different models to measure the differ- ence between their performance. The best model has been compared to other competitors in the Cell Tracking Challenge.

4.1 PREPAATION OF DATASET

The organizers of Cell Tracking Challenge created 13 datasets of dif- ferent cell lines captured in different modalities [5]. The photos were taken by several research groups using different imaging techniques, e.g., phase contrast microscopy (PhC), differential interference contrast microscopy (DIC), and fluorescence microscopy. Two of datasets were simulated. Every dataset consists of training and challenging time-lapse video sequences. For every sequence, the authors created unified segmentation and tracking ground truth (GT). The training dataset ground truth is public. The challenge datasets ground truth is private due to the competition. For our experiments, we chose one of the most challenging datasets of the Cell Tracking Challenge [6] called DIC-C2DH-HeLa. The imaged cell line can be characterized as a dense cell population because clusters of touching cells have often different size, textural and intensity characteristics. An example of one dataset image is in Figure 4.1.

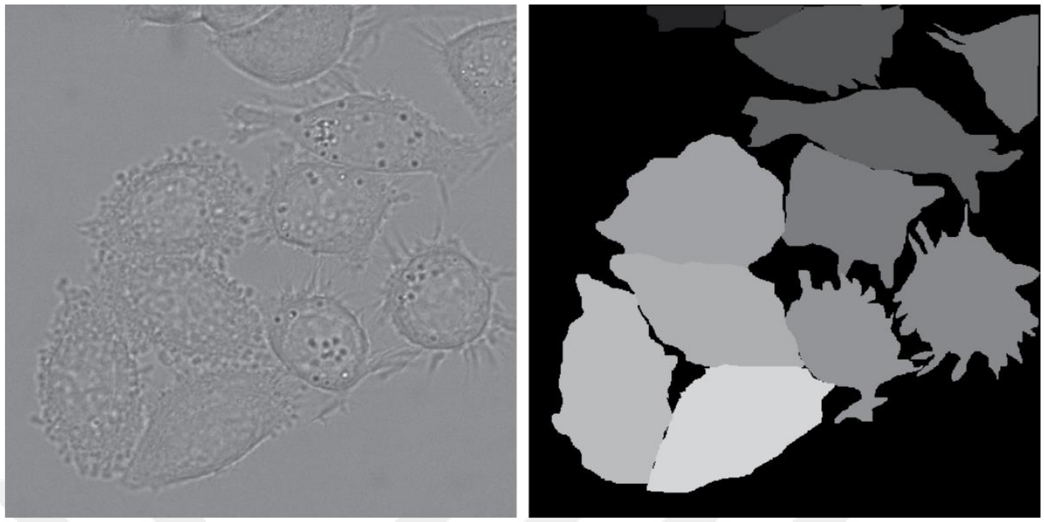

Figure 4.1: An example of a dataset image

4.1.1 DIC-C2DH-HeLaDataset

These dataset images were captured in 2010 by Dr Gert van Cappellen in Erasmus Medical Center, Rotterdam. Using differential interference contrast microscopy, it represents a HeLa cell population. HeLa is a cell line derived from cervical cancer cells. It is often used for basic cell biology research because this cell line is easy to cultivate.

The dataset consists of four different time-lapse video sequences, which are compared in the Table 4.1. Two training video sequences $S1$ and S2 contain 85 frames each. Two challenge video sequences S3and S4 contain 115 frames each. In our work, we used the segmentation ground truth. The original dataset offers only 9 fully segmented GT frames in each training sequence. For that reason, we asked an expert to manually segment the whole S1 sequence and used these segmentations for the training, too. Figure 4.1 compares original image and its ground true. The frames were taken in 10 minutes intervals. One frames represented by a grayscale 512 x512 image stored in 8-bit tiff format. The photos given by this dataset have several challenging properties. The cells on the images are very tightly packed, and the borders between the cells are hard to find due to complex intensity patterns

Table 4.1: DIC-C2DH-HeLa dataset size

Sequence	training		challenge	
	S1	S2	S3	S4
number of images	85	85	115	115
size of ground truth	9	9	-	-
manual segmentations	85	-	-	-

Inside the cells. Furthermore, the boundary between the cells is simi- lar to the boundary between the nucleus and the cytoplasm. Another problem is that the cells change their appearance while they are dividing. Two drawbacks are also given by a structure of the dataset. First, it does not contain enough manually segmented images to train a CNN properly. A typical neural network image dataset contains tens of thousands labelled samples. Secondly, most of the manually segmented frames in the dataset comes from the same sequence, so these frames describe only a limited set of possible features. There is a danger of model overfitting to these features.

4.1.2 Data Augmentation

Data augmentation is a procedure often used to improve the neural network performance if the training dataset is too small. The goal is to generate new data, which is from the same probability distribution as testing samples. Moreover, this procedure is used to extend a set of features in a training dataset by generating samples with new features. Most augmentation methods are based on providing random modifications of existing samples.

There are several image transformation methods used to augment training datasets. We need to choose the suitable ones for the given task. Because cells have no fixed orientation, and they vary in shape, and at the same time the convolution is not equivariant to the basic geometric transformations such as rotation, mirroring and change of the size we decided to enrich our training dataset by all these transformations. We also used a technique published in 2003 by Simard [34] called random distortion. It can simulate variations in the cell shape. A reflection approximates missing data outside the image domain.

