BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF BIOMEDICAL ENGINEERING

SEGMENTATION OF BIOLOGICAL SAMPLES IN CRYO-ELECTRON MICROSCOPY IMAGES USING MACHINE LEARNING METHODS

MASTER'S THESIS
DIPLOMOVÁ PRÁCA

AUTHOR
AUTOR PRÁCE
Bc. Norbert Sokol

ADVISOR
VEDÚCI PRÁCE
Ing. Jiří Chmelík, Ph.D.

BRNO 2021

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Master's Thesis

Master's study program **Biomedical Engineering and Bioinformatics**

Department of Biomedical Engineering

Student: Bc. Norbert Sokol ***ID:*** 186690

Year of study: 2 ***Academic year:*** 2020/21

TITLE OF THESIS:

Segmentation of biological samples in cryo-electron microscopy images using machine learning methods

INSTRUCTION:

1) Get acquainted with principles of imaging of biological samples using cryo-electron microscopy (EM). 2) Carry out a search of literature dealing with convolutional neural networks (CNN) and their application for image data segmentation. 3) Based on the previous search, propose a concept of computer program dealing with the segmentation of biological samples from cryo-EM images. 4) In the appropriately chosen programming framework, implement the proposed CNN together with a suitable data pre-processing. Train the implemented CNN on the available image dataset. 5) Optimize the proposed approach in terms of the maximisation of its segmentation quality. 6) Statistically evaluate the functionality of the proposed method, objectively compare with other segmentation approaches, and appropriately discuss the achieved results.

RECOMMENDED LITERATURE:

[1] ASGARI TAGHANAKI, Saeid, Kumar ABHISHEK, Joseph Paul COHEN, Julien COHEN-ADAD a Ghassan HAMARNEH. Deep semantic segmentation of natural and medical images: a review. Artificial Intelligence Review. DOI: 10.1007/s10462-020-09854-1. ISSN 0269-2821.

[2] DANEV, Radostin, Haruaki YANAGISAWA a Masahide KIKKAWA. Cryo-Electron Microscopy Methodology: Current Aspects and Future Directions. Trends in Biochemical Sciences. 2019, 44(10), 837-848. DOI: 10.1016/j.tibs.2019.04.008. ISSN 09680004.

Date of project specification: 8.2.2021 ***Deadline for submission:*** 21.5.2021

Supervisor: Ing. Jiří Chmelík, Ph.D.
Consultant: Ing. Matej Dolník

prof. Ing. Ivo Provazník, Ph.D.
Chair of study program board

ABSTRACT

Cryo-electron microscopy imaging has its irreplaceable position in analysis of various biological structures. Localization of the cells cultivated on grid and their segmentation towards background or contamination is essential. With the development of various deep learning methods, the performance of semantic segmentation tasks dramatically increased. In this thesis, we will develop a deep convolutional neural network for semantic segmentation of the cells cultivated on grid. Dataset for this thesis was created with dual-beam cryo-electron microscope developed by Thermo Fisher Scientific Brno.

KEYWORDS

cryo-electron microscopy, grid sample preparation, deep learning, machine learning, convolutional neural networks, semantic segmentation.

ABSTRAKT

Zobrazovanie pomocou kryo-elektrónovej mikroskopie má svoje nezastúpiteľné miesto v analýze viacerých biologických štruktúr. Lokalizácia buniek kultivovaných na mriežke a ich segmentácia voči pozadiu alebo kontaminácii je základom. Spolu s vývojom viacerých metód hlbokého učenia sa podstatne zvýšila úspešnosť úloh sémantickej segmentácie. V tejto práci vyvinieme hlbokú konvolučnú neurónovú sieť pre úlohu sémantickej segmentácie buniek kultivovaných na mriežke. Dátový súbor pre túto prácu bol vytvorený pomocou dual-beam kryo-elektónového mikroskopu vyvinutého spoločnosťou Thermo Fisher Scientific Brno.

KĽÚČOVÉ SLOVÁ

kryo-elektrónová mikroskopia, preparácia vzorku na mriežke, hlboké učenie, strojové učenie, konvolučné neurónové siete, sémantická segmentácia.

SOKOL, Norbert. *Segmentation of biological samples in cryo-electron microscopy images using machine learning methods*. Brno, 2021, 62 p. Master's Thesis. Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Biomedical Engineering. Advised by Ing. Jiří Chmelík, Ph.D.

DECLARATION

I declare that I have written the Master's Thesis titled "Segmentation of biological samples in cryo-electron microscopy images using machine learning methods" independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the thesis and listed in the comprehensive bibliography at the end of the thesis.

As the author I furthermore declare that, with respect to the creation of this Master's Thesis, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll., Section 2, Head VI, Part 4.

Brno ....

author's signature

ACKNOWLEDGEMENT

I would like to thank the advisor of my thesis, Ing. Jiří Chmelík, Ph.D. for his valuable comments and helpful consultations during elaboration of this thesis, and Ing. Matej Dolník for his support and mediation of helpful resources from the company.

Contents

List of Figures

Introduction

Cryo-electron microscopy is popular method for imaging biological samples. Quick cooling renders the specimen stationary and, while constantly cooled, may be analyzed. Various procedures obtaining this conditions are in use [1]. Sample analysis and further processing dictates a demand for automatic distinction of the samples towards background or contamination. This task can be subjected to semantic segmentation.

Numerous methods for image segmentation has been published, as summed in [2]. However, with the development of machine learning and Deep neural networks, these challenges can be tackled with higher precision, robustness and automation.

In the first chapter, we will introduce principles of cryo-electron microscopy imaging with its advantages and disadvantages, together with illustration of the procedures used in practice. Second chapter will outline deep learning principles and practices. Third chapter is dedicated to convolutional neural networks, theoretical background, components and their impact in reinforcing the network. Fourth chapter is an overview research of deep neural networks used in practical applications for semantic image segmentation and proposal of the architecture for practical part of this thesis. Fifth chapter presents proposed architecture for the task of semantic image segmentation of the cell samples acquired from cryo-electron microscopy. In this chapter, proposed architecture and several experiments, together with optimization of the final model's performance will be described. In sixth chapter, results of the fifth chapter will be discussed.

1 Electron microscopy imaging

1.1 Past and Present

One of the essential traits of humanity is progress. People were always fascinated by diving into the unknown. History depicts, that observation plays a crucial part on the path of gaining new knowledge. This was a cause of developing new methods and practices, when observing and collecting information. There was a need to observe smaller and smaller objects, that surround us, over the time. Small things (from Greek mikros), needed to be looked on (skopeo).

The first invention of compound electron microscope took place in year 1590 and belongs to Janssens. With magnification up to 30 times of the object's natural size, microscopy started to evolve. In following century, Antonie van Leeuwenhoek improved magnification up to 300 times, with development of single lens microscope. 1000 times magnification and 0.2 μm distinction of adjacent objects was possible in early twentieth century. Biological structures showed dynamic, complex and vivid behavior.

1930s were accompanied with observation of submicroscopic world via electron microscope. With many times shorter wavelength of the electron, it was possible to increase the magnification by a thousandfold. This was a tremendous leap for many fields of science including biology. It was a time, when the smallest structures of cell were inspected, following with DNA, RNA, viruses, molecules and even the atom itself. From that point, refinement of the technology and imaging was essential [3].

Eventually, two basic types of technology evolved. Both using electron beam, but with different applications. The main microscope types are Transmission Electron Microscope (TEM) and Scanning Electron Microscope (SEM) [4].

TEM focuses electrons through specimen, that is very thin, to form a 2D image. Number of transmitted electrons through the sample controls the brightness of particular area of the image. SEM scans the surface of the sample, creating secondary electrons emitted from the specimen, that are subsequently detected via sensor. The image is formed sequentially, as the sample is being scanned [5].

While TEM images offer information about internal structure of the specimen, SEM produces impression of 3D imaged surface. More precisely, magnified images of SEM offer variety of information of the examined specimen including shape, size, composition, crystallography and other chemical and physical properties. Currently, SEM and TEM have many components in common. However, the main difference regarding image acquisition is their spatial resolution. SEM is limited approximately to 0.5 nm. For TEM, reports with spatial resolution lesser than 50 pm have been published [6]. Fig. 1.1 depicts componential difference of TEM and SEM.

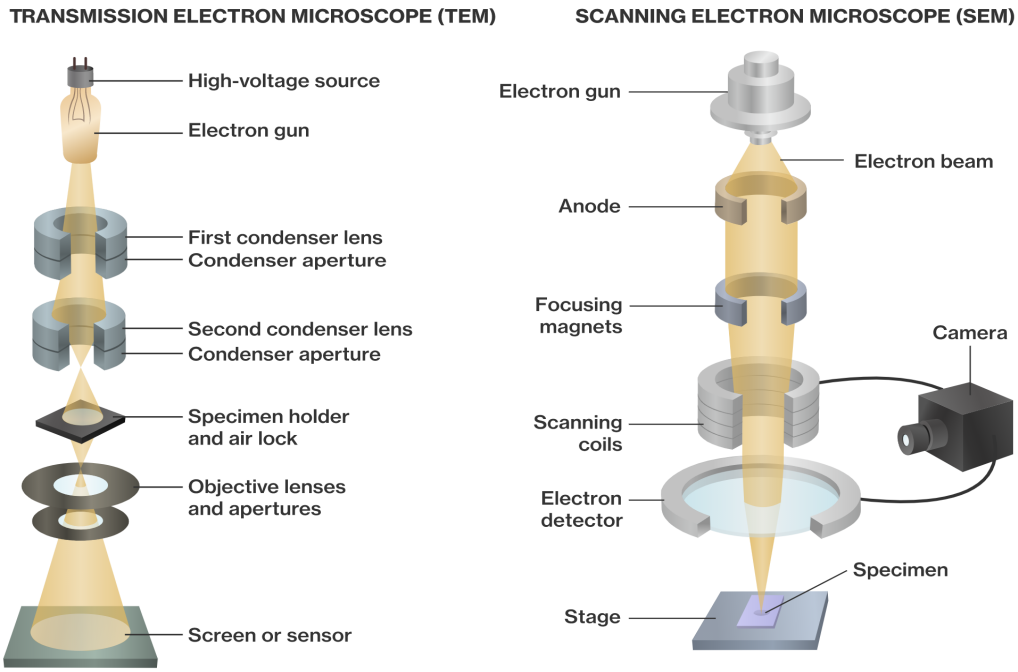

Fig. 1.1: Operating principle of SEM and TEM. TEM on the left transmits electron beam through the sample and transmitted electrons are captured under the sample, while SEM on the right scans with electron beam and emitted electrons are captured over the sample [7].

Practical solution of this thesis process images formed via cryo-SEM. Thus, SEM principle of operation will be discussed, along with an explanation of possibilities and drawbacks of using cryogenic mode.

1.2 Scanning electron microscopy

There are several operating parameters, by which the operator can manipulate the character of electron beam reaching the surface of the sample. Energy can typically range from 0.1 - 30 keV, diameter from 0.5 nm to 1 μm, beam current from 1 pA to 1 μA and beam convergence angle from 0.001 to 0.05 rad.

When focused electron beam reaches the surface of the material, atom density rapidly increases due to high density of the specimen. This is accompanied with a set of physical manifestations, that are known as scattering events. These events produce signals in the form of back-scattered electrons, secondary electrons, characteristic x-ray radiation and other photons with different energy levels [1].

For SEM, the primal interest lies in secondary and backscattered electrons. The

depth of field and the shadow relief effect of these electrons allows three-dimensional appearance of produced images. SEM is preferred for producing images at high-resolution, when scanning bulk objects. Next valuable feature is its depth of field. With greater depth of field, higher amount of information from specimen is acquired, as well as 3D specimen appearance. This feature allows examination of 3D objects, which is widely used in stereoscopy. In addition, SEM is able to produce images at very low magnification. In applications such as forensic studies, these images can serve as a complement to images taken via light microscope.

Further development of SEM allowed structure analysis. This capability of SEM can determine crystalline structure and crystal's grain orientation on the specimen's surface. This is allowed mainly by backscattered electrons, and hence it is known as electron back-scattering diffraction. The specimen's surface is generally tilted up to 70° from horizontal orientation in order to acquire maximal intensity of the diffraction pattern. This pattern of back-scattered electrons is referred to as Kikuchi's pattern. Its intensity is very low and highly sensitive cameras (such as charge-coupled devices in the present) are essential in analysis [6].

X-rays in SEM are used in energy-dispersive x-ray analysis (EDX). When SEM is equipped with an EDX detector, it is possible to obtain information about chemical composition of the sample, elements present in the bulk, as well as their distribution and concentration. The principle of EDX analysis can be briefly described as follows. When an electron beam is focused on the sample, it has a chance to knock-off electron from the inner shell of the atom, that creates positively charged electron hole. After displacement, another electron from outer shell will fill the vacancy. While moving from outer higher-energy shell to inner lower-energy shell, the remaining energy can be released among others in the form of x-ray radiation with characteristic properties. These properties, such as x-ray energy, are specific for particular elements and transitions. Followingly, x-rays are detected via silicon drift detectors and interpreted through software. Various visualizations of the chemical composition of a sample can be performed, giving qualitative and also quantitative information of elements present, as well as their concentrations [1], [8].

1.3 Low-temperature sample preparation

Low-temperature sample preparation is essential, when it is required to study hydrated specimens. Applications are various. It is a prerequisite to examine naturally frozen samples, such as snow and ice. In the food industry, it is used to examine frozen foods like meat, vegetables or ice cream. Chemical industry requires use of frozen samples of different substances, emulsions, suspensions, non-aqueous materials like oils, even gases and volatile materials. Cryo-temperatures are beneficial also

in preparations of plastic materials, polymers or elastomers. Construction industry requires closer examination of different soils, clays or cements. In biological studies, there is a need to freeze aqueous and non-aqueous fluids of biological specimen, which are very active and move in natural temperatures.

When an examination and following analysis of these materials is needed, they require to be solidified with application of cooling substance and acquire temperature below their melting point. Main advantage of this process is maintenance of particular triphasic state of the material. Liquid, flexible or soft materials and their dissolved elements and molecules remain *in situ* in the form of solid matrix. In biological applications, low temperatures, especially cryogenic temperatures, can immobilize dynamics and physiology of the specimen. In addition, very low temperatures mitigate radiation damage from high-energy electron beam.

The drawback of cooling the hydrated material is formation of crystalline materials. When a hydrated material is in transit from liquid to solid phase, it normally forms high-ordered eutectic mixture of ice crystals and partitioned solids. This diminishes the ability to examine structural information of the specimen in comparison to its originally random organization [1].

1.3.1 Properties of water and ice

Water is connected with numerous interactions via chemical reactions. Its properties are related to the ability of its molecules to form hydrogen bonds with hydrophilic substances and with each other. Water's crucial property, in order to solidify, is its binding state. The ability to form ice is decreased, when water is more closely bonded to hydrophilic solids or solutes. Another factors affecting transition to solid state are pressure and temperature, based on which water can stay in one metastable amorphous form and several crystalline forms.

