Lehrstuhl für Medizinische Biotechnologie
Technische Fakultät
Friedrich-Alexander-Universität Erlangen-Nürnberg

Design and Evaluation of a Deep Learning Approach to Quantify Immune Cell Infiltrates in Volumetric Autofluorescence Image Data

Master's Thesis

In the program Advanced Optical Technologies
to acquire the academic degree
Master of Science (M.Sc.)

Submitted by
Sergei Dobrovolskii
Matrikelnummer: 22745251

Erlangen, 22.09.2022

Supervisors:
PD Dr. Sebastian Schürmann

Abstract

Design and Evaluation of a Deep Learning Approach to Quantify Immune Cell Infiltrates in Volumetric Autofluorescence Image Data

Immune cells infiltrating into tissue are an important factor in inflammatory processes. An automated quantification of immune cells in a tissue volume would therefore be of great value yet poses a very challenging task.
In this thesis, a Deep Neural Network is implemented to approach the problem of immune cell quantification in volumetric autofluorescence image data acquired from colon tissue with a multiphoton microscope. Deep neural networks may perform remarkably in tasks like picture segmentation and classification, but they need a lot of training data to do so. When images are three-dimensional, labeling training data becomes considerably more difficult. As an intermediate step to address this issue, a 3D cell simulation framework was developed in this thesis, which can create unlimited annotated datasets resembling real 3D image stacks for training and testing neural networks. Furthermore, a deep convolutional network for automated cell classification in tissue volumes was developed based on LinkNet architecture and evaluated using simulated datasets and real image stacks.

Zusammenfassung

Design und Evaluierung eines Deep-Learning Ansatzes zur Quantifizierung von Immunzellinfiltraten in volumetrischen Autofluoreszenz-Bilddaten

Immunzellinfiltration in Gewebe ist ein wichtiger Faktor in Entzündungsprozessen. Eine automatisierte Quantifizierung von Immunzellen in Gewebe ist daher von hohem Wert, stellt aber eine sehr schwierige Aufgabe dar.
Im Rahmen dieser Masterarbeit wurde das Problem mit Hilfe eines Deep Neural Networks adressiert. Deep Neural Networks sind eine der Techniken, die im Datenanalyseprozess verwendet werden. Diese Netzwerke können bei Aufgaben wie der Bildsegmentierung und -klassifizierung bemerkenswerte Leistungen erbringen, benötigen dafür jedoch viele Trainingsdaten. Wenn Bilder dreidimensional sind, wird die Kennzeichnung von Trainingsdaten erheblich schwieriger. Um dieses Problem anzugehen, ist ein Simulator für synthetische Daten erforderlich, um dieses Problem zu lösen. Es kann unbegrenzt kommentierte Daten für neuronale Netze erstellen, um deren Leistung zu testen. Diese Arbeit entwickelt einen Simulationsrahmen, vergleicht seine Ergebnisse mit tatsächlichen Stapeln, die mit einem Multiphotonenmikroskop aufgenommen wurden, und trainiert ein tiefes Faltungsnetzwerk, das diese künstlichen Daten verwendet, um Immunzellen zu zählen und zu klassifizieren.

Table of Content

Abbreviations

MPM	Multiphoton Microscope
CNN	Convolutional Neural Network
H&E	Hematoxylin and Eosin
RF	Random Forest
DTC	Decision Tree Classifier
FAD	Flavin Adenine Dinucleotide
TRITC	Tetramethylrhodamine
FITC	Fluorescein isothiocyanate
GPU	Graphical processing unit
PDF	Probability density function

1 Introduction

Inflammatory processes are very complex involving many different cell types and signaling molecules. For a better understanding of inflammation and refined diagnostics, microscopic imaging of cells in their three-dimensional tissue environment may play an important role. A robust automated classification and counting of cells in tissue volumes would be of great value here.

In general, the observation of experiment can be done via imaging modalities, such as conventional microscope, cell-counter, computed tomography or by naked eye. For investigation of small cells usually microscopy is a good modality. There is a plethora of different microscopic modalities, and in the scope of this project, a multiphoton microscope was used. It can establish three-dimensional image acquisition and utilize fluorescence for morphological sample understanding.

Acquired images themselves are useful at a first glance. The images can be examined by knowledgeable pathologists or biologists, and then a conclusion can be drawn from those observations. Because biology is a quantitative field of study, the significance of statistical analysis is something that must not be overlooked. To gain a deeper understanding of it, analysis is required to find more insights. Images need to be processed and analyzed before they can provide a better understanding of the experiment.

The problem of image processing and analysis can be solved using a variety of mathematical and software techniques. These techniques can either be traditional, such as thresholding, watershed, or region growing algorithms, or they can be addressed to more sophisticated techniques, such as decision tree, random forest, and support vector machines. All these methods are still in use today, but recent developments in deep learning have made it possible to perform image processing that is both more general and more concise.

The way these networks are working is not the same as for the traditional approaches. For Neural Networks no particular algorithm is being developed to perform image processing and classification, but network is being “trained” on datasets that consist of original images and the output that is required to get from the network given that image. The output is usually created manually, by hand, and this time-consuming process is named labeling. Network “learns”

mapping between input and output, and hence can be used on unseen data to do the same. This makes it possible to process a great variety of images, which makes it superior to any of the traditional methods.

These networks may demonstrate remarkable performance in tasks such as image segmentation and classification, but they require a substantial amount of training data to do so. When images contain a third dimension, labeling the training data becomes considerably more difficult. A data simulation tool is required to find a solution to this issue. It can generate an unlimited amount of annotated data for use in testing the performance of neural networks. This simulator must generate images that resemble real ones and a random assortment based on parameter settings. The advantage of using a simulator instead of manually annotated data is data with flawless annotations and unlimited dataset capacity. Any experiment of image annotation and augmentation can be done in a matter of minutes, but not weeks. The downsides of using simulated data could be low variation, bias, low level of complexity. These reasons not only raise questions in the scope of neural network training, but also encourages to develop and improve methods that will make simulated training data more similar to real.

In this thesis, a simulation framework for FAD cytoplasm fluorescence in volumetric data is developed, its results are compared to actual stacks acquired with a multiphoton microscope, and a deep convolutional network is trained to count and classify immune cells using the simulated data.

2 State of the Art

The human body is a highly complex system with a multitude of organs composed of tissues composed of billions of grouped cells. These cells interact with one another to maintain life in the body. As with any complex system, this one has a potential to fail; any disease can make life difficult or impossible. The initial step in treating any illness is diagnosis. In modern medicine, there are numerous imaging modalities, including X-ray, Ultrasound, Computed tomography, Optical coherence tomography, Magnetic resonance imaging, and Microscopy.

The purpose of microscopy is to visualize single cells and tissue. Typically, tissue samples must be fixed (with a substance such as formaldehyde) and stained prior to image acquisition. Hematoxylin and Eosin are the most common stains for brightfield imaging. The most common stains for fluorescence microscopy are DAPI, Hoechst, FITC, and TRITC. These are utilized to make tissue or cellular structures visible