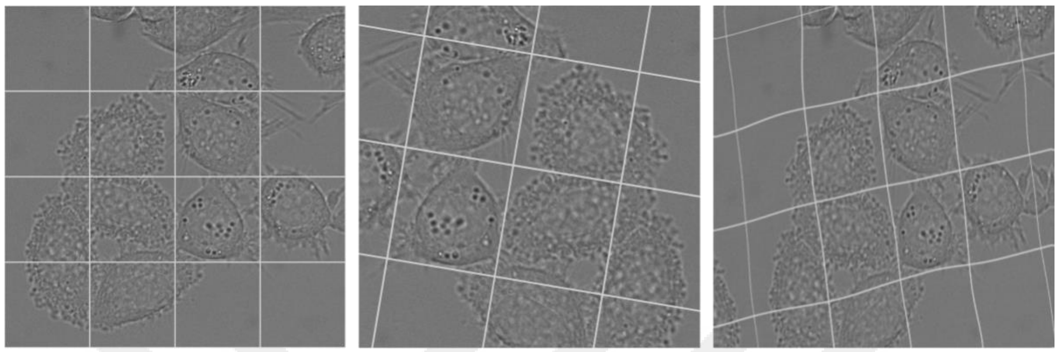

Figure 4.2: Transformations used for the dataset augmentation

Examples of transformed images are in Figure 4.2.

The training data can be augmented before the training or on the fly. An advantage of the first approach is that it is easier to control all the samples used for training. The second approach allows us to use a wider range of augmenting methods to generate more different samples, and it is not necessary to store new samples. We have decided to combine both . Before the training, we augment training dataset by 40 distorted copies of every training sample. The geometric transformations were applied on the fly.

We split training data to training dataset and validation dataset. The training dataset contains all the images from the first training sequence. The ground truth images from the second training sequence form the validation set. (Table 4.2) We need to use as much data as possible for training. Moreover, if we contaminated the validation dataset by images similar to the training samples, we would lose the precision of the validation. Because training and validation dataset is from the different sequences, our training dataset can be highly biased

Table 4.2: Training dataset size

Dataset	S1	$S1_{aug}$	S2	$S2_{aug}$	total size
training dataset	85	3400	-	-	3485
validation dataset	-	-	9	360	369

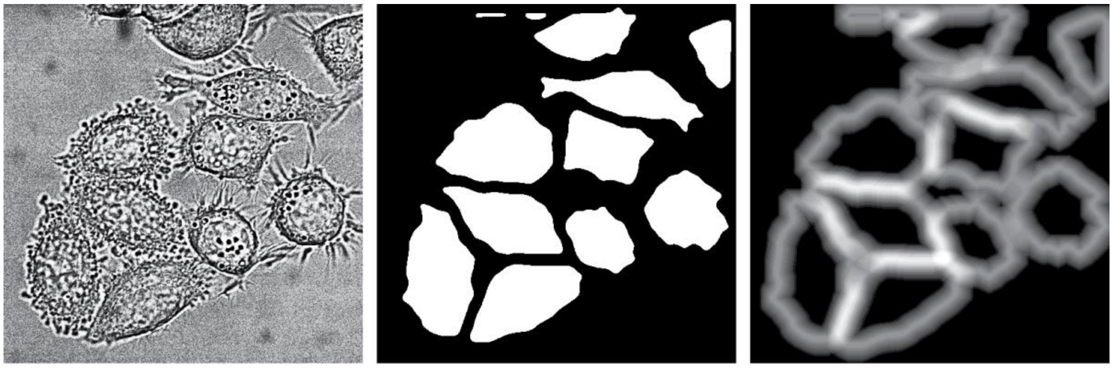

Figure 4.3: Image pre-processing

4.1.3 Image Pre-Processing

Neither input images nor their manually segmented masks have the right format to be put directly into the network. It is also necessary to extract some additional information from the samples to improve learning. For that reason, we pre-process all the images to be more suitable for training.

The input images are taken in different light conditions. We noticed that samples with a different illumination level have on average a higher error than the others. Therefore we decided to normalise all input images to the same illumination level. For the normalization, we used a histogram normalisation method called Contrast Limited Adaptive Histogram Equalization [35].

Next, we modified ground truth images. The used neural network is not capable of producing directly labelled images. The neural net- work we are going to use is designed to decide whether the given pixel represents a cell or the background. Therefore we changed the labelled images to binary masks. The cell masks can touch each other in general. Therefore we created gaps between them in order to separate them. A wider gap increases the chance of an accurate cell detection in the case of incorrectly classified pixels in the gap. Before the binarization, all the labelled regions were eroded by a 5x5 elliptic kernel. We removed all disconnected objects and smoothed the mask's borders. An example of the training mask is in Figure 4.3. The following pseudo-code shows how the training masks were created:

```
## training mask##
# input - manually_segmented_image
# ouput- binary mask for training CNN

man_seg := manually
segmented_imagefinal_mask:= empty image
kernel:=ellipseofsize5x5px

for every object in man_seg do:

    # erode and remove small  areas object
    := erode object by thekernel
    object := opening object by the kernel

    # smooth the new mask
    objectUp:=up_sample object for
    ifrom 1 to 10 do:
        objectUp := blur objectUp by the kernel object
    := down_sampleobjectUp
    object:=thresholdobjectby130

    # add to the final  maskfinal_mask :=
    final_mask +bject
returnfinal_mask
```

In the next section, we will define a network cost function, which takes into account the sum of distances between the given pixel and all the objects on the training mask. Let define the pixel *weight*, which can be pre-computed for all the training samples The goal is to highlight all the pixels close to object borders. The weight of pixel $\boldsymbol{i} \in \mathrm{N}$ is given by the following formula:

$$W_i = a \sum_0 \max\,(b - c.\,dist(p_i\ , 0), 0) \tag{4.1}$$

where $dist(pi,\ o)$ denotes a distance between the object $o$and the pixel pi, parameter c∈$\boldsymbol{R}$regulates thickness of a highlighted area and a, b∈$\boldsymbol{R}$range of the weights. The range is set to real values between 1 and 4.

Figure 4.4: Mask post-processing

All the weights are stored in weight mask. An example is shown in Figure 3.3. Pixels with high weight are important for good classification. They define the cell mask shape as well as the gaps between cells. If these pixels are classified correctly, there is a high chance of the correct final segmentation. On our method, weight mask is a necessary part of each training sample.