Temperature below 273.15 K is specific for hexagonal crystalline form of ice. Transition from disorganized liquid to highly organized solid at the temperature below 273.15 K causes substantial restructuralization. Characteristic examples are snowflakes, ice cubes or frozen food. This crystalline form is undesirable when preparing samples. On the other hand, when the sample is extremely small and rapidly cooled, non-crystalline, i.e., amorphous form of ice is present. This form is metastable, which creates higher pressure on conditions for its maintenance. Once this non-crystalline ice is formed, it must be maintained below glass transition temperature. Otherwise, the random arrangements of water start to devitrificate and initiate crystallization. The glass transition temperature of demineralized water is 135 K. In practice, water is bounded and contains solutes, which elevates glass transition temperature [9].

1.3.2 Transition from liquid to solid

There are three closely related processes in transition from liquid water to solid ice that occur, respectively:

1. *Heat removal.* Specimen-dependent process, in which thermal energy is removed from the specimen and transmitted to the ambient low-temperature environment until thermal equilibrium is balanced.
2. *Nucleation.* Random movement of water molecules creates tiny clusters, that allow condensation of another water molecules, and eventually form the ice crystal's nucleus. These nuclei define crystal structure of the solid. Homogenous nucleation occurs when pure water is present. In practice, this process is rarely seen because water in specimen is not completely pure. Heterogenous nucleation occurs around a disturbance, solvents or contamination of water. The effect of this process can be reduced with higher concentration of water or with lower temperature.
3. *Ice crystal growth.* Nucleation created nuclei for crystal growth. The rate of growth is affected by several factors such as transport speed of water molecules to the crystallization point, accommodation of the molecules and transport of the heat from crystal-growing site [10].

Overall crystal growth rate is crucially dependent on heat removing rate of the material. This can be categorized by cooling rate. Very slow cooling with the rate approximately 0.01 K/s, creates only few nucleation events and develops large crystals. Increasing the rate up to ultrarapid cooling with rate approximately 100 000 K/s, produces so many nucleation events, that crystallization cannot virtually happen in such a short time. Higher the cooling rate, the higher amount of amorphous ice is produced [5].

1.3.3 Quench cooling

The process of cooling the specimen as rapidly as possible is called quench cooling. It is the most crucial part in the preparation of specimen. The aim is to create as little crystals of ice as possible. Cooling substances are referred to as cryogens and can be solid or liquid. Important characteristics of cryogens are:

1. Low melting point and high boiling point
2. High thermal capacity and thermal conductivity
3. Low viscosity and high density at boiling point

4. Harmless, environmentally friendly, inexpensive...

There are various cryogens that are used in practice and various methods for quench cooling. Modality for cryogenic sample preparation and image acquisition for this thesis used liquid nitrogen as cryogen [5].

1.4 Cryogenic sample preparation

Samples for Cryo-SEM were prepared via Thermo Fisher's cryogenic sample preparation guideline. The main principles are summed by:

1. Vacuum in the chamber should be lower than $4 \times 10^{-4} Pa$.
2. Cryo stage and cryo trap were supplied by liquid nitrogen with flows at least 2 l/min for two hours.
3. Every item in the preparation station should be dry and clean (avoiding contamination with water crystals and impurities).
4. Samples are cooled in a preparation base with liquid nitrogen. Grid with cell samples is gently placed to the shuttle. Preparation base and shuttle is visible in Fig. 1.2 a, b.
5. Cooled samples are removed from the preparation base and inserted into the airlock lid of the microscope via transfer rod. Transfer rod is show in Fig. 1.2 c.
6. Samples are pushed with the transfer rod to the cryo stage, which is constantly cooled from the cryogen dewar.
7. In order to render sample conductive, sputter coating of the grid with samples needs to be applied. This will depone a thin layer of a metal over the sample. It is a compulsory step in imaging organic samples.
8. In the microscope control environment, the grid is centered and each window in the grid is imaged. This creates final images that are used in practical part of this thesis. Image of the whole grid is depicted in Fig. 1.2 d.

After image acquisition, it is common to create a thin lamella from the samples. This is accomplished via focused ion beam that is installed in the microscope. Ion beam (mainly Gallium) is focused on the sample. Ions are large particles which, in this particular case, are milling through the sample from two opposite directions, creating thin layer of sample. These thin layers are then extracted from the grid and transported to transmission electron microscope for further analysis.

Fig. 1.2: Grid sample preparation. Part (a) depicts the preparation base with cryogen reservoir and sample shuttle. (b) provides closer look to the samples in the preparation base. (c) shows transfer rod to move the sample from preparation base to microscope chamber. (d) depicts the image of the whole grid with the samples. [11]

2 Deep Learning principles

2.1 Artificial neuron model

Deep neural networks with numerous layers are state-of-the-art models for a variety of challenges in computer vision. Deep Learning (DL) is subset of machine learning (ML) and its progress was facilitated by substantial increase in computational capability of Graphics Processing Units (GPU) [12].

The computation occurs on the fundamental basis of neural network (NN), an artificial neuron. It is a mathematical representation inspired by processes in mammal brain. Using artificial neuron chaining and activating the neurons with different activation functions can create specific architectures of NNs [13]. Fig. 2.1 shows basic schematic of artificial neuron with its activation function.

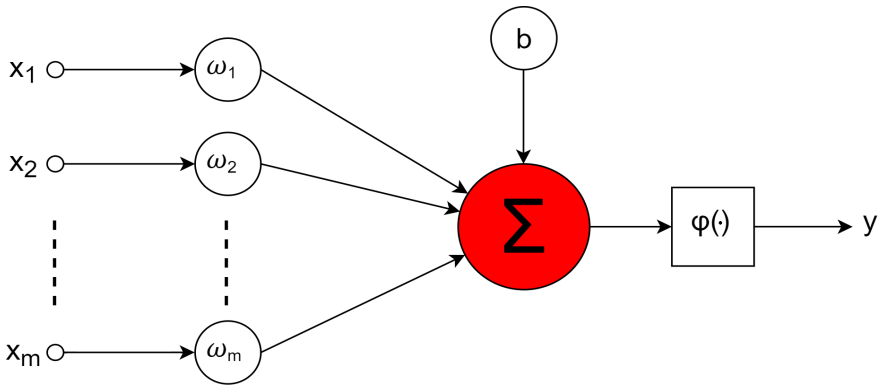

Fig. 2.1: Artificial neuron model. Inputs x are multiplied with weights ω and proceeded to summation element. Bias b is added and activation function $\varphi(\cdot)$ is applied to generate output y.

Output y is computed as follows:

$$y = \varphi(\sum_{i=1}^{m}(\omega_i x_i) + b) \tag{2.1}$$

y	Artificial neuron output
$\varphi(\cdot)$	Activation function
ω_i	Input weight i
x_i	Input value i
m	Number of input values
i	Index of input value
b	Bias - modifies activation function input

Activation is a key operation in transferring information from neuron to output. Different activation functions will be presented in section 3.2.2.

2.2 Designing a neural network

Multitude of architectures solving different challenges of deep learning has been published [13]. Generally, an architecture comprises input layer, hidden layers and output layer. Basic arrangement of neural network layers is shown in Fig. 2.2.

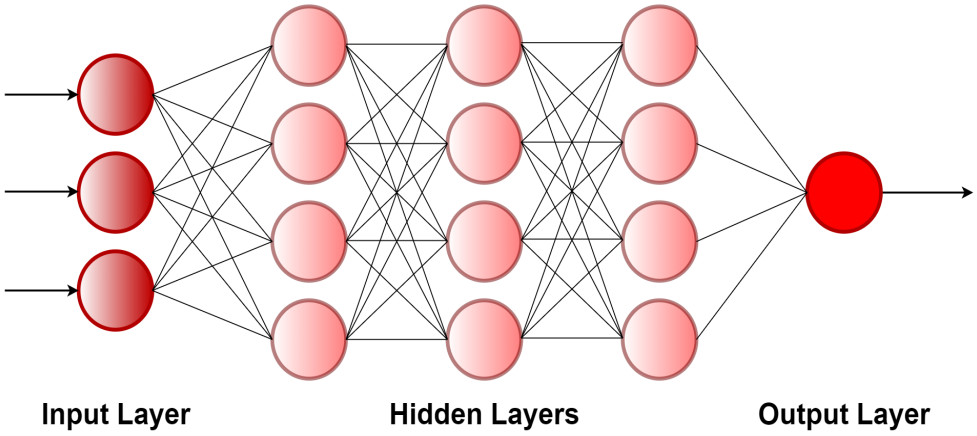

Fig. 2.2: Basic arrangement of NN layers. Input layer transfers the data to hidden layers for processing and output layer produces output representation.

Input layer's function is only to pass data, activated by activation function to the hidden layer. Hidden layer then applies weights to the inputs, applies specific activation function, and subsequently passes the data to another hidden layer or to the output. Output layer is then responsible for calculation the output from the previous layers via its activation function [13].

2.3 Training neural network

With designed architecture of neural network, training may begin. The first step is weight initialization, which aims to generate initial parameters from certain distribution. There are several techniques for weight initialization. The most used are Xavier initialization, used with hyperbolic tangent activation function [14] and He initialization, used with ReLU activation function [15].

The next phase is the forward pass. In this phase, input values are passed through each layer of the architecture in the forward direction. Calculations for each neuron

are performed, neurons are activated and the process ends in the output layer. Neurons of the output layer are activated depending on solving either classification or regression task.

Followingly, it is required to estimate how well the model predicts with given set of values. The loss function will estimate the error of the prediction from the actual ground truth. Loss functions can be distinguished by the aspect of particular task [16]. The most common functions include:

- Regression loss functions – used in predicting real number values.

    1. L1 / Mean Absolute Error (MAE) loss – finds average of sums of absolute differences between the predictions and ground truth [17]. MAE is computed as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.2}$$

    y_i Ground truth value
    $\hat{y}_i$ Predicted value
    n Number of observations
    i Observation index

    2. L2 / Mean Square Error (MSE) loss – finds average of sums of squared differences between the predictions and ground truth [17]. Equation is given by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.3}$$

- Classification loss functions – used in predicting correspondence to the particular class.

    1. Cross-entropy (log loss) – computes differences between the probability of predicted class and ground truth class across logarithmic scale. Cross-entropy can be expanded to weighted cross-entropy, which favors under-represented classes in imbalanced datasets. This can be further expanded to multi-class weighted cross-entropy to calculate weighted loss for more than two classes [18]. Binary cross-entropy loss can be calculated as:

$$BCE = \frac{1}{n}\sum_{i=1}^{n}(y_i \cdot log(p_i) + (1 - y_i) \cdot log(1 - p_i)) \tag{2.4}$$

y_i Ground truth class
p_i Predicted probability of a class

2. Tversky loss – computes overlap between the predicted and ground truth class with 1 as entire overlap. Used heavily in image segmentation tasks. By setting α and β, which are weights of false positive and false negative predictions, respectively, other loss functions can be derived:

   (a) $\alpha = \beta = 0.5$ – Dice coefficient (F1 score)
   (b) $\alpha = \beta = 1$ – Jaccard coefficient (Tanimoto)
   (c) $\alpha + \beta = 1$ – Higher α results in higher weight for false positives. Higher β results in higher weight for false negatives. Used for imbalanced datasets [19].

   Binary Tversky loss function is defined as:

$$Tversky = \frac{\sum_{i=1}^{n} y_{i,1} \cdot \hat{y}_{i,1} + \epsilon}{\sum_{i=1}^{n} y_{i,1} \cdot \hat{y}_{i,1} + \beta \sum_{i=1}^{n} y_{i,1} \cdot \hat{y}_{i,0} + \alpha \sum_{i=1}^{n} y_{i,0} \cdot \hat{y}_{i,1} + \epsilon} \tag{2.5}$$

$y_{i,1}, y_{i,0}$ Ground truth classes 1 and 0
$\hat{y}_{i,1}, \hat{y}_{i,0}$ Predicted probability of classes 1 and 0
β Weight for balancing false negatives
α Weight for balancing false positives
ϵ Coefficient preventing zero division

Backward pass phase follows. Main task of this phase is to adjust weights and biases to minimize loss. Level of optimization is determined by gradients of loss function over network's parameters. Assume an objective function w.r.t. all parameters of the network $J(\theta)$. Gradients ∇_θ determine the difference of these parameters w.r.t. difference in loss. Parameters are updated in the opposite gradient's direction of the objective function $-\nabla_\theta J(\theta)$. This is multiplied by learning rate μ, which controls the magnitude of gradient updates to reach a local optimum [20]. This optimization method is known as batch gradient descent defined by:

$$\theta = \theta - \mu \cdot \nabla_\theta J(\theta) \tag{2.6}$$

θ	Network parameters
μ	Learning rate
∇_θ	Gradient w.r.t. network parameters
$J(\theta)$	Objective function w.r.t network parameters

Batch gradient descent is, however, poor optimization technique. More sophisticated techniques will be discussed in the following section 2.3.1.

Computation of gradients is directed backwards in architecture via back-propagation algorithm [21]. After update of parameters, new iteration with forward pass begins. Gradual minimization of loss function is the actual learning in context of neural networks. This progressively boosts performance of the model. Here is also introduced the first hyper parameter of the network, which is the learning rate μ.

One process of forward and backward pass of the entire training dataset is called an epoch. Batch is a subset of training dataset, when it not possible to pass whole training dataset to network. The number of iterations in epoch is defined by ratio of training dataset's size to batch size [22].

2.3.1 Optimization algorithms

Back-propagation algorithm works in conjunction with an optimization method. Optimization methods of neural network via gradient-based methods are referred to as optimizers. Several optimizers are available and their performances differ given the task.

Stochastic gradient descent (SGD)

Unlike basic batch gradient descent (BGD) method, which is not in use, weight update is performed every iteration. It is computationally expensive with large datasets and the variance of updates is on high level. This can be beneficial for jumping to more suitable local extrema, however, the global extreme can be leaped over. Combination of BGD and SGD called Mini-batch GD arose, where weight update is performed after n iterations. Computational time is reduced, so as the variance of updates. Mini-batch GD algorithm enables computational parallelization and is widely used in practice [20].

SGD with momentum, Nesterov acceleration

This optimizer incorporates momentum. When previous iteration has the same direction of movement as current iteration, SGD movement will accelerate. Opposite direction will decelerate the movement. Besides having parameter of learning rate μ, this approach has another hyperparameter of momentum α, which controls exponentially decayed average of the past gradients. SGD with momentum is more willing to abandon local extrema and tracks the loss function more effectively. Another modification of SGD with momentum is Nesterov accelerated gradient (NAG). NAG is suchlike SGD with momentum, however, gradient is calculated from the point of prior estimation of the movement. This approach restrains fast movement in improper direction, which helps converge faster than SGD with momentum [23].