4.1.4 Masks Post-Processing

Let's move to the situation after the training. The network produces some image data, and we need to transform it into a correct segmentation mask. The network output is not binary, the values of pixels are real numbers in the range from 0 to 1. We have to binarize them to define, which pixel represents a cell and which one represents the background. The threshold value is manually set to 0.7 because we prefer a correct classification of the background. After the thresholding, we refine the masks by opening to remove weak connections between different segments. Next we remove holes in all connected components. The last step is the region labelling. The results are illustrated in Figure 3.4. The following code illustrates the whole process:

```
## mask post-processing##
# input - grayscale_image, values from 0to 1    #
output - labelled image

kernel:=ellipseofsize20x20px
#binarisation
mask := threshold grayscale_imageby0.7

# removing thin connections mask
:= opening by the kernel

# filling holes
mask := fill holes in connectedcomponents

#labelling
labelled_image := label connected components

returnlabelled_image
```

4.2 NETWORK TRAINING

This section describes the process of training the neural network. Our aim is not only to get the best model but also demonstrate some of the network properties. For that reason, we designed two different convolutional neural networks and trained them on the same dataset. The networks have a different number of layers. After the model evaluation, we will be able to conclude the effect of the network depth on the performance.

4.2.1 Model Topology

A network topology defines many of network properties, therefore we compare two different network architectures. Both networks that we called *model A* and *model B* are pictured in Figure 3.5. They are the SegNet types, and they are inspired by the U-Net. The networks consist of an encoder and a decoder part. There are four down-sampling steps in the encoder and four up-sampling steps in the decoder part. The network input and output have size 512x512 pixels. In between the encoder and the decoder, a feature map size is reduced down to32x32. Moreover, the encoder and the decoder are at each sampling step interconnected by a skipping connection.

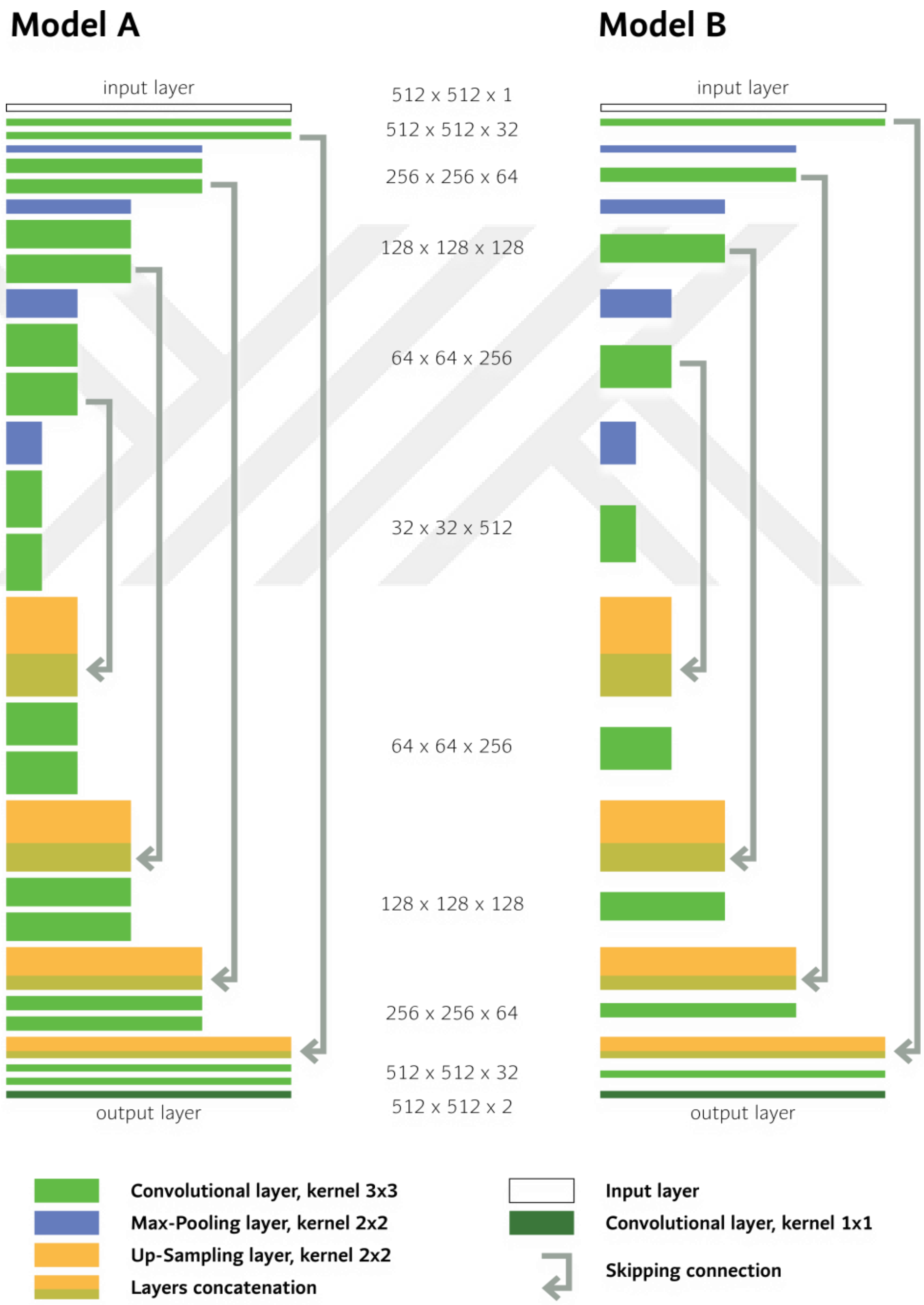

Figure 4.5: The architecture of the trained networks

From convolutional layers with 3 × 3 convolution filters and frommax-pooling and up-sampling layers. The output layer provides 1 × 1 pixel-wise convolution.

These architectures differ in the number of convolutional layers. Model A has the same depth as the U-Net, i.e., 19 convolutional layers.

Model B is smaller and has only 10 convolutional layers. The networks also differ in the number of learning parameters. Model A contains over 7.8 millions learnable parameters; model B contains more than 3.9 millions learnable parameters.

4.2.2 Training

We implemented a loss function called weighted mean square error to refine the provided segmentation by the importance of different pixels. The loss function is described by the formula:

$$\boldsymbol{\omega MSE} = \frac{\sum_I \boldsymbol{\omega}(\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i)^2}{\sum_i \boldsymbol{\omega}_i} \tag{4.2}$$

where $wi$is a weight of the pixel i, variables $y_i$and yidenote the predicted and the expected output of the pixel i, respectively. The formula is directly inspired by the MSE loss function. The normalisation by the sum of all weights is necessary because the number of highlighted pixels is not the same in different samples. The range of $wMSE$is from0 to 1.

The used optimizer is called Adam. It was published in 2014 by D. Kingma and J. Ba [36]. It is one of the most advanced gradient-based optimizers nowadays. It combines several optimization techniques, and it is still computationally efficient even for difficult optimization tasks. It has only three hyper parameters for tuning the performance.

The most important one is the learning rate. The initial learning rate value is set to 0.001. For the training, we used a virtual machine provided by MetaCentrum. Most of the computations were made on a graphics card. We used NVIDIA Tesla K20m with available memory of 4.56 GB. Training process consumes another 12 GB of the computer memory as well as some CPU time. Due to the high number of learnable parameters and hardware limitations, we were able to train on mini-batches of maximum size.

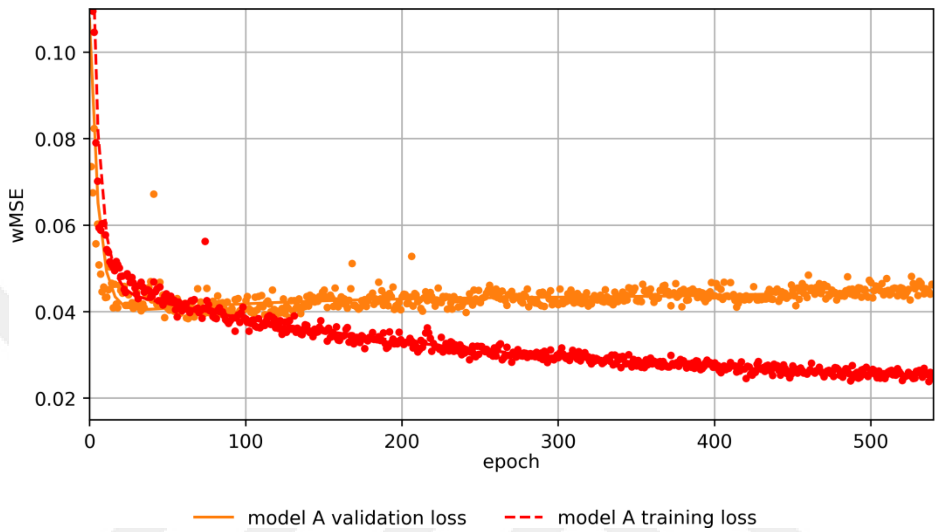

Figure 4.6: Model A learning curves

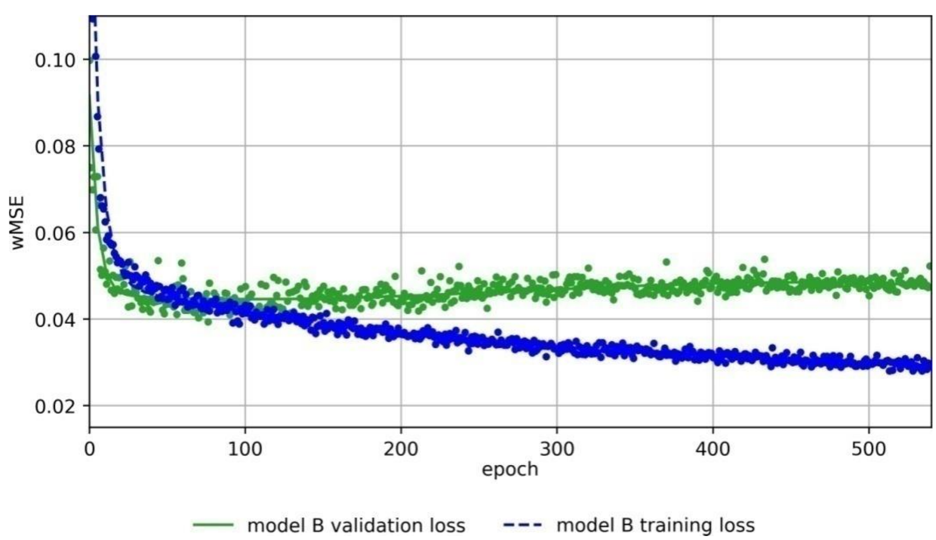

Figure 4.7: Model B learning curves

Table 4.3: Training statistics

	Model A	Model B
avg epoch training time	81s	75s
min training error	0.025	0.028
min validation error	0.040	0.042

We used 50 mini-batches in one epoch. The whole training dataset proceeds in 15 epochs. Every input sample is modified before feeding into the training process due to the on-the-fly augmentation. Both the training loss and the validation loss are recorded in each epoch. We also store the model in every 10th epoch to pick the one with the best performance later. We trained both models for 540 epochs. After 240th and 390th epoch, we reduced the learning rate from 10−3 to 10−4 and 10−5, respectively, to allow a more precise gradient descent. The other learning statistics are in Table 4.3. Learning curves of both models are shown in Figure 4.6 and in Figure 4.7.