Adagrad, RMSprop/Adadelta

Adagrad helped solving learning rate initialization problem by changing learning rate separately for each weight. Algorithm accumulates sum of squares of past gradients for each weight as a matrix. As the gradient accumulates, learning rate experience shrinking, which may vanish the weights. Further optimization methods, such as RMSprop/Adadelta, solved shrinking of the weights by calculating only with w of past accumulated gradients instead of all past squared gradients [24], [25].

Adaptive moment estimation (Adam)

This method benefits from using both exponentially decaying average of past squared gradients, as incorporates RMSprop/Adadelta, and exponentially decaying average of past squared gradients, as incorporates SGD with momentum. Optimizer works with additional hyperparameters β_1 and β_2, which control decay rate of first and second moment. It is a state-of-the-art optimizer with many variants explained below. Since AdamW is used as an optimizer in the practical part of this thesis, further explanation will be provided. The core of this optimizer with its modifications lies in the following equations:

$$g_t = \nabla_\theta f_t(\theta_{t-1}) + w_t \cdot \theta_{t-1} \tag{2.7}$$

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \tag{2.8}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \tag{2.9}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.10}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.11}$$

$$\theta_t = \theta_{t-1} - \eta_t \left(\frac{\alpha \cdot \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + w_t \cdot \theta_{t-1} \right) \tag{2.12}$$

g_t	Gradient in timestep t
$\nabla_\theta f_t(\theta)$	Vector of partial derivates of differentiable objective function f_t, w.r.t. parameters θ, evaluated at timestep t
θ_{t-1}	Parameters at the previous timestep
w_t	Rate of the weight decay at timestep t
m_t, v_t	Biased first and second moment estimates
β_1, β_2	Decay rates for first and second moment estimates
$\hat{m}_t, \hat{v}_t$	Bias-corrected first and second moment estimates
α	Learning rate
η_t	Schedule multiplier
ϵ	Coefficient preventing zero division (10^{-8})

The original Adam optimizer, as proposed in [26], is calculated without the highlighted parts. m_t and v_t are actually moving averages of mean and uncentered variance of the gradient, which are updated as in equation 2.8 and 2.9. They are initialized by zeroes, and when β_1, β_2 are close to 1, they are zero-biased. This is thus negated by bias-corrected moments $\hat{m}_t$ and $\hat{v}_t$, updated by equation 2.10 and 2.11.

The update of learning rate α, as described by equation 2.12, is prudent, guided by element $\hat{m}_t / \sqrt{\hat{v}_t}$ (assuming omission of ϵ). When this element is small, (authors resemble it to a signal-to-noise ratio), the effective step size $\alpha \cdot \hat{m}_t / \sqrt{\hat{v}_t}$ is going to be close to zero. This behavior is desirable, for instance, when close to optimum, the effective step will be smaller and vice versa. α dictates the upper bound of the range of steps in parameter space, which allows to deduce correct range order to achieve optimum in a few iterations.

Addition of the pink element to the equation 2.7, is known as L2 regularized Adam or Adam with weight decay. However, authors of [27] have observed, that L2 regularization in not identical with weight decay. While for SGD algorithm L2 regularization equals weight decay, in Adam, it is not effective. When L2 is combined with adaptive gradients of Adam, large past gradients are regularized less than with weight decay. In order to improve the regularization, authors have proposed weight decay decoupled from the gradient-based update called AdamW. This improvement is changing the pink element of the equation 2.7, to red highlighted part in equation 2.12. In result, the weight decay is not calculated with the moving averages. η_t is

the scaling factor, which allows scheduling learning rate α, as well as weight decay rate w_t.

Decoupling the weight decay in Adam also improved the performance of the previously proposed method of the same authors for learning rate scheduling. This modification combines AdamW with normalized weight decay, where the scaling factor η_t decays according to cosine annealing. This method is known as Adam with warm restarts, AdamWR [27].

As an extension of AdamW algorithm, the calculation can be improved with AMSGrad modification [28]. The core amend is usage of maximum of past squared gradients instead of its exponential average. Thus, equation 2.11 is modified to $\hat{v}_t = max(\hat{v}_{t-1}, \hat{v}_t)$.

2.4 CNN regularization

Required ability of ML model is its performance on inputs previously unseen. This ability is called generalization. However, DL models produces substantial number of parameters and they are prone to overfitting in training phase. Overfitting occurs when the difference between training and testing (or validation) error is too high. In addition, the training error is low, the model's complexity is high and fits too much for the training data. This will result in high testing error on new data. For model to perform well, the possibility is either lower the training error, or reduce the difference between testing and training error [21], [29].

Regularization techniques are tackling these challenges. Regularization is modification of the model, whose outcome is to diminish testing error, but not training error. Regularization approaches are various in their idea, and those, which are related to Convolutional Neural Networks (CNNs), will be introduced.

2.4.1 Dataset augmentation

In general, if the model can be trained on more input data, higher level of generalization can be achieved. However, in several cases (usually sensitive, lawfully protected data, e.g., medical images), it is difficult, if not even impossible, to acquire plentiful dataset. There is a possibility to generate fake data and add them to dataset. Such an approach is called dataset augmentation. It has proven more than useful, especially with images, because they contain high amount of variability, which can be effortlessly simulated [21]. The augmented data are produced from original data in the dataset. Approaches for image augmentation include:

1. *Geometrical transformations.* Cropping, scaling, rotation or flipping
2. *Modifications of intensity.* Point-wise operations, color modifications
3. *Blurring/sharpening*
4. *Noise addition*
5. *Random image generation via Generative Adversarial Networks*

2.4.2 Early stopping

As mentioned earlier, sophisticated models with high representational capacity are prone to overfit after several epochs. This behavior is accompanied with continued decrease of the training error, however, validation error switches from decreasing tendency to increasing. Theoretically, when the training process is terminated in this point, the most suitable model with the lowest validation error (and in the test phase supposedly lower test error) is acquired. Development of testing and validation error is presented in Fig. 2.3.

Early stopping algorithm is ensuring, that the parameters of the model with lowest validation error are returned at the end of training (and not the parameters from the last epoch). Such an approach has proven to behave effectively and is valued for its simple implementation, which does not affect learning dynamics of the training process. Another feature of early stopping is, that it reduces computational time, as the additional training can be ceased, when there are no improvements in diminishing validation error for certain number of iterations. Early stopping requires validation dataset, so training dataset must split into training and validation part, which naturally reduces training ability of the model. Early stopping algorithm needs to be incorporated and a copy of the best parameters needs to be stored [21], [30].

2.4.3 Dropout

Dropout is powerful regularization tool. Solving advanced challenges of Deep Learning often requires building several architectures with gigantic number of parameters. Ensemble methods are striving to suffice the demand, however, this approach is highly time consuming, complex and requires enormous computational power. Dropout can be depicted as an approximation to ensemble methods, but with exponential number of neural networks, compared to a few networks in bagging for instance. Dropout's core function is to disable percentual number of neurons in a layer, by setting neuron's output to zero. More specifically, as Fig. 2.4 depicts,

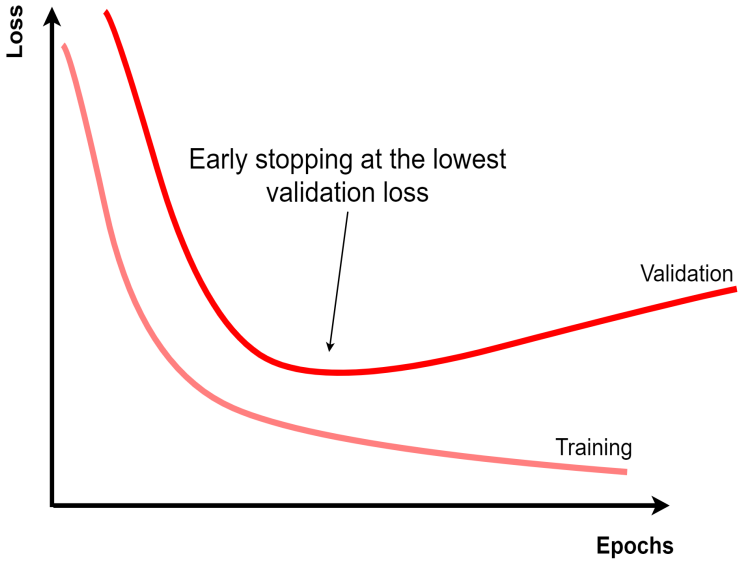

Fig. 2.3: Idealized development of learning curves. When validation error (red) switches from decreasing to increasing tendency, training should be stopped in that epoch.

there is no input or output connection of a neuron dropped out within the architecture. This is applicable only for hidden layers (dropout in input or output layer is counterproductive).

Each neuron is given a probability to be kept (higher probability equals lower dropout ratio). Forward pass follows with division of output from each neuron by the probability from previous layer to conserve energy in the architecture. Backward pass computes gradient and loss for not disabled neurons and weights are updated. Finally, all neurons are activated and another epoch or mini-batch is started with another realization of dropout. In testing phase, all sub-architectures are combined by activating all neurons (omitting the dropout or setting the probability to 1).

The resemblance with bagging is, that in each epoch/mini-batch, a subset of architecture's neurons is actually participating in the training process, creating different architecture. However, in the case of dropout, different architectures share weights, in contrast with independent weights in architectures of bagging. To summarize, it is very simple, fast and easy to implement regularization. Dropout can provide 2^N possible sub-architectures, where N stands for number of neurons in whole network [21].

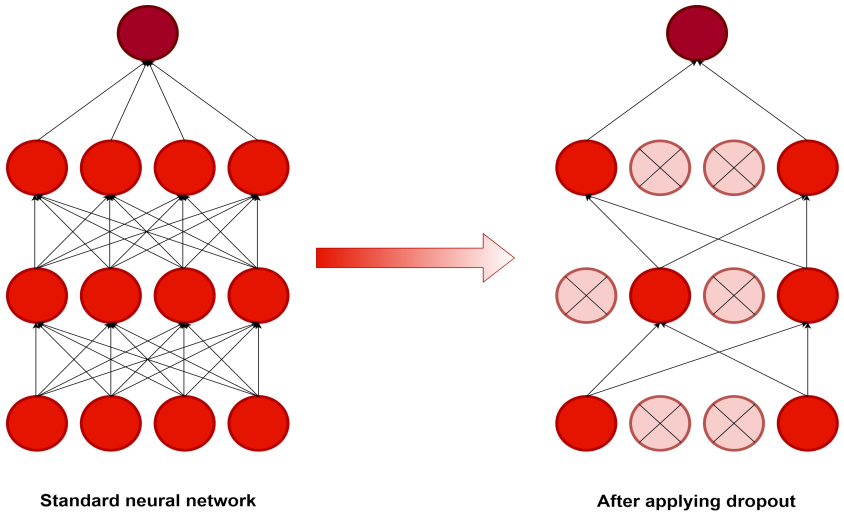

Fig. 2.4: Dropout schematics. Left image represents neural network without dropout. Right image depicts an architecture with dropout included, containing suppressed neurons depicted with a cross.

2.4.4 Batch normalization

Batch Normalization is efficient regularization technique used in very deep networks. As the name suggests, this procedure takes an input batch and normalizes mean and variance of the output activations from convolutional layer to acquire Gaussian distribution. This method suppress internal covariance shift of the convolutional layer's activations. This shift can be understood as a change in the distribution, that a layer is trying to predict. If it is high, the convergence and training process will be slowed. The effect of batch normalization has multiple advantages including:

1. Derivable functions allowing incorporation of batch normalization layer to the network, thus allowing end-to-end network training
2. Stabilization of the network, such as diminishing exploding or vanishing gradient and reduction of asymptotic saturation
3. Reduction of overall epochs required for training, thus reducing computational time
4. Regularization effect allows the network to increase its generalization performance
5. Training is less prone to wrong hyper parameter setting (e.g., learning rate)
6. Robustness against poor weight initialization
7. Making model less dependent on dropout or similar regularization methods

In CNNs, placement of batch normalization layer is directly between convolutional layer and activation layer. After normalization, activations are given scale and shift to adapt to searched representation. The scale γ and shift β are trainable parameters updated during back-propagation [29], [31].

2.5 Improving model performance

2.5.1 Ensemble methods

On the pathway to increase generalization capability of the model, reliance on more models is possible. Ensemble methods such as Bagging, which belongs to model averaging methods, use several models, which are trained separately and then tested on particular task. For classification tasks, the final predictions are collected, and the final prediction is stated by a majority vote from individual models. This technique is also used to decrease variance in the prediction.

Various patterns for ensemble formation exist. In the process, it is possible to change models, their components, activation functions and loss functions. Also, dataset variations can be changed. It is possible to use the same dataset on all models, use random subsets with or without replacement from original dataset, use subsets with or without repetition etc. Even if the dataset and individual model structure is invariant in the ensemble, it is possible for ensemble parts to produce partially independent errors by random weight initialization, different hyperparameters, or random mini-batch initiation.

By contrast to averaging methods, boosting methods, such as AdaBoost, is aiming to sequentially connect and thus, produce empowered ensemble from generally weaker models. This type of approach preferrable, when main goal in creating an ensemble is to reduce the bias of the combined estimator [21] [32].

2.5.2 Hyperparameter optimization

Building a solid model is always accompanied with hyperparameter tuning. Every model has several parameters, and the purpose is to find the optimal combination to improve performance.

The first option is to create a set of values for every parameter and perform a brute-force combination of all. This approach is called grid search and despite being simple and parallelizable, it is computationally expensive and results are poor. Grid search is used sparsely [33].

Random search is a variation of grid search using random numbers from uniform distribution. Computational time is equal to grid search, however, provides higher performance [33].

Bayesian optimization is defining a probabilistic surrogate model, that includes prior distribution describing unknown objective function. Surrogate model is iteratively fitted to all observations of objective function to update the prior, which will provide more precise posterior distribution in parameter space. Acquisition function is then maximized to define the most suitable configuration of hyperparameters for the next iteration. The configuration is guided by trading off exploration (amount of uncertainty of the surrogate model) and exploitation (precision in model's prediction). This approach, with high number of iterations, can converge to optimum. Nevertheless, the drawback lies in inability to process categorical hyperparameters and, given algorithm nature, inability to be parallelized [34].

Successive halving, belong to bandit-based algorithms. It assumes initial amount of resources (time, iterations, data samples etc.) and number of model configurations. After one iteration, half of the worse performing models is discarded, and next iteration starts with double amount of resources. This process is repeated till maximum budget is reached or when only one model remains. The disadvantage of this approach is in number of configurations and initial resources trade-off, which was solved by Hyperband optimization. Hyperband method allocates resources among random configurations of hyperparameters and performs a grid search over the optimal culling threshold in the outer loop of successive halving. This approach showed solid performance, however, since it is model-free algorithm (based on random search), the effectivity is only moderate [35].

A successful combination of Hyperband and Bayesian optimization was performed in BOHB (Bayesian Optimization and HyperBand). Bayesian part is computed by a modification of tree Parzen estimator with multidimensional kernel density estimators. Hyperband begins the computation with deciding the number of configurations and resources, however, the random search in configuration selection is replaced with Bayesian optimization. In this fashion, algorithm benefits from strong anytime performance by fast improvements in the initial iterations, due to low fidelity successive halving in Hyperband. Strong final performance is then achieved by replacement of random search with Bayesian optimization. This approach further benefits from parallelization, high range of hyper parameter space dimensions, adapting to multitude of machine learning tasks and different hyperparameter types processing [36].

Many various approaches for hyperparameter optimization are available, including utilizing particle swarm in [37] or populations in [38].

3 Convolutional neural networks

3.1 Background

Convolutional Neural Networks (CNNs) are special kind of neural networks, that have proven to have significant success in processing grid-like data. Especially high-dimensional data e.g., time-lapse signals, images or videos. Traditional neural networks use matrix multiplication of input matrix with matrix of separate weights. CNNs are based on operation of convolution. Convolution is an operation of two functions, of which one is reversed and shifted [39]. For 2D input, such as images, 2D convolution is calculated from functions K (kernel) and I (image) and is defined as:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{3.1}$$

$S(i,j)$	2D convolution function of image axes i,j
$K * I$	Kernel and image functions. Asterisk denotes convolution operation
m, n	Kernel axes

From equation 3.1 is noticeable, that convolution uses flipped kernel. Since convolution is commutative, it is possible to flip the kernel again in relation to image. Outcome of this change is cross-correlation. For implementation in machine learning, these two operations are equivalent. In fact, several machine learning libraries implemented cross-correlation instead of convolution [21].

CNNs are valued for **sparse connectivity** and **weight sharing** attributes. In traditional Fully Connected Neural Network (FCNN), each neuron has separate weight and process information from every neuron from the previous layer and forwards it to every neuron in the following layer. Sparse connectivity is achieved by using smaller kernel (typically 3×3, 5×5, 7×7...), compared to image. Spatial dimensions (width and height) of the kernel inform about **receptive field** of the input image, processed by output neuron.

The contrast between FCNN and CNN network in the context of their receptive field is depicted for one-dimensional case in Fig. 3.1.

This leaves us with a few advantages. Input from small receptive field is important, because it is possible to capture small specific features e.g., edges, which are located in different parts of the image. This is reducing the total number of weights and also the number of calculated operations, which diminishes computational time.

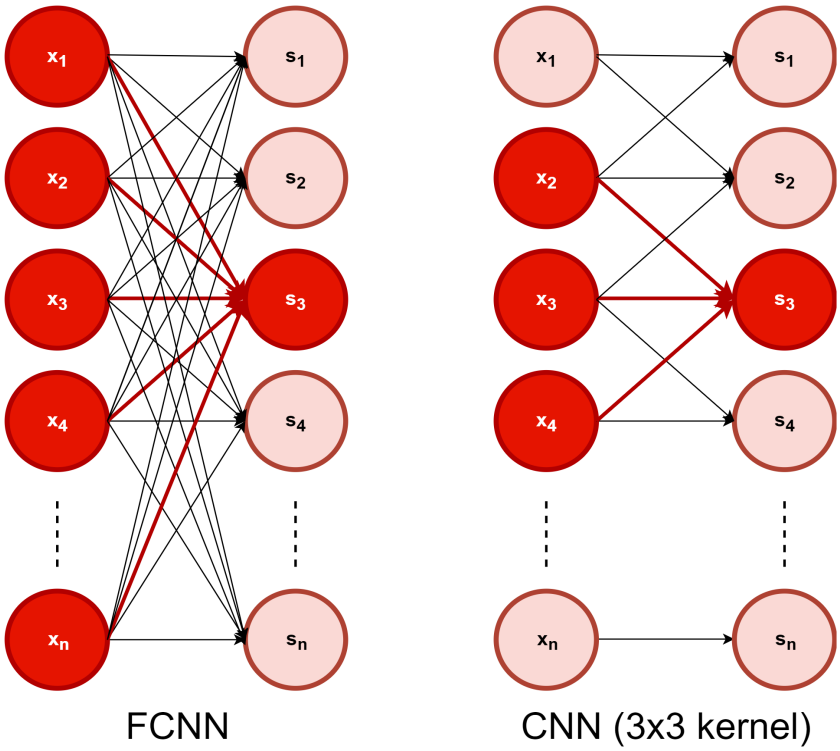

Fig. 3.1: Receptive field of output neuron s3. Highlighted are neurons contributing to the receptive field. In FCNNs (left), the receptive field of output neuron $s3$ is defined by size of the data, and thus, n (n^2 for image) input neurons. In CNNs (right) the receptive field is k, defined by $k \times k$ kernel, thus, only three (9 for image) input neurons affect output neuron $s3$.

As mentioned above, in FCNN each neuron has its separate weight, so the number of weights is determined by the number of neurons. CNNs, however, have shared weights, which reduces the number of all weighs in the network, by allowing them to be used for other neurons from the layer. This dramatically reduces memory requirements for weight storage during computation. Reducing the number of weights prevents overfitting, and thus, is a form of regularization [21]. Weight sharing also enable, that when an interesting feature in the image is discovered, this feature can be recognized at any other location in the image [39].

For instance, the computational demand for FCNN networks and CNNs in a single layer is provided. An input RGB image (C) has height (H) = 600, width (W) = 500, and the number of neurons in the layer (N) is 100. Number of total weights for FCNN network is defined as $H \times W \times C \times N = 90$ million weights to be stored. On the CNNs side, if kernel 3x3 ($M \times N$) is used, and the number of searched features in the image (F) is 2, the final number of weights is defined as $M \times N \times C \times F = 54$ weights.

3.2 Building blocks

3.2.1 Convolutional layer

Convolutional layer is basis of CNNs. As mentioned above, it comprises filter (kernel), which is convolved with the input image to form an output image, that is referred to as a feature map. The learning parameters in this layer are values of the kernel. Kernel is multiplied with region of the same dimensions in the input image or feature map and sum is performed. In order to calculate whole feature map, the kernel moves one pixel vertically or horizontally over the input, till a whole image is scanned.

The step of the kernel movement can also be greater than one pixel. Therefore, the movement can be extended by **stride**, which is a hyperparameter. Higher stride results in sub-sampled output feature map, which moderates pose and scale invariance of the objects [29]. Kernel is most of the time square-shaped ($F \times F$), therefore, input feature map is square-shaped as well ($D \times D$). After denoting stride as S, the output feature map dimension can be then calculated as follows, while $\lfloor\rfloor$ denotes floor operation:

$$D' = \lfloor \frac{D - F + S}{S} \rfloor \tag{3.2}$$

In some tasks including segmentation or de-noising, constant spatial dimensions after convolution or even bigger may be ensured. In these tasks, denser pixel-level predictions are needed. Moreover, deeper architectures can be proposed, since spatial dimensions of feature maps are not reduced along depth of the architecture. One of the approaches is **zero-padding**, when zeroes are added around the input feature map. In general, zero-padding extend the size of input feature map in order to acquire specific dimension of output feature map [29]. Calculation of output feature map with zero-padded input feature map and stride of two is shown in Fig. 3.2.

After introduction of P as padding, it is possible to extend the equation 3.2 for calculation of output feature map dimensions to:

$$D' = \lfloor \frac{D - F + S + P}{S} \rfloor \tag{3.3}$$

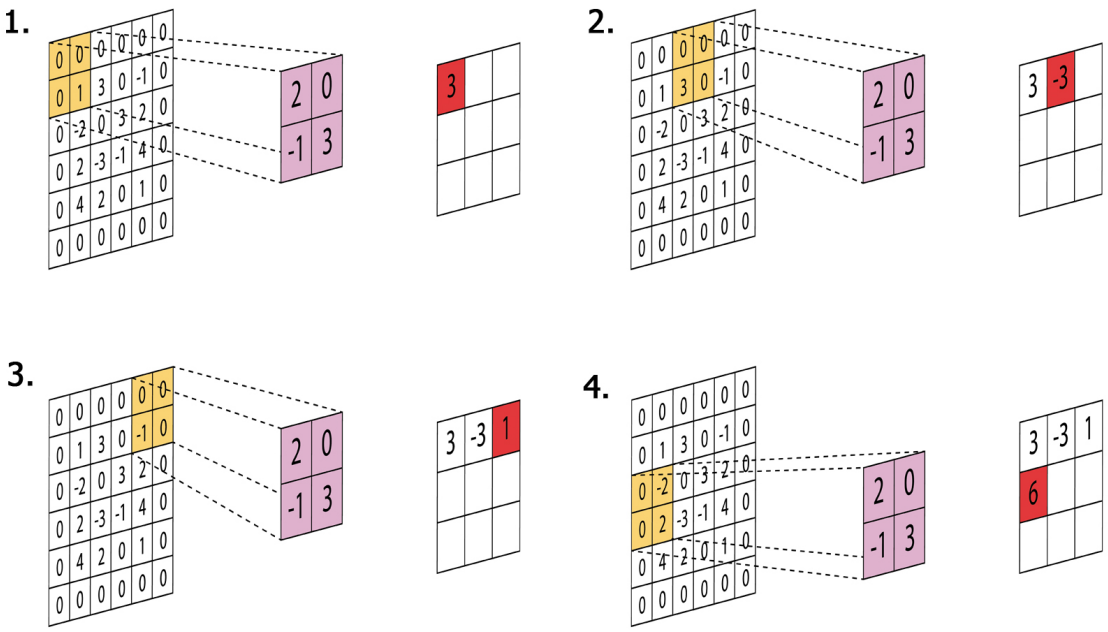

Fig. 3.2: Process of feature map obtention with zero-padding and stride of 2. Input image's (orange) size is 6×6, kernel's (violet) is 2×2. Kernel will move in nine steps to obtain final feature map (red), with size of 3×3.

Zero-padding convolution is generally divided into three categories:

1. *Valid Convolution.* Zero-padding is not involved. Convolution is performed from "valid" positions of input feature map. Output feature map's dimension D is reduced by F-1.
2. *Same Convolution.* Zero-padding is performed with respect to stride and kernel size. Output feature maps' dimension will be "same" as the input feature map's dimension.
3. *Full Convolution.* Zero-padding is performed in a manner, where in extremes (corners) at least one valid value will be present when convolution if calculated. This is tantamount to padding the input feature map with F-1 zeroes.

Regarding the receptive field of the kernel it is necessary to mention, that it relates to input image dimensions [29]. During stacking of convolutional layers along the depth of the architecture, it is essential to include **effective receptive field** into account. Effective receptive field of the current convolutional layer is a function of all receptive fields from previous convolutional layers used in the architecture. Effective receptive field for N convolutional layers involving stride and different kernel sizes is defined as:

$$RF_{eff}^{n} = RF_{eff}^{n-1} + ((f_n - 1) * \prod_{i=1}^{n-1} s_i), \qquad n \in [2, N] \tag{3.4}$$

RF_{eff}^{n}	Effective receptive field of current convolutional layer
RF_{eff}^{n-1}	Effective receptive field of previous convolutional layer
f_n	Kernel size of current layer
s_i	Stride of all previous layers

As a demonstration of effective receptive field, assume two filters 5×5 and 3×3, both with stride 1. Equation 3.4 can calculate the effective receptive field in relation to input image as 7×7. If no padding is added in the architecture, the final output feature map will have its spatial dimensions reduced by the size of effective receptive field [29].

As was shown previously, if the same kernel size is used, effective receptive field is increasing linearly (two 3×3 filters returns effective receptive field of size 5×5, three 3×3 filters size 7×7, four 9×9 etc.). Possible solution to exponentially magnify effective receptive field is to expand kernel size in every convolutional layer. This approach increases number of weights with each layer and thus, with deep networks is computationally demanding. Dilated convolution was introduced. This approach extends receptive field size with the same number of weights. Dilated convolution introduces parameter δ, that defines the gap between the weights of the kernel. These empty spaces in the kernel are set to zeroes [29].

For instance, assume three convolutional layers with dilated convolution with parameter δ equal to one, two and three, respectively. First layer's kernel size is 3×3 and $\delta = 1$. Its effective receptive field is therefore 3×3. Second layer's kernel has the same number of weights as the first, however, with δ of 2, its kernel size is increased to 5x5. From equation 3.4, effective receptive field is equal to 7×7. Third layer's kernel has again same number of weights and the δ factor of three makes the kernel size expand to 7×7. By usage of equation 3.4 again, the final effective receptive field is 13×13.

Possible disadvantage of this approach is, that several positions of the input feature map are disregarded based on the dilation factor. However, in applications such as labeling or segmentation, moreover segmentation of images in high-resolution, it is required to assemble wider contextual relationships with bigger receptive fields [29].

Parameters and hyper parameters

As mentioned before, elements which are going through learning process are called parameters. They are weights and biases. In CNNs, it is the kernel parameters and biases, that are learnable in the training process. However, there are also hyper-parameters. Hyperparameter is any attribute of the neural network, that can be controlled and set before the training process. In convolutional layers they are:

1. *Number of kernels in the layer*
2. *Spatial dimensions of the kernels*
3. *Stride*
4. *Dilation factor*
5. *Padding*

3.2.2 Activation layer

After convolutional layer comes the activation in the form of activation layer. The parameters from feature map act as an input to activation layer. Activation layer applies specific non-linear function point-by-point on the parameters, which also transforms them into small range (usually $[0; 1]$ or $[-1; 1]$). This is desirable, because during learning process, some values can increase rapidly and create huge data range. Output of activation layer has the same dimensions as its input and there are no hyperparameters to be set. Activation function decides, whether the neuron will fire or otherwise [29]. In order to successfully accomplish this, activation function should offer some desirable features including:

1. *Differentiability.* This is necessary feature. At least partial differentiability is required, since the training process is calculated via gradient descent.
2. *Avoiding gradient vanishing.* With depth of the architecture of neural network comes the problem of gradient vanishing. Consequence of this problem happens during back-propagation of the gradients, when small values of the gradient are multiplied along the architecture's depth. In result, first layers are learning very slowly, even stop the learning process and compromise prediction metrics.
3. *Zero-centering.* Activation function's output must be symmetrical at zero. Otherwise, gradients will be shifted to that direction.
4. *Computational efficiency.* Activation functions are calculated after each convolutional layer on huge number of weights [40].

Common activation functions used in deep learning are listed in [41] and shown in the Fig. 3.3.

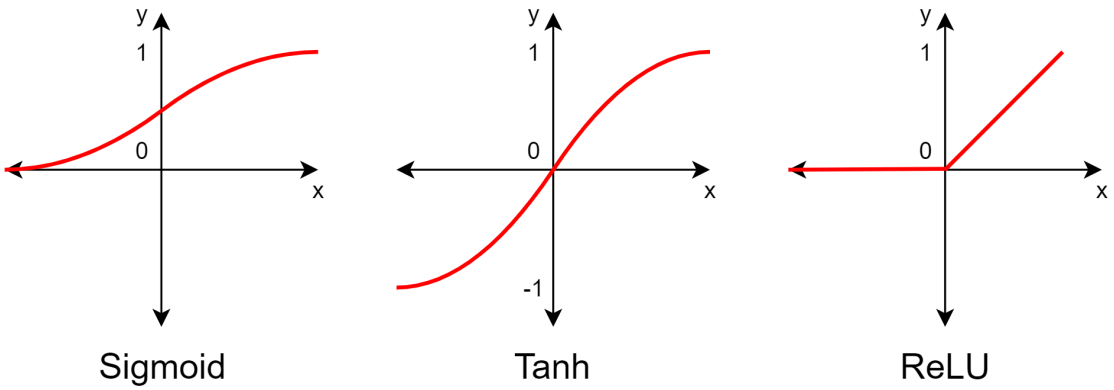

Fig. 3.3: Activation function representations used in deep neural networks, where x axis stands for input parameter value and y axis for output parameter value. ReLU is the most common.