4.2.3 Evaluation

For the evaluation, we used a script published by authors of the Cell Tracking Challenge. The challenge sequences a ground truth is not public due to a competition purpose, therefore we were able to com- pare our models just on training sequences. It is necessary to avoid contamination of the evaluation data by training data. Our training dataset contains only images from the first training sequence. Thuswecan evaluate both models on the second training sequence.

AsegmentationmeasureSEG[4]]isbasedontheJaccardsimilarity index. The measure picks several cells from the given images and compares them with the irpredicted masks.The measurement is given by function $J(S, R)$, which is describedas:

$$J(S,R) = \frac{|R \cap S|}{|R \cup \acute{S}|}$$

Where is the set of pixels of the chosen object and is the set of pixels of its segmentation mask predicted by an algorithm. The sets are considered to be matching if and only if the following condition

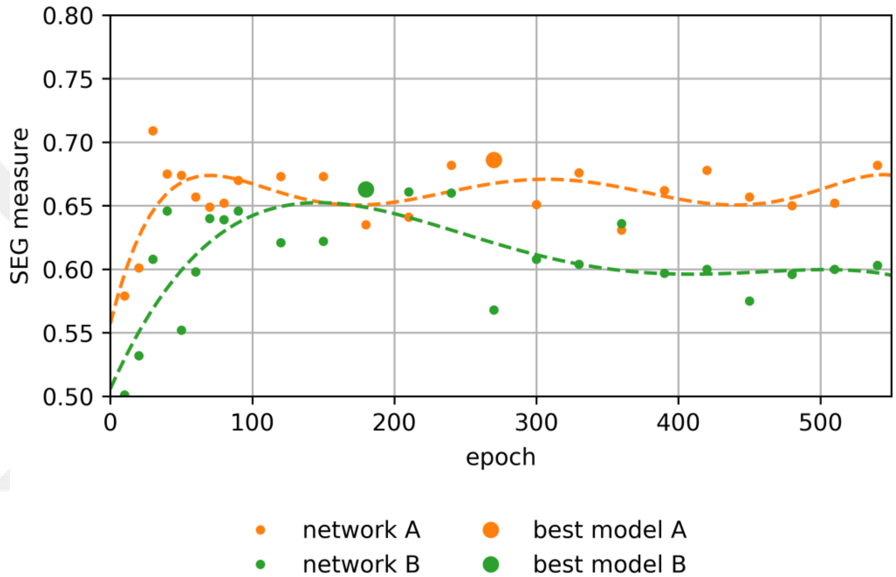

Figure 4.8: Evaluation of the models

Holds:

$$/R \cap S/ > 0.5 \ ./R/$$

We evaluate both models in every learning stage. Results are shown in Figure3.8. The model A performs slightly better than model B. It is expected behavior because A is more complex than B. The performance of both models is not stable during the learning process. It is oscillating in a range(0.63, 0.71) for model A and(0.57,0.67) for model B. The difference between model A and B in SEG measure is approximately0.5 . According to the measurement, we pick the best models. In the case of the model A, we did not pick the highest value of the measure, because it was reached just after 40 epochs and in the context of learning, it seems to be an outlier. We rather choose model A that we reached after 270 epochs with SEG measure equal to 0.685883. The best model B trained 180 epochs, and it is SEG measure equal to 0.662657..

To demonstrate the performance of the trained models, we also offer a visual comparison of the produced segmentation masks. We

To demonstrate the performance of the trained models, we also offer a visual comparison of the picked two images from each challenge sequence and let both models predict their segmentation masks. They are all shown in Figure 4.9.