1. *Sigmoid function.* Transforms output to range $[0; 1]$. It is rarely used after convolutional layers for causing vanishing gradient. It is computationally difficult and not zero-centered. Nevertheless, it is used as a final layer's activation, which transforms output feature map values to probabilities. It is defined as:

$$f_{sigm}(x) = \frac{1}{1 + e^{-x}} \tag{3.5}$$

2. *Hyperbolic tangent.* Transforms output to range [-1;1]. Compared to sigmoid, it is zero-centered, however, computationally more demanding. Definition is given as:

$$f_{tanh}(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.6}$$

3. *Rectified Linear Unit.* ReLU transforms negative values to 0 and positive leaves unchanged. It is quick to compute, solves gradient vanishing and prevents saturation. It is popular among CNNs. Disadvantage is its dying ReLU problem, when negative values are zeroed. This is solved by ReLU modifications. Definition of ReLU is:

$$f_{ReLU}(x) = max(0, x) \tag{3.7}$$

4. *Noisy ReLU.* Sample form Gaussian distribution with zero mean and variance is added to positive inputs. Definition is:

$$f_{n-ReLU}(x) = max(0, x + \epsilon) \qquad \epsilon \sim \mathcal{N}(0, \sigma(x)) \tag{3.8}$$

5. *Leaky ReLU.* Improves dying ReLU problem by downscaling negative input by small leak factor α. It is defined as:

$$f_{l-ReLU}(x) = max(\alpha x, x) \tag{3.9}$$

New activation functions have been developed. For instance, Swish, developed by Ramachandran et.al. in [42], or its computationally effective version Hard-Swish [40].

3.2.3 Pooling layer

After convolution and activation, it is preferable to use pooling. Pooling layer applies certain function on input activation map's neighborhood, to combine them. Most used function is maximum or average, which process rectangular region in feature map. Pooling layer downsamples the input feature map's spatial dimensions and helps to obtain moderate invariance to scale transformations or translations of the image. This is beneficial, when augmentation of dataset is performed to increase generalization ability. No parameter learning occurs in pooling layer. Its hyperparameters include stride and spatial size (most common two or three) [21].

3.2.4 Fully connected layer

Fully connected layer is equivalent to the layers present in FCNNs. Each neuron of fully connected layer is densely interconnected with all neurons in the previous layer.

These layers within CNNs are typically situated at the end, with softmax activation. Their purpose is to learn a non-linear relationship of high-level features and classify them to the corresponding classes. However, there are architectures, that implement fully-connected layers at intermediate locations within the architecture, such as Network-in-Network architecture proposed by Lin et al. [43].

3.2.5 Transposed convolution layer

This layer is heavily used in encoder-decoder CNN architectures. While convolutional layers reduce spatial dimensions (downsample feature maps), transposed convolution layers upsample feature maps. This operation enables the architecture to restore dimensions of original input image. There are other methods for upsampling the image, such as interpolation or max-unpooling, however, transposed convolution creates desired feature map via learnable parameters of the kernel. Parameters of transposed convolution layer are updated over iterations to provide more precise upsampling of input feature maps. Sometimes this layer is incorrectly referred to as a deconvolution layer. In practice, the kernel forms a Toeplitz matrix K, which is transposed and then multiplied with a vectorized input feature map x. The equation follows:

$$y = K^T x \tag{3.10}$$

Visually, transposed convolution can be understood with imputing null values between input feature map's values and padding of input feature map, along with reversed filter values. The desired size of output feature map is achieved by amending these hyperparameters. Regarding implementation, it is more efficient to calculate transposed convolution with the equation 3.10, rather than with imputing and padding with zeroes [29].

4 CNN architectures for semantic segmentation

The field of semantic segmentation has been experiencing a grand improvement with expanding possibilities, which deep learning offers. Implementation of deep learning algorithms proficiently improved understanding the scene, as well as reinforced current methods for semantic segmentation. Great advancements were achieved with deep neural networks, especially convolutional neural networks [44].

Since development of AlexNet in 2012 [45], numerous researchers began developing CNN based algorithms. AlexNet also popularized ReLU activation function in its architecture. In 2013, network-in-network structure was developed [43], with implementation of global average pooling, which decreased tendency to overfitting. In 2014, ImageNet database was subjected to VGGNet and GoogLeNet, with the outcome of accuracy improvement [46]. Later versions used Inception module with 1×1 convolutions, to decrease computational requirements. Successful incorporation of residual connections, batch normalization and Inception module further improved the performance [31] [47]. Fixed-length representations generation with various input image's size or scale, together with increase in generalization accuracy is possible with spatial pyramid pooling [48].

First efforts to use deep learning approaches for semantic segmentation was aimed at training classification networks e.g., VGG and then adjusting the model for segmentation. These approaches, however, could not sufficiently describe image's semantics with given number of layers. Fully connected layers were representing the issue with high quantity of parameters.

These layers were removed in in Fully Convolutional Networks (FCNs) [49]. Architectures, such as FCN-32, FCN-16 or FCN-8 were developed. With removed fully connected layers, inference of an image was significantly accelerated. Another benefit lied in generation of feature maps for images with various resolutions. Additionally, FCNs proposed skip connections, which were transferring local information between unconnected layers. This information is generally lost by dropout or pooling layers. Skip connections were later incorporated into encoder-decoder architectures, such as U-Net or SegNet. Performance of FCN on a PASCAL VOC 2012 dataset was 62.2 % mIoU.

U-Net is a popular FCN based architecture, widely used for medical and biological imaging [50]. It is based on encoder-decoder structure, where encoder progressively extracts features and downsamples the input image and decoder progressively restores the spatial dimensions of the feature map via up-convolution while further convolutional layers are applied. U-net also utilize skip-connection features by con-

catenating the feature map from one level of encoder to the same level of decoder. This allows the decoder to recover spatial information lost by down-sampling, as well as enables feature reusability for stabilization of convergence during training [51]. Performance of U-Net on a PASCAL VOC 2012 dataset was 72.7 % mIoU.

SegNet was initially developed to solve intelligent robots and autonomous driving, but has been used for the biological image segmentation as well [52]. The architecture is based on encoder-decoder structure, similar to U-Net. When the down-sampling occurs with max-pooling operation, SegNet saves information about element position and restores the image in decoder part accordingly. Final layer applies softmax as final activation compared to U-Net's 1x1 convolution. Performance of SegNet on a PASCAL VOC 2012 dataset was 59.9 % mIoU.

DeepLab architecture utilizes expanding effective receptive field with atrous convolution. It enables control over the resolution of feature maps, along with capturing wider contextual information without introduction of additional parameters. DeepLab's final layer is fully connected conditional random field (CRF). CNNs usually lack solid segmentation performance in the area of class borders. CRF is addressing this problem with construction of multi-class interaction map, which contributes with global contextual information and finer label localization [53]. DeepLab.v2 further introduced atrous spatial pyramid pooling (ASPP). This feature allows convolutional layers to incorporate a set of kernels with several sampling rates and effective receptive fields, which capture multi-scale contextual information. DeepLab.v3 has further optimized hyperparameters of ASPP layer and CRF layer was omitted for faster computation. DeepLab.v3+ finally enhanced DeepLab.v3 with encoder-decoder structure modified to perform atrous convolutions. Performance of DeepLab, DeepLab.v2, DeepLab.v3 and DeepLab.v3+ on a PASCAL VOC 2012 dataset was 66.4, 79.7, 85.7 and 87.7 % mIoU, respectively.

GCN is an architecture based on residual blocks incorporated in ResNet architecture as encoder with combination with FCN-4 for segmentation. Authors proposed Global Convolutional Network (GCN), which utilizes usage of large kernels to combine low- with high-level features, while extracting multi-scale feature maps along the network. This arrangement is extending valid receptive field. Additionally, a boundary refinement residual block is connected in the network [54]. Performance of SegNet on a PASCAL VOC 2012 dataset was 83.6 % mIoU.

EMANet architecture incorporates attention modules in its architecture. Attention modules generally bring focus to important or foreground context of the scene. Output of attention module is attention map, which is pixel-wise multiplied with the feature map, to highlight useful context. Attention brings global contextual information of the scene, nevertheless, it is computationally demanding. EMANet introduces Expectation-Maximization Attention (EMA) module with more compact

computation bases, that are iteratively computed via expectation-maximization algorithm. EMA's output is resilient to input variance and can be quickly incorporated to another CNN architectures [55]. Performance of SegNet on a PASCAL VOC 2012 dataset was 88.2 % mIoU.

Results achieved with particular architectures were acquired from [56]. There are several other methods challenging semantic segmentation including utilization of recurrent neural networks [57] or even transformers [58].

In practical part of this thesis, U-Net architecture with batch normalization and transposed convolutions was chosen. Researching for a method suitable for practical part, there has not been an article discussing any deep segmentation algorithm processing cell on grid images from SEM in cryo conditions. One of the reasons to choose this architecture was, that it achieved solid results in numerous cell or cell component segmentation tasks [59] [50]. Besides already incorporated skip-connections in encoder-decoder structure, proposed architecture was enhanced by batch normalization layers for benefits discussed in section 2.4.4 and with decoder upsampling via transposed convolution for benefits discussed in section 3.2.5.

5 Thesis Results

Practical purpose of this thesis is to implement, train, test and optimize appropriate deep convolutional neural network for images acquired from cryo-electron microscope.

Practical solution was implemented in Python with Pytorch library for neural networks. Google Colab Pro was used as a developing environment mainly for the benefit of connection to the hosting GPU and its usage in calculation. Implementation code for the model was created and adjusted on the basis of GitHub commit available at [60]. Hyperparameter optimization was performed via Bayesian optimization algorithm available at GitHub repository [61]. Calculations were executed via GPU NVIDIA P100 with 16 GB memory and with RAM with 27.4 GB of usable memory. Algorithms created for the purposes of this thesis together with final model are available for public at: `https://github.com/Pinc0/Masters-thesis-U-Net`.

The dataset was created, consisting of 120 grayscale images of yeast cells from cryo-SEM. Images contain cells on a cultivation mesh, with occasional ice crystals and mesh deformations. Acquisition of images was performed in transversal plane, in different resolutions and saved with different bit depth. The dataset was annotated manually, with binary masks containing information about the cells and background. Images were randomly ordered and further divided into two subfolders. One for testing with 20 images and the other for training and validation with 100 images. The variability of features in the dataset is presented in Fig. 5.1. Images vary in their mean intensities, rotations, amount of ice crystals and mesh deformations, shapes of the cells, as well as intensity profiles of particular cells or cell clusters.

5.1 Algorithm

Proposed architecture is widely used in image segmentation tasks. It is encoder-decoder type convolutional neural network based on U-Net architecture. A variation of this architecture with modifications described in Fig. 5.2 was implemented.

In data preprocessing, images were resized 512×512. In training phase, Dice loss was selected for error calculation, due to its consideration of both local and global information, while ignoring true negative background in the image. AdamW was chosen as optimizer for its benefits described in section 2.3.1. Learning rate scheduler was added. Weight initialization was chosen as proposed by Xavier in [14] and biases were set to zero. Forward-pass and backward-pass is followed by thresholding output probabilities from sigmoid with 0.5 to acquire final mask of prediction.

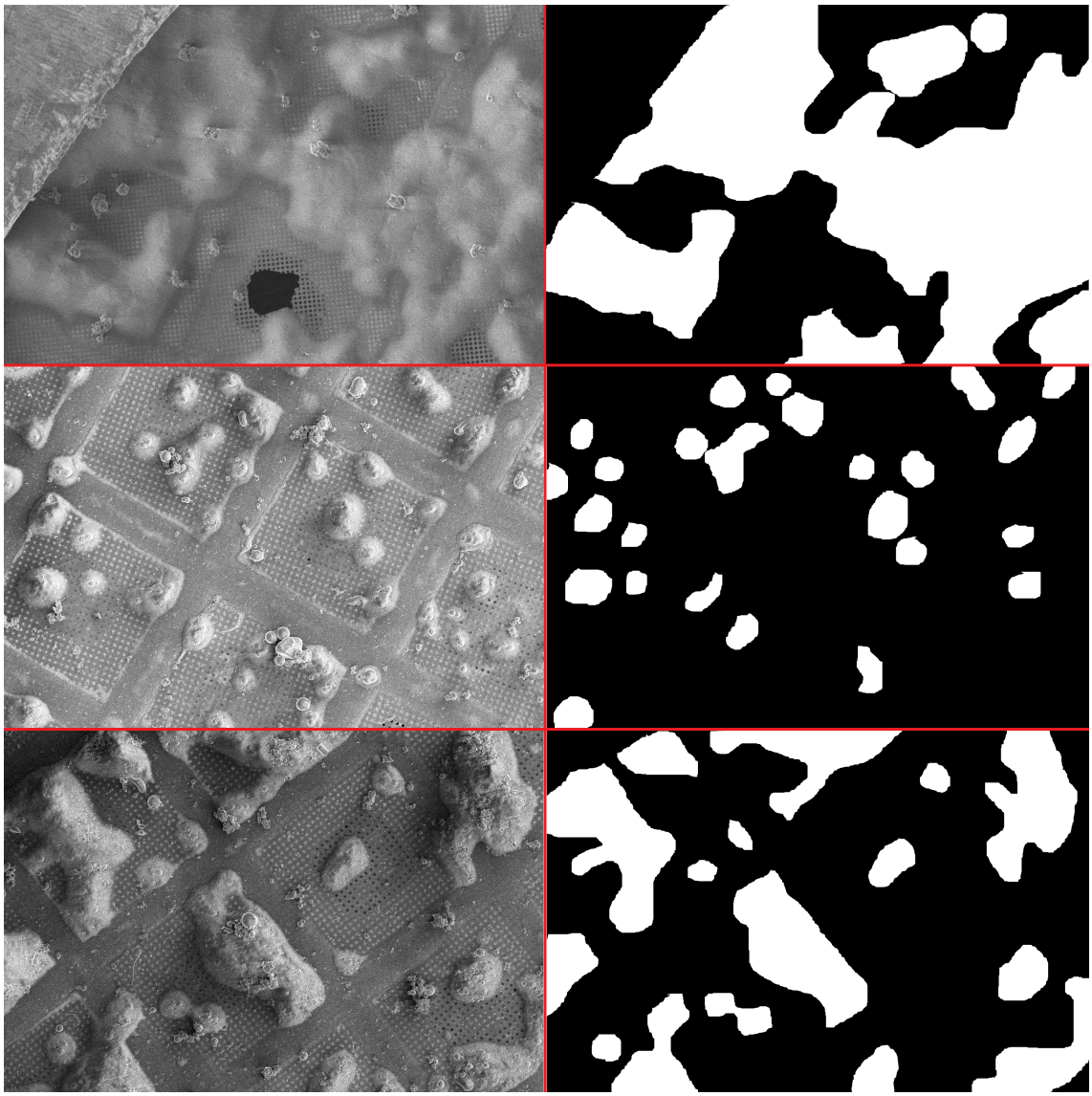

Fig. 5.1: Variability of features in the dataset. Images on the left were intentionally picked to demonstrate dataset feature variability. Images on the right corresponds to their ground truth masks.

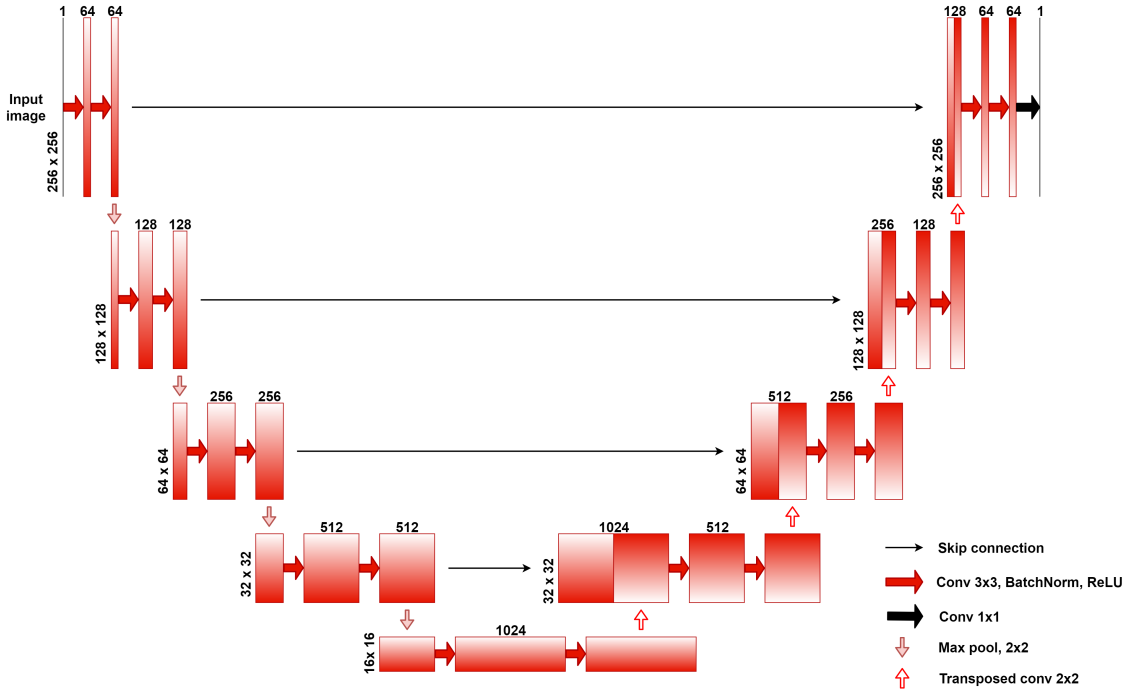

Fig. 5.2: Proposed architecture. The difference compared to original U-Net architecture [50] is in input size reduced to 256×256 pixels, as well as output size, upconvolution replaced with transposed convolution and incorporation of batch normalization layer after convolutional layers.

Training and validation dataset was split into five folds for the purposes of cross-validation with 80/20 ratio. All five models with the lowest Dice loss on the validation set were saved after the training. These models were then evaluated on 20 images from the test set. Reproducibility of experiments was ensured, by setting random number generator to the same seed. Since the task was semantic segmentation, metrics that incorporate true negative values was omitted. Three metrics were chosen: Recall, Precision and Dice score.

5.2 Experiments

The first experiment to perform was to train the model with empirically set hyperparameters. The hyperparameters were:

Train. and valid. batch size	20
Number of epochs	30
Initial learning rate	10^{-3}
Learning rate decay	0.1
AdamW weight decay	10^{-8}

Tab. 5.1 shows evaluation metrics of the models with their validation loss and the index of epoch, in which was this loss minimal during training phase.

	Model 1	**Model 2**	**Model 3**	**Model 4**	**Model 5**	**Model avg.**
Training loss	0.335	0.527	0.534	0.749	0.376	0.504
Lowest loss epoch	30	19	13	10	8	-
Avg. recall	0.612	0.627	0.604	0.441	0.599	0.577
Avg. precision	0.612	0.569	0.581	0.663	0.631	0.611
Avg. Dice score	0.586	0.574	0.560	0.484	0.577	0.556

Tab. 5.1: Evaluation of the first experiment. Table merges 5-fold cross-validation results, i.e., reached training Dice loss, epoch, in which the loss was recorded with average recall, precision and Dice score achieved during testing on all five folds and their average.

It is noticeable, that the outcome of this experiment produced models with high Dice loss, in extreme case for the model 4 at 0.749. Lowest loss epoch row suggests, that in four models, the training was sufficient early before the end of 30 epochs. Given the outcome, it may be assumed, that models were not capable to generalize with training on such a small dataset.

One way to increase generalization ability of a model is to increase the number of training samples, which was added via training and validation dataset augmentation. This approach, as was mentioned in section 2.4.1, can also prevent overfitting of the model and improve its performance. As a result, every image from training and validation dataset produced 10 augmented images.

It is usual, that images in general process of image acquisition from cryo-electron SEM microscope are rotated, zoomed, flipped, blurred, or contain specific noise. Augmentation was performed via random crop to half-sized image, random flipping horizontally or vertically and random 90-degree rotation. Since the software for image acquisition produces images, that are already focused and occasionally blurred images are discarded, blur was omitted from augmentation. Noise augmentation was also omitted, since the data were collected from different microscope systems and obtention of all point spread functions of noise to mimic real noise could be challenging. The hyperparameters were set the same, as in the first experiment.

Tab. 5.2 summarizes the outcome of training on augmented dataset.

	Model 1	**Model 2**	**Model 3**	**Model 4**	**Model 5**	**Model avg.**
Training loss	0.168	0.156	0.156	0.153	0.171	0.161
Lowest loss epoch	28	29	30	30	30	-
Avg. recall	0.692	0.721	0.729	0.747	0.664	0.711
Avg. precision	0.762	0.766	0.773	0.760	0.753	0.763
Avg. Dice score	0.713	0.729	0.737	0.742	0.688	0.722

Tab. 5.2: Evaluation of the augmentation experiment. Table merges 5-fold cross-validation results, i.e., reached training Dice loss, epoch, in which the loss was recorded with average recall, precision and Dice score achieved during testing on all five folds and their average.

Table shows, that the training Dice loss decreased heavily, even in the model 4. From the lowest loss epoch row may be suggested, that training of the models ended close to the end of 30 epochs, which implies underfitting. Thus, adding more epochs may improve the performance of the models (it will be done during hyperparameter optimization). Last column informs about approximately 0.15 improvement of the average recall, precision and Dice score, compared to the experiment with no augmentation. Thus, experiments will continue with augmented training and validation dataset containing 1000 images.

Skip-connections, as was explained in chapter 4, have advantages of enabling feature reusability and stabilize training and convergence. In this experiment, skip-connections were removed from the architecture, to assess the level of impact on model's performance. The hyperparameters were set the same, as in the first experiment.

Tab. 5.3 summarizes the outcome of removing skip-connections from the architecture.

	Model 1	**Model 2**	**Model 3**	**Model 4**	**Model 5**	**Model avg.**
Training loss	0.160	0.165	0.168	0.182	0.166	0.168
Lowest loss epoch	30	29	30	29	28	-
Avg. recall	0.733	0.710	0.727	0.684	0.693	0.709
Avg. precision	0.766	0.756	0.757	0.719	0.773	0.754
Avg. Dice score	0.735	0.720	0.726	0.682	0.717	0.716

Tab. 5.3: Evaluation of the skip-connection experiment. Table merges 5-fold cross-validation results, i.e., reached training Dice loss, epoch, in which the loss was recorded with average recall, precision and Dice score achieved during testing on all five folds and their average.

By removing skip-connections from the architecture, the last column shows, that there has been only slight depression of the average model metrics in the order of thousandths. This experiment shows, that despite the general advantages of using skip-connections, their removal has depressed average model evaluation metrics only marginally. However, the difference in average metrics of individual models ranged from thousandths to hundredths. Experiments will continue with skip-connections incorporated into the architecture.

Another way to increase generalization performance of the model is to perform hyperparameter optimization. There are several hyperparameters optimization techniques, as was mentioned in section 2.5.2, from which Bayesian optimization was chose for its ability to converge to optimal solution, simple implementation and because no categorical hyperparameters were optimized (only integers and floats). Since the model 4 from augmentation experiment was the most successful in average metrics on test set, it was decided to optimize hyperparameters with training and validation dataset split equivalent to split in this model.

Process was executed on above mentioned graphic card with 30 iterations. One iteration lasted approximately 80 minutes and the target for maximalization was maximal Dice score on validation dataset. As was observed from previous experiments, models with the lowest validation loss were acquired very close to the total number of epochs, which implies underfitting. Additionally, validation curve did not reached plateau in convergence. The parameter space of the optimization was thus set to:

Train. and valid. batch size	8–16
Number of epochs	60–140
Initial learning rate	10^{-4}–10^{-2}
Learning rate decay	0.1–1
AdamW weight decay	10^{-9}–10^{-5}

First run of the optimization was interrupted in 17th iteration as a cause of disconnection from hosting GPU. Google Colab Pro platform only allows hosting of GPU up to 24 hours. However, the best iteration of the trial maximized Dice score to 0.809, compared to 0.742 achieved in augmentation experiment in Tab. 5.2. Hyperparameters, which achieved this result are summarized below:

Train. and Valid. batch size	16
Number of epochs	140
Initial learning rate	10^{-2}
Learning rate decay	1
AdamW weight decay	10^{-8}

These hyperparameters indicated, that the best result can be achieved disregarding learning rate scheduler. Despite time limitation of GPU computing, another optimization was performed, with modified hyperparameter space, according to results from the first run.

Second optimalization was again interrupted in 19th iteration for the same cause as the first. Nevertheless, during this run, the iteration with most successful combination of hyperparameters achieved worse mean Dice score, as in the first run.

Following hyperparameter suggestion from the first optimization run, five models were trained resulting from 5-fold cross-validation. The results were refined compared to augmentation experiment from Tab. 5.2 and are summarized in Tab. 5.4.

	Model 1	**Model 2**	**Model 3**	**Model 4**	**Model 5**	**Model avg.**
Training loss	0.122	0.118	0.173	0.159	0.114	0.136
Lowest loss epoch	130	137	133	135	137	-
Avg. recall	0.801	0.800	0.797	0.762	0.853	0.803
Avg. precision	0.758	0.827	0.823	0.868	0.774	0.810
Avg. Dice score	0.763	0.806	0.804	0.805	0.806	0.797

Tab. 5.4: Evaluation of hyperparameter optimization. Table merges 5-fold cross-validation results, i.e., reached training Dice loss, epoch, in which the loss was recorded with average recall, precision and Dice score achieved during testing on all five folds and their average.

It is noticeable, that average loss on validation fold has decreased, compared to augmentation experiment, except model 4 which shows slight increase. Average evaluation metrics has increased from 0.711, 0.763 and 0.722 into 0.803, 0.810 and 0.797, for recall, precision and Dice score, respectively. As optimization was performed on fourth fold for validation (model 4), the biggest improvement in fact, was shown in model 5, with almost two tenths increase in recall and over one tenth increase in Dice score. This could be caused by small amount of iterations with the hyperparameter optimization. Despite the fact, that suggested hyperparameters increased Dice target score for model 4, however, they were not yet specific for particular model, eventually these hyperparameters actually improved model 5 to a

higher extent. Lowest loss epoch row is again near the number of maximal epochs for the models.

Throughout all the experiments, the models reported precision metric higher compared to recall. Based on this, it can be claimed, that trained models are more susceptible to produce false negative prediction, rather than false positive. Average recall, precision and Dice score of model 5 was comparable also to models 2 and 4, all reaching value of 0.811. However, model 5 achieved the lowest Dice loss during validation. As model 2 and 4 were more successful regarding average precision, model 5, on the other hand, was most successful in average recall. After discussion with consultant of this thesis, it was decided to mark model 5 as final for utilization in the company. Results of model 5 after hyperparameter optimization were further analyzed. Fig. 5.3 depicts boxplots of testing images of model 5 for each metric.

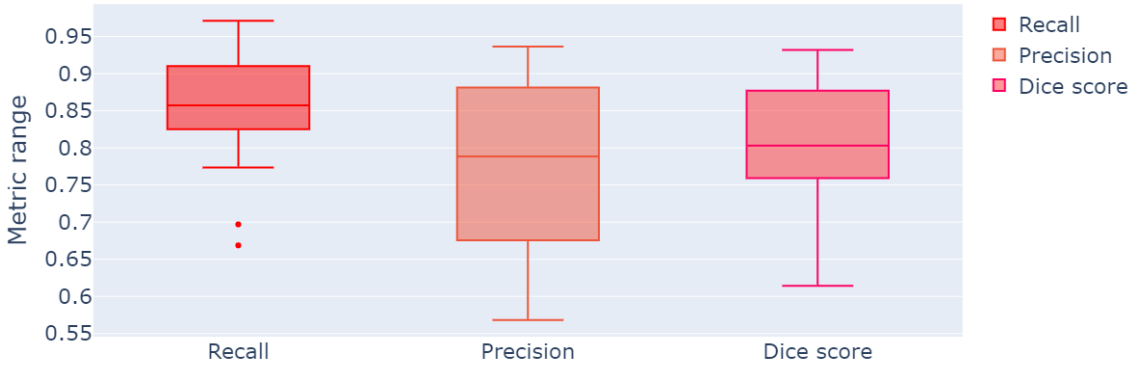

Fig. 5.3: Boxplot metrics from testing of model 5. Recal boxplot with two outliers in red, precision boxplot in orange and Dice score in red.

Boxplot of recall has the shortest range and interquartile range from all metrics. This behavior is attributable only to this model (other models showed the opposite behavior). Spread of false negatives across models is thus lower than false positives. Precision, on the other hand showed wider range and interquartile range for this model. All boxplots show, that more than 50 % of all metrics lies higher than 0.75 % of metric range. Also, no testing image resulted with any metric below 0.55. Tab. 5.5 presents final results of the model 5.