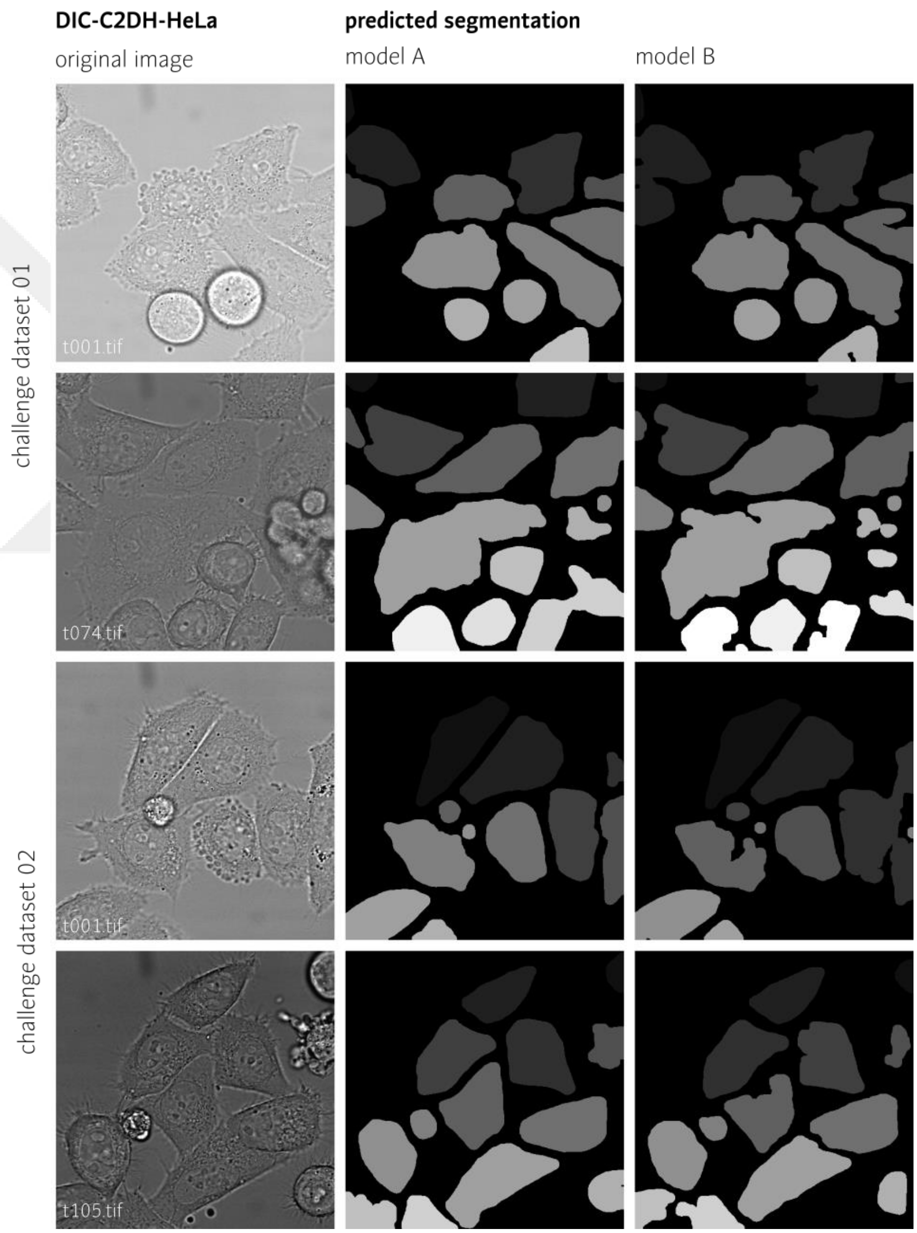

Figure 4.9: Visual comparison on the challenge sequences

Both models can correctly detect most of the cells. Masks are not covering the whole objects due to the gaps between them. The masks are also not touching to each other often. There is a distinctive gap between every two cells. The performance of both models are visually very similar. We cannot say that model B is much worse than model A. There are some detection errors in both predictions. We can conclude, that visually both models can detect cells on given images. The provided masks are not very precise. Their shape recalls the shape of masks, that was used for training. We can say, that provided segmentation corresponds to the expectations.

4.3.4 Cell Tracking Challenge

To measure model performance on the competition sequences, we decided to submit the best model to the Cell Tracking Challenge. The competition requires both the segmentation results and the tracking results. We did not implement any advanced tracking algorithm and approximated the tracking result only by a listing all the object on each frame. This tracking did not connect any two objects on different frames.

There are two measures, which express the quality of the algorithm in this challenge; the SEG measure for a segmentation quality and TRA measure for algorithm tracking ability. The third value used in the competition evaluation is called OP measure, and it is an average between SEG and TRA measures on all the tested sequences. Our *model A* reached the absolute third place for the DIC-C2DH-HeLa dataset. It ranked at third place in SEG measure and fifth place in TRA measure. The comparison between the best algorithms is in Table 3.4. The total third position with only a naive tracking algorithm is a success. It shows that segmentation provided by our model reaches a high quality.

Table 4.4: Cell Tracking Challenge results

Participant	DI C OP	-C2DH- H EG	eLa TRA
AC(2)	0.864	0.814	0.915
FR-Ro-GE	0.828	0.767	0.881
model A	**0.747**	**0.725**	**0.767**
KTH-SE	0.626	0.460	0.797
IMCB-SG	0.523	0.293	0.752
AC(1)	-	0.511	-

4.4 DISCUSSION

We successfully built a neural network that solves the given task. The prerequisite for this work was a deep study of the task as well as the methods we used. During the work, we also developed algorithms that help us to train and test different networks efficiently. We get familiar with different approaches how to deal with an insufficient size of the dataset. We used several pre-processing and post-processing techniques, which helped us to reach good results. We evaluated our model on public datasets and also get official results from the Cell Tracking Challenge organizers.

The chosen method was powerful in the given task. We out performed all the traditional image processing techniques on the given dataset. We used one of the most difficult datasets of the challenge so we expect, that this method would have good results also in the other datasets.

The method has several drawbacks. It seems to be hard to guarantee the expected performance on the unknown input. E.g., we trained a network that was successful in both training and validation datasets. Unfortunately, the performance was much worse on the challenge dataset. The reason was that light conditions of these sequences were different. We solved this problem efficiently by histogram equalization of input images, but it is hard to predict these situations. Other drawbacks are computational expenses and requirement of sufficient dataset.

In the field of machine learning and neural networks, it is not easy to compare different models. dataset quality and the process of training, too. For that reason, we did not include any wider comparison between different approaches. The correctness of such a comparison would not be guaranteed, because models can be trained in different conditions. The only relevant measure we have, except repeating all the experiments by ourselves, is the CTC competition.

During the research, we trained and tested many different models. They vary in topology, in hyperparameter setting either in the learning algorithm we used. We also tried several ways how to prepare the dataset. All this work experienced us, but it is not included in the final text. Not all of these experiments were finished or evaluated properly. Due to the wide variety of possible settings, it would be hard to deduce any general knowledge.

Our model was successful in the Cell Tracking Challenge. It was not the primary goal of this work, but it motivates us to future research. There are several possibilities, how the model can be improved. Firstly, it is possible to train a bigger network. We tried to keep model architecture relatively small due to the efficiency of all experiments. We showed that more complex model could improve the performance on the same dataset. It is also possible to change the network architecture more significantly. Secondly, there are many variants, how to pre-process training images. We can use another augmentation meth- ods to provide a larger dataset. There are also several options, how to define training masks and weight masks and we can also focus on post-processing. Thirdly, we can tune all the hyperparameters more precisely or try a different learning algorithm.

From our point of view, the promising way is also to connect ma- chine learning methods with traditional image processing methods. Both approaches differ in their advantages, and by proper combination, we can come with new methods.

5. CONCLUSION AND FUTURE WORK

In the thesis, we developed a successful deep learning method for segmentation of biomedical images. Constitutional neural networks .are suitable for solving this task The work on the topic was not as straightforward as it looks in. the final text All began by studying deep learning approaches suitable for this task as well as the image processing methods. We had to choose right software for the whole work and learned how to use it properly. Another important topic was how to reach appropriate computing resources. To train deeper networks, it was necessary to move all the computations to more powerful machines.Due to the lack of available experiences, it was not easy to make decisions about the future development, and we faced several impasses. In the end, the task itself was not easy to solve. The segmentation of biomedical images by constitutional neural networks is not well-explored field. It was necessary to make many non-deterministic decisions to train the model correctly and to finish the work. They should be investigated. deeper in the future.

In the end, I would like to clarify my motivation for research this field. In the beginning, there was a wish to improve my knowledge in the field of machine learning and constitutional neural networks. I am interested in this field because of the power of neural networks to solve certain kind's difficult tasks. It is exciting to deal with the problems that were classified as very hard or almost impossible.

The second reason is that I am already experienced in visual tasks. I like them because they are close to the natural human problem understanding. The *Segmentation of Dense Cell Population* is a task that combines both of these motivations.

I consider the whole research as successful. We met all the goals we set. I learned a lot about the training of convolutional neural networks, and we got experienced in the designing of networks. I also became familiar with many useful methods of image processing. I am still in- retested in the topic of semantic segmentation as well as convolutional neural networks and I am looking forward to continuing the research in this field in the future.

REFERENCES

[1] SAXTON, Michael J.; JACOBSON, Ken. SINGLE-PARTICLE TRACKING:Applications to Membrane Dynamics. *Annual Re- view of Biophysics and Biomolecular Structure*. 1997, vol. 26, no. 1, pp. 373–399. Available from DOI: 10.1146/annurev.biophys.26.1.373. PMID: 9241424.

[2] BRANDENBURG, Boerries; ZHUANG, Xiaowei. Virus trafficking–learning from single-virus tracking. *Nature Reviews Microbiology*. 2007, vol. 5, pp. 197–208.

[3] SINHA, Bidisha; BHATTACHARYA, Dipanjan; SINHA, Deepak Kumar; TALWAR, Shefali; MAHARANA, Shovamayee; GUPTA, Soumya; SHIVASHANKAR, G.V. Chapter 3 - Dynamic Organi- zation of Chromatin Assembly and Transcription Factories in Living Cells. In: SHIVASHANKAR, G.V. (ed.). *Nuclear Mechanics & Genome Regulation*. Academic Press, 2010, vol. 98, pp. 57–78. Methods in Cell Biology. ISSN 0091-679X. Available from DOI: https://doi.org/10.1016/S0091-679X(10)98003-5.

[4] CHENOUARD, Nicolas et al. Objective Comparison of Particle Tracking Methods. 2014, vol. 11.

[5] MAŠKA, Martin et al. A benchmark for comparison of cell track- ing algorithms. *Bioinformatics*. 2014, vol. 30, no. 11, pp. 1609–1617. Available from DOI: 10.1093/bioinformatics/btu080.

[6] ULMAN, Vladimír et al. An Objective Comparison of Cell Track ing Algorithms. 2017, vol. 14.

[7] KRÖSE, Ben; KROSE, Ben; SMAGT, Patrick van der; SMAGT, Patrick. *An introduction to Neural Networks*. 1993.

[8] MCCULLOCH, Warren S.; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. *The bulletin ofmathemat- ical biophysics*. 1943, vol. 5, no. 4, pp. 115–133. ISSN 1522-9602. Available from DOI:[8] MCCULLOCH, Warren S.; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. *The bulletin ofmathemat- ical*

biophysics. 1943, vol. 5, no. 4, pp. 115–133. ISSN 1522-9602. Available from DOI:10.1007/BF02478259.

[9] ROSENBLATT, F. The Perceptron: A Probabilistic Model for In- formation Storage and Organization in The Brain. *Psychological Review*. 1958, pp.65–386.

[10] WIDROW, Bernard; HOFF, Marcian E. Adaptive Switching Cir- cuits.In:*1960IREWESCONConventionRecord*,*Part4*.NewYork: IRE, 1960, pp.96–104.

[11] MINSKY,M.;PAPERT,S.*Perceptrons*.Cambridge,MA:MITPress, 1969.

[12] HOPFIELD, John. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*. 1982, vol. 79, no. 8, pp. 2554–2558. ISSN0027-8424.

[13] HOPFIELD, J. J. Neurons with Graded Response Have Collec- tiveComputational Properties like Those of Two-State Neurons. *Proceedings of the National Academy of Science*. 1984, vol. 81, pp. 3088–92. ISSN 0027-8424. Available from DOI: 10.1073/pnas.81. 10.3088.

[14] RUMELHART, David E.; HINTON, Geoffrey E.; WILLIAMS, Ronald J. Learning representations by back-propagating errors. *Nature*.1986,vol.323,no.6088,pp.533–536.Availablealsofrom: http://dx.doi.org/10.1038/323533a0.

[15] FUKUSHIMA, Kunihiko. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*. 1980, vol. 36, no. 4, pp. 193–202. ISSN 1432-0770. Available from DOI: 10 . 1007 / BF00344251.

[16] POMERLEAU, Dean A. ALVINN: An Autonomous Land Vehicle in a Neural Network. In: TOURETZKY, D. S. (ed.). *Advances in Neural Information Processing Systems 1*. Morgan-Kaufmann, 1989, pp. 305–313. Available also from:http://papers.nips.cc/paper/95alvinn-an-autonomous-land-vehicle-in-neural-network.pdf.