Learning curves of the final model are depicted in Fig. 5.4. Both training and validation curves exhibit exponential improvement. This shows healthy convergence of the network. Validation curve showed prominent spikes during training. They are caused by training with mini-batches. Generally, spikes occur when learning rate is too high and predictions oscillate in the parametric space toward optimum. Another cause could have been small batch size compared to training dataset size, which in

	Final model
Training loss	0.114
Lowest loss epoch	137
Avg. recall	0.853
Avg. precision	0.774
Avg. Dice score	0.806

Tab. 5.5: Final model's results. Table shows reached training Dice loss, epoch, in which the loss was recorded with average recall, precision and Dice score achieved during testing of final model.

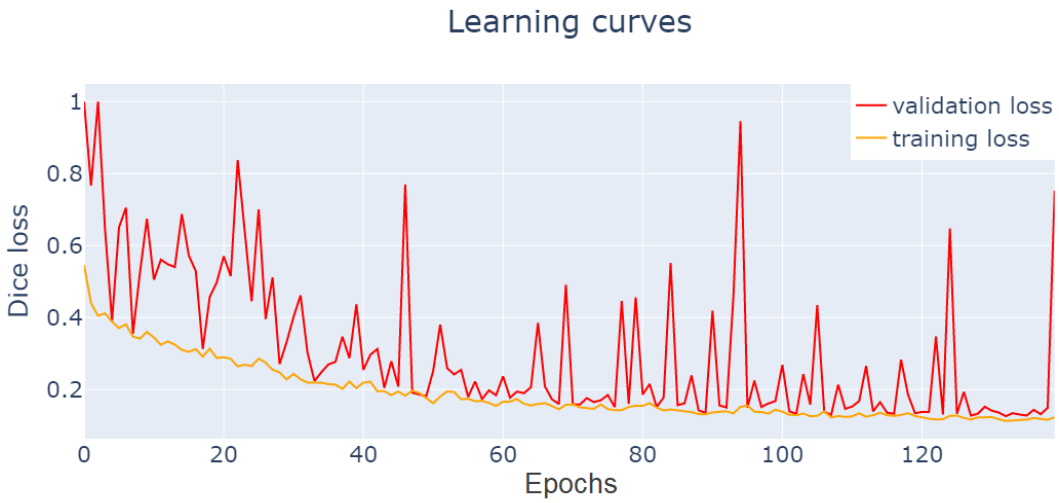

Fig. 5.4: Final model's testing and validation learning curves. Curves show average dice loss during training epochs. Validation loss in red shows prominent spikes. Training loss is showed in orange.

this model's case was 1/50. Spikes can also manifest when training data contain high amount of features or noise. It may be suggested, that the most probable cause was small mini-batch to training size ratio. After 70th epoch the convergence of the curves reached almost plateau, however, still decreasing until final epoch.

Successively, it was decided to perform analysis of the final model's results and how effective it was, coping with test set variability. Outliers will be examined and behavior of the model on testing images will be described. Following figure depicts the most salient testing images in overlay with cell borders from mask image (yellow) and output of the model (red). These results were consulted with a specialist in the company and the following observations were made.

Top left image in Fig. 5.5, depicts, that the model was able to identify all cells from the grid. This image also shows a remarkable feature of this model, when borders surrounding a cell were predicted more authentically by the network, than

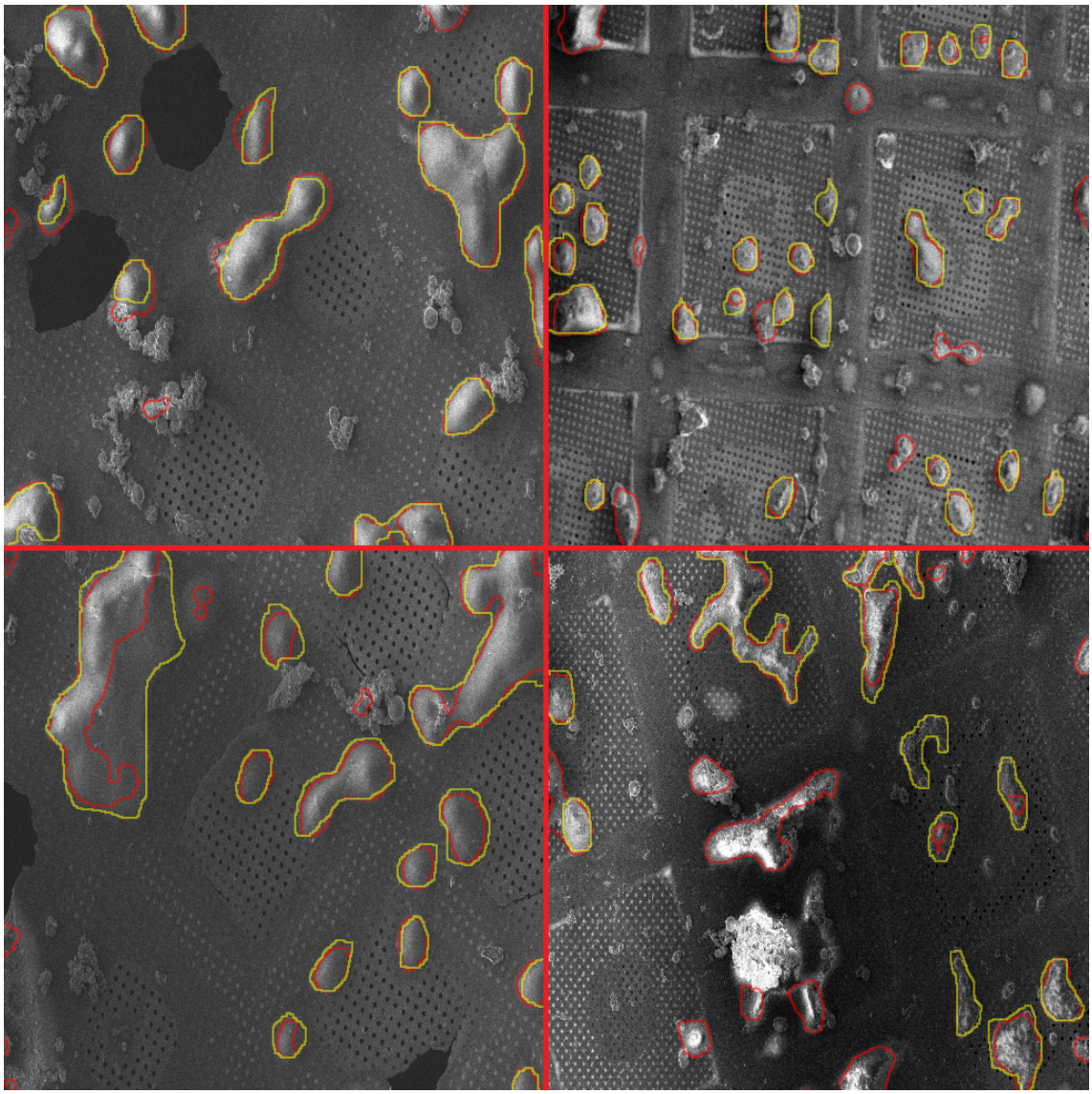

Fig. 5.5: Salient testing image results of the final model. Images depict four testing images in overlay with cell borders from mask image (yellow) and output of the model (red).

with the manual annotation. Recall, precision and Dice score for this image were 0.926, 0.833, and 0.877, respectively.

Top right image belongs to the minority of images added to the dataset, which contained mostly single individual cells. The focus from the company was to annotate mostly bigger cell clusters, than a single individual cells. However, image shows, that some of the cells not marked in label image was predicted correctly, even though are evaluated as false positive. This had an overall impact on depression of precision metric. Correct prediction of unlabeled small cells was present in most of the testing images. Recall, precision and Dice score for this image were

0.774, 0.676, and 0.722, respectively.

Bottom left image shows again, that all labeled cells were identified and segmented. However, the big cell cluster on the left was segmented approximately in the area of high charge from electron beam (area with high intensities), omitting the rest of the cluster with low intensities. Some false positive areas are present here as well. Recall, precision and Dice score for this image were 0.697, 0.937, and 0.799, respectively.

Bottom right image shows, that the central object was recognized as a cell, even though most of the cell's surface is covered with ice. Despite the fact, that the model successfully ommited the object below as an ice crystal, some objects are indistinguishable for a specialist to assess as a cell or ice crystal. Image also shows, that in this case the network was not able to segment small cell clusters, which contained very small charge from electron beam (having very low intensities, similar with the mesh). Nevertheless, there are cells at the bottom, that was evaluated as false positives, even though they are in fact small cells, which was only not labeled in ground truth images. Additionally, these false positive regions successfully evaded ice crystal contamination. Recall, precision and Dice score for this image were 0.669, 0.568, and 0.614, respectively.

Algorithms created for the purposes of this thesis together with final model were provided to the research and development department of Thermo Fisher Scientific Brno, and are available at: `https://github.com/Pinc0/Masters-thesis-U-Net`.

6 Discussion

Semantic cell segmentation utilizing deep learning algorithms is a widespread approach. Nevertheless, as far as was researched, there has not been an article discussing any deep segmentation algorithm processing cell on grid images from SEM in cryo conditions. Practical part of this thesis is thus a first proposal to cope with this challenge without an option to compare the results.

The dataset feature variability is influenced by a variety of factors. To begin with quality of cultivated cells on the mesh, their intracellular fluid volume, handling during sample preparation, cryo-cooling time, contamination of the sample, formation of cells or cell clusters, fusion of ice crystals with the cells, thickness of metal coating layer, electron beam charge, optical defects of the microscope system and more.

Annotation of the dataset should be done with utmost precision and deliberation by an expert. Moreover, sharp distinction level should be adopted to precisely separate ice crystal contamination from cells. One of the main contributors to mitigation of segmentation results was that mostly, only cell clusters in training images were annotated and small individual cells were not. However, the final proposed model has shown its ability to detect even unannotated cells correctly and with a tendency to separate ice crystal contamination from cells.

It was proven, that if the dataset is limited, data augmentation of the dataset via naturally present transformations, such as cropping, rotation or flipping, can substantially improve model's generalization ability.

Throughout the experiments, it has been shown, that high-end graphic card is essential in computation. Together with high graphical and RAM memory requirements, also data storage for the models should be created. Trained models with proposed architecture contain 31,044,289 parameters with 335 MB size. Training of final model lasted around 90 minutes, which w.r.t modern DL architectures is really quick. Nevertheless, computational time requirements can be elevated, especially with sequential hyperparameter optimization, such as Bayesian.

Results of the experiment with dataset containing original 120 images without augmentation were poor, improper for utilization in practice. Even though skip-connections have an irreplaceable place with encoder-decoder type of architectures, it was suggested that implementation of skip-connections to the architecture for this task contributed only negligibly.

Bayesian hyperparameter optimization can converge, with the high number of iterations, to the global extreme. However, this approach encountered the limitation in the form GPU hosting time. Nevertheless, even the small number of iterations has indicated more accurate hyperparameters and the most suitable model. The biggest

improvement in hyperparameters occurred disregarding learning rate scheduler and with higher learning rate.

Among the positive qualities of the final model, after discussion with expert, a notable property is its ability to distinguish a proper cell without actual annotation in the label image. Secondly, in most of the cases the cell boundaries, that were predicted by the model are actually more genuine and natural-looking, compared to annotated dataset. The model was also partially successful in recognition of ice crystal contamination incorporated into cells clusters.

One of the observed disadvantages of final model was in prediction of small individual cells, rather than bigger cell clusters. This occurred partially due to major presence of large, annotated cell clusters in the training images. Another inability of the model to distinguish the cells occurred with small cell clusters with low level of charge from electron beam, so they contained small variability of features and their intensity profile was comparable with that of a background mesh. The partial distinction was present in bigger cell clusters with subarea containing high level of charge from electron beam.

Results of this thesis cannot be compared due to lack of a similar segmentation method. Even though the advisor from the company considered contribution of this thesis as sufficient, the potential to expand this work is present. Expansion of the dataset with new real images is a likely scenario that will definitely improve current model. Presence of the contamination in form of ice crystals or mesh deformations can expand this work to a multi-class segmentation. New architecture, with different optimizer can be used. More recent hyperparameter optimization method can be utilized. Final model achieved the lowest validation in epoch near to total number of epochs, which indicates, that the convergence did not completely stopped and there is a space for improvement with a fine tuning of hyperparameters. Tuning only on five hyperparameters was performed. However, β_1 and β_2 in AdamW optimizer or stride, padding and kernel sizes of the convolutional layers was omitted, which is an option to cover in the future optimization.

Conclusion

Cryo-electron microscopy has its irreplaceable position in analysis of biological samples. Early parts of this thesis explained fundamental principles of cryo-electron microscopy imaging, what is essential in sample preparation and its stages, together with illustration of the procedures used in practice. In the following chapter, a concept of neural networks and machine learning principles required for proper understanding the behavior of models and their dynamics was illustrated. Followingly, the concept of convolutional neural networks was explained. Depicted was an improvement in incorporation of CNNs to neural architectures and uniqueness of individual layers used in conjunction with them. Followingly, a research of deep neural networks used in practical applications for semantic image segmentation was elaborated. Based on this knowledge the final architecture was chosen.

It was confirmed, that high-end graphic card, high RAM memory and disk storage are essential coping with the challenges of deep learning. An annotated dataset with 120 images from cryo-SEM modality was created in collaboration with Thermo Fisher Scientific Brno. In practical solution, several experiments were performed. It was illustrated, that such a small dataset with high feature variability produced unsatisfactory results for practical implementation. It was shown that augmentation of training dataset is essential in boosting model's performance. Skip-connections in the architecture were considered, as with negligible effect in the overall performance. Incorporating hyperparameter optimization, was depicted, that it is an essential step in generation of successful model, with regard to numerous hyperparameters of the architecture. Learning curves are the first indicator of performance, which were considered during hyperparameter optimization in order to achieve satisfactory results.

The detailed evaluations of results and final model were provided with the analysis of outliers and notable testing image results. This was further discussed with an expert and final findings regarding optimal hyperparameters, together with description of advantages and disadvantages of proposed model were stated. Discussion continued with explanation of factors affecting quality of the model and possible extensions for future work.

Results of this thesis were provided to research and development department of Thermo Fisher Scientific Brno. Algorithms created for the purposes of this thesis together with final model are available for public at: `https://github.com/Pinc0/Masters-thesis-U-Net`.

Bibliography

[1] GOLDSTEIN, I. Joseph. Scanning electron microscopy and X-ray microanalysis. 3rd ed. New York: Kluwer, 2003. ISBN 0-306-47292-9.

[2] ZHANG, Yu-Jin. An Overview of Image and Video Segmentation in the Last 40 Years. ZHANG, Yu-Jin, ed. Advances in Image and Video Segmentation [online]. IGI Global, 2006, 2006, s. 1-16 [cit. 2021-04-16]. ISBN 9781591407539. Dostupné z: doi:10.4018/978-1-59140-753-9.ch001

[3] An Introduction to Electron Microscopy: History of Electron Microscopy [online]. Thermo Fisher Scientific, 2019 [cit. 2021-04-16]. Dostupné z: www.fei.com/introduction-to-electron-microscopy/history/

[4] Transmission Electron Microscopy vs Scanning Electron Microscopy [online]. Thermo Fisher Scientific, 2021 [cit. 2021-04-16]. Dostupné z: https://www.thermofisher.com/cz/en/home/materials-science/learning-center/applications/sem-tem-difference.html

[5] BOZZOLA, John a Lonnie RUSSELL. Electron Microscopy: Principles and Techniques for Biologists. Second Edition. Sudbury: Jones and Bartlett Publishers, 1999. ISBN 0–7637-0192-0.

[6] GOLDSTEIN, Joseph, Dale E. NEWBURY, Joseph R. MICHAEL, Nicholas W. M. RITCHIE, John Henry J. SCOTT a David C. JOY. Scanning electron microscopy and X-ray microanalysis. Fourth edition. New York: Springer, [2018]. ISBN 978-1-4939-6674-5.

[7] Methods and Tools for Studying Cell Organization [online]. Course Hero, 2021 [cit. 2021-5-18]. Dostupné z: https://www.coursehero.com/sg/cell-biology/methods-and-tools-for-studying-cell-organization/

[8] NANAKOUDIS, Antonis. EDX Analysis with SEM: How Does it Work? [online]. Thermo Fisher Scientific, 2019 [cit. 2021-04-16]. Dostupné z: www.thermofisher.com/blog/microscopy/edx-analysis-with-sem-how-does-it-work/

[9] ESMON, Alex. The Physics of Ice: It All Begins with Nucleation [online]. Thermo Fisher Scientific, 2014 [cit. 2021-04-16]. Dostupné z: https://www.thermofisher.com/blog/biobanking/the-physics-of-ice-it-all-begins-with-nucleation/

[10] VODÁREK, Vlastimil. FÁZOVÉ PŘEMĚNY. Ostrava: VŠB – Technická univerzita Ostrava, 2013. ISBN 978-80-248-3376-7.

[11] WAGNER, Felix R., Reika WATANABE, Ruud SCHAMPERS, et al. Preparing samples from whole cells using focused-ion-beam milling for cryo-electron tomography. Nature Protocols [online]. 2020, 15(6), 2041-2070 [cit. 2021-4-18]. ISSN 1754-2189. Dostupné z: doi:10.1038/s41596-020-0320-x

[12] SINGH, Satya P., Lipo WANG, Sukrit GUPTA, Haveesh GOLI, Parasuraman PADMANABHAN a Balázs GULYÁS. 3D Deep Learning on Medical Images: A Review. Sensors [online]. 2020, 20(18) [cit. 2021-4-19]. ISSN 1424-8220. Dostupné z: doi:10.3390/s20185097

[13] PEDRYCZ, Witold a Shyi-Ming CHEN, ed. Deep Learning: Concepts and Architectures [online]. Cham: Springer International Publishing, 2020 [cit. 2021-4-19]. Studies in Computational Intelligence. ISBN 978-3-030-31755-3. Dostupné z: doi:10.1007/978-3-030-31756-0

[14] GLOROT, Xavier a Yoshua BENGIO. Understanding the difficulty of training deep feedforward neural networks. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics [online]. 2010, (9), 249-256 [cit. 2021-4-30]. Dostupné z: http://proceedings.mlr.press/v9/glorot10a.html

[15] HE, Kaiming a Xiangyu ZHANG. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. CoRR [online]. 2015 [cit. 2021-4-30]. Dostupné z: http://arxiv.org/abs/1502.01852

[16] ISACSSON, Martin. Choosing and Customizing Loss Functions for Image Processing [online]. 2021 [cit. 2021-04-16]. Dostupné z: https://towardsdatascience.com/choosing-and-customizing-loss-functions-for-image-processing-a0e4bf665b0a

[17] ZHAO, Hang a Orazio GALLO. Loss Functions for Neural Networks for Image Processing [online]. 2018 [cit. 2021-4-30]. Dostupné z: https://arxiv.org/pdf/1511.08861.pdf

[18] PHAN, Trong a Kazuma YAMAMOTO. Resolving Class Imbalance in Object Detection with Weighted Cross Entropy Losses [online]. 2020 [cit. 2021-4-30]. Dostupné z: https://arxiv.org/abs/2006.01413

[19] SALEHI, Seyed a Deniz ERDOGMUS. Tversky loss function for image segmentation using 3D fully convolutional deep networks. CoRR [online]. 2017 [cit. 2021-4-30]. Dostupné z: https://arxiv.org/abs/1706.05721

[20] RUDER, Sebastian. An overview of gradient descent optimization algorithms. CoRR [online]. 2016 [cit. 2021-4-30]. Dostupné z: http://arxiv.org/abs/1609.04747

[21] GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. Deep Learning [online]. MIT Press, 2016 [cit. 2021-04-16]. Dostupné z: http://www.deeplearningbook.org

[22] SHARMA, Sagar. Epoch vs Batch Size vs Iterations [online]. 2017 [cit. 2021-04-16]. Dostupné z: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9

[23] SUTSKEVER, Ilya, et al. On the importance of initialization and momentum in deep learning. In: International conference on machine learning [online]. PMLR, 2013. p. 1139-1147 [cit. 2021-04-16]. Dostupné z: http://proceedings.mlr.press/v28/sutskever13.html

[24] DUCHI, John; HAZAN, Elad; SINGER, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research, 2011, 12.7.

[25] ZEILER, Matthew. ADADELTA: An Adaptive Learning Rate Method. CoRR [online]. [cit. 2021-4-30]. Dostupné z: https://arxiv.org/abs/1212.5701

[26] KINGMA, Diederik a Jimmy BA. Adam: A Method for Stochastic Optimization [online]. 2017 [cit. 2021-5-5]. Dostupné z: https://arxiv.org/abs/1412.6980

[27] KINGMA, Diederik a Jimmy BA. Decoupled Weight Decay Regularization [online]. 2017 [cit. 2021-5-5]. Dostupné z: https://arxiv.org/abs/1412.6980

[28] REDDI, Sashank, Satyen KALE a Sanjiv KUMAR. On the Convergence of Adam and Beyond. CoRR [online]. 2019 [cit. 2021-5-5]. Dostupné z: http://arxiv.org/abs/1904.09237

[29] KHAN, Salman, Hossein RAHMANI, Syed Afaq Ali SHAH a Mohammed BENNAMOUN. A Guide to Convolutional Neural Networks for Computer Vision. Synthesis Lectures on Computer Vision [online]. 2018, 8(1), 1-207 [cit. 2021-04-16]. ISSN 2153-1056. Dostupné z: doi:10.2200/S00822ED1V01Y201712COV015

[30] PRECHELT, Lutz. Early Stopping — But When? MONTAVON, Grégoire, Geneviève B. ORR a Klaus-Robert MÜLLER, ed. Neural Networks: Tricks of the Trade [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, 2012, s. 53-67 [cit. 2021-04-16]. Lecture Notes in Computer Science. ISBN 978-3-642-35288-1. Dostupné z: doi:10.1007/978-3-642-35289-8_5

[31] IOFFE, Sergey a Christian SZEGEDY. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift [online]. 2015 [cit. 2021-04-16]. Dostupné z: https://arxiv.org/abs/1502.03167

[32] PEDREGOSA, Fabian. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011, (12), 2825 to 2830.

[33] HUTTER, Frank, Lars KOTTHOFF a Joaquin VANSCHOREN, ed. Automated Machine Learning [online]. Cham: Springer International Publishing, 2019 [cit. 2021-5-11]. The Springer Series on Challenges in Machine Learning. ISBN 978-3-030-05317-8. Dostupné z: doi:10.1007/978-3-030-05318-5

[34] SNOEK, Jasper, Hugo LAROCHELLE a Ryan ADAMS. Practical Bayesian Optimization of Machine Learning Algorithms [online]. 2012 [cit. 2021-5-11]. Dostupné z: https://arxiv.org/abs/1206.2944

[35] LI, Lisha a Kevin JAMIESON. Efficient Hyperparameter Optimization and Infinitely Many Armed Bandits. CoRR [online]. 2016 [cit. 2021-5-11]. Dostupné z: http://arxiv.org/abs/1603.06560

[36] FALKNER, Stefan, Aaron KLEIN a Frank HUTTER. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. CoRR [online]. 2018 [cit. 2021-5-11]. Dostupné z: http://arxiv.org/abs/1807.01774

[37] LI, Yaru a Yulai ZHANG. Hyper parameter estimation method with particle swarm optimization. CoRR [online]. 2020 [cit. 2021-5-11]. Dostupné z: https://arxiv.org/abs/2011.11944

[38] JADERBERG, Max a Valentin DALIBARD. Population Based Training of Neural Networks. CoRR [online]. 2017 [cit. 2021-5-11]. Dostupné z: http://arxiv.org/abs/1711.09846

[39] VASILEV, Ivan a Daniel SLATER. Python Deep Learning. Second Edition. Birmingham: Packt Publishing, 2019. ISBN 978-1-78934-846-0.

[40] VANDIT, Jain. Everything you need to know about "Activation Functions" in Deep learning models [online]. 2019 [cit. 2021-04-16]. Dostupné z: https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253

[41] HOON CHUNG, SUNG JOO LEE a JEON GUE PARK. Deep neural network using trainable activation functions. In: 2016 International Joint Conference on Neural Networks (IJCNN) [online]. IEEE, 2016, 2016, s. 348-352 [cit. 2021-3-16]. ISBN 978-1-5090-0620-5. Dostupné z: doi:10.1109/IJCNN.2016.7727219

[42] RAMACHANDRAN, Prajit. Searching for Activation Functions [online]. 2017 [cit. 2021-04-16]. Dostupné z: https://arxiv.org/abs/1710.05941

[43] LIN, Min, Qiang CHEN a Shuicheng YAN. Network In Network [online]. 2014 [cit. 2021-04-16]. Dostupné z: https://arxiv.org/abs/1312.4400

[44] GARCIA-GARCIA, Alberto, Sergio ORTS-ESCOLANO, Sergiu OPREA, Victor VILLENA-MARTINEZ, Pablo MARTINEZ-GONZALEZ a Jose GARCIA-RODRIGUEZ. A survey on deep learning techniques for image and video semantic segmentation. Applied Soft Computing [online]. 2018, 70, 41-65 [cit. 2021-04-16]. ISSN 15684946. Dostupné z: doi:10.1016/j.asoc.2018.05.018

[45] KRIZHEVSKY, Alex, Ilya SUTSKEVER a Geoffrey E. HINTON. ImageNet classification with deep convolutional neural networks. Communications of the ACM [online]. 2017, 60(6), 84-90 [cit. 2021-04-16]. ISSN 0001-0782. Dostupné z: doi:10.1145/3065386

[46] SZEGEDY, Christian, Wei LIU a Yangqing JIA. Going Deeper with Convolutions [online]. 2014 [cit. 2021-04-16]. Dostupné z: https://arxiv.org/abs/1409.4842

[47] SZEGEDY, Christian a Sergei IOFFE. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning [online]. 2016 [cit. 2021-04-16]. Dostupné z: https://arxiv.org/abs/1602.07261

[48] HE, Kaiming a Xiangyu ZHANG. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition [online]. 2014 [cit. 2021-04-16]. Dostupné z: https://arxiv.org/abs/1406.4729

[49] LONG, Jonathan, Evan SHELHAMER a Trevor DARRELL. Fully Convolutional Networks for Semantic Segmentation. CoRR [online]. 2015 [cit. 2021-03-17]. Dostupné z: http://arxiv.org/abs/1411.4038

[50] RONNEBERGER, Olaf, Philipp FISCHER a Thomas BROX. U-Net: Convolutional Networks for Biomedical Image Segmentation [online]. 2015 [cit. 2021-03-17]. Dostupné z: https://arxiv.org/abs/1505.04597

[51] ADALOGLOU, Nikolas. Intuitive Explanation of Skip Connections in Deep Learning [online]. 2020 [cit. 2021-03-17]. Dostupné z: https://theaisummer.com/skip-connections/

[52] DAIMARY, Dinthisrang, Mayur Bhargab BORA, Khwairakpam AMITAB a Debdatta KANDAR. Brain Tumor Segmentation from MRI Images using Hybrid Convolutional Neural Networks. Procedia Computer Science [online]. 2020, 167, 2419-2428 [cit. 2021-03-17]. ISSN 18770509. Dostupné z: doi:10.1016/j.procs.2020.03.295

[53] CHEN, Liang a George PAPANDREOU. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. CoRR [online]. 2016 [cit. 2021-03-17]. Dostupné z: http://arxiv.org/abs/1606.00915

[54] PENG, Chao a Xiangyu ZHANG. Large Kernel Matters - Improve Semantic Segmentation by Global Convolutional Network. CoRR [online]. 2017 [cit. 2021-03-17]. Dostupné z: http://arxiv.org/abs/1703.02719

[55] LI, Xia a Zhisheng ZHONG. Expectation-Maximization Attention Networks for Semantic Segmentation. CoRR [online]. 2019 [cit. 2021-03-17]. Dostupné z: http://arxiv.org/abs/1907.13426

[56] ULKU, Irem a Erdem AKAGÜNDÜZ. A Survey on Deep Learning-based Architectures for Semantic Segmentation on 2D images. CoRR [online]. 2019 [cit. 2021-3-17]. Dostupné z: http://arxiv.org/abs/1912.10230

[57] SHUAI, Bing a Zhen ZUO. DAG-Recurrent Neural Networks For Scene Labeling. CoRR [online]. 2015 [cit. 2021-3-17]. Dostupné z: http://arxiv.org/abs/1509.00552

[58] STRUDEL, Robin a Ricardo GARCIA. Segmenter: Transformer for Semantic Segmentation [online]. 2021 [cit. 2021-3-17]. Dostupné z: https://arxiv.org/abs/2105.05633

[59] GARCIA, Ruben a Joan SEGURA. MicrographCleaner: A python package for cryo-EM micrograph cleaning using deep learning. Journal of Structural Biology [online]. 2020 [cit. 2021-5-17]. ISSN 1047-8477. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1047847720300642

[60] BATERIWALA, Malav. Unet-Segmentation-Pytorch-Nest-of-Unets. GitHub [online]. 2020 [cit. 2020-11-02]. Dostupné z: https://github.com/bigmb/Unet-Segmentation-Pytorch-Nest-of-Unets

[61] NOGUEIRA, Fernando. Bayesian Optimization. GitHub [online]. 2014– [cit. 2021-04-11]. Dostupné z: https://github.com/fmfn/BayesianOptimization

List of symbols, quantities and abbreviations

AI	Artificial Intelligence
BGD	Batch Gradient Descent
CNN	Convolutional Neural Network
DL	Deep Learning
$FCNN$	Fully Connected Neural Network
FCN	Fully Convolutional Network
GPU	Graphics Processor Unit
ML	Machine Learning
NN	Neural Network
RGB	Red Green Blue
SEM	Scanning Electron Microscope
SGD	Stochastic Gradient Descent
TEM	Transmission Electron Microscope