[17] KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. ImageNet Classification with Deep Convolutional Neural Net- works. In: PEREIRA, F.; BURGES, C. J. C.; BOTTOU, L.; WEIN- BERGER, K. Q. (eds.). *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012,pp. 1097–1105. Available also from: http : / / papers . nips . cc / paper/ 4824-imagenet-classification-with-deep- convolutional- neural networks. pdf.

[18] CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*. 1989, vol. 2, no. 4, pp. 303–314. ISSN 1435-568X. Available from DOI: 10.1007/ BF02551274.

[19] HAHNLOSER, Richard H. R.; SARPESHKAR, Rahul; MA- HOWALD, Misha A.; DOUGLAS, Rodney J.; SEUNG, H. Sebas- tian. Erratum: Digital selection and analogue amplification coex- ist in a cortex-inspired silicon circuit. *Nature*. 2000, vol. 405, no. 6789, pp. 947–951. Available from DOI: 10.1038/35016072

[20] GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook. org.

[20] GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook. org.

[21] LECUN, Yann; BOSER, Bernhard E.; DENKER, John S.; HEN- DERSON, Donnie; HOWARD, Richard E.; HUBBARD, Wayne E.; JACKEL, Lawrence D. Handwritten Digit Recognition with a Back-Propagation Network. In: *Advances*

[22] LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient- based learning applied to document recognition. *Proceedings of the IEEE*. 1998, vol. 86, no. 11, pp. 2278–2324. ISSN 0018-9219. Available from DOI:10.1109/5.726791

[23] OLAH,Chris;MORDVINTSEV,Alexander;SCHUBERT,Lud- wig. Feature Visualization. *Distill*. 2017. Available from DOI: 10.23915/distill.00007.https://distill.pub/2017/feature- visualization.

[24] RUSSAKOVSKY, Olga et al. ImageNet Large Scale Visual Recog- nition

Challenge. *CoRR*. 2014, vol. abs/1409.0575. Available from arXiv: 1409.0575.

[25] SZEGEDY, Christian; LIU, Wei; JIA, Yangqing; SERMANET, Pierre; REED, Scott E.; ANGUELOV, Dragomir; ERHAN, Du- mitru; VANHOUCKE, Vincent;RABINOVICH, Andrew. Going Deeper with Convolutions. *CoRR*. 2014, vol. abs/1409.4842. Avail- able from arXiv: 1409.4842.

[26] HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Deep Residual Learning for Image Recognition. *CoRR*. 2015, vol. abs/1512.03385. Available from arXiv: 1512.03385.

[27] CIRESAN, Dan Claudiu; GIUSTI, Alessandro; GAMBARDELLA, Luca Maria; SCHMIDHUBER, Jürgen. Mitosis Detection in Breast Cancer Histology Images with Deep Neural Networks. In: *MIC- CAI*. 2013, vol. 2, pp. 411–418.

[28] ARGANDA-CARRERAS, Ignacio et al. Crowdsourcing the cre- ation of image segmentation algorithms for connectomics. *Fron- tiers in Neuroanatomy*. 2015, vol. 9. Available from DOI: 10.3389/ fnana.2015.00142.

[29] TORRALBA, A.; MURPHY, K. P.; FREEMAN, W. T. Sharing fea- tures: efficient boosting procedures for multiclass object detec- tion. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Washington, DC, 2004, vol. 2, pp. 762–769.

[30] LONG, Jonathan; SHELHAMER, Evan; DARRELL, Trevor. Fully Convolutional Networks for Semantic Segmentation. *CoRR*. 2014, vol. abs/1411.4038. Available from arXiv: 1411.4038

[31] BADRINARAYANAN, Vijay; KENDALL, Alex; CIPOLLA, Roberto. SegNet: A Deep Convolutional Encoder-Decoder Archi- tecture for Image Segmentation. *CoRR*. 2015, vol. abs/1511.00561. Available from arXiv: 1511.00561.

[32] MAŠKA, Martin et al. A benchmark for comparison of cell track- ing algorithms. *Bioinformatics*. 2014, vol. 30, no. 11, pp. 1609–1617. Available from DOI: 10.1093/bioinformatics/btu080

[33] MAGNUSSON, Klas E. G. *Segmentation and tracking of cells and particles in time-lapse microscopy*. 2016. PhD thesis. KTH, School of Electrical Engineering (EES), Signal Processing.

[34] SIMARD, Patrice Y.; STEINKRAUS, Dave; PLATT, John C. Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In: *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2*. Wash- ington, DC, USA: IEEE Computer Society, 2003, pp. 958–. ICDAR '03. ISBN 0-7695-1960-1. Available also from: http://dl.acm.org/citation.cfm?id=938980.939477.

[35] ZUIDERVELD, K. Contrast Limited Adaptive Histogram Equal- ization. *Graphics Gems*. 1994, pp. 474–485. Available also from: https://ci.nii.ac.jp/naid/10031105927/en/.

[36] KINGMA, Diederik P.; BA, Jimmy. Adam: A Method for Stochas- tic Optimization. *CoRR*. 2014, vol. abs/1412.6980. Available from arXiv: 1412.6980.

APPENDIX A

SOFTWERE IN USE

In the whole work we used open source software. Python was the main programming language. The list of used packages with a description and the usage is in the table A.1.

Table A.1: Python packages

Package	Link	description and usage
Keras	keras.io	- high-level neural networks API, wide range of optimisation algorithms - creating models, training models
Tensorflow	tensorflow.org	- machine learning framework, supports CPU and GPU, high performance numerical computation - Keras backend, cost function definition, visualising learning
Numpy	numpy.org	- basic Python library for scientific computing - numerical array manipulation
OpenCV	opencv.org	- computer vision library - image processing
h5py	h5py.org	- interface to the HDF5 binary data format - manipulation with models stored by Keras
Jupyter	jupyter.org	- IDE for Python - interactive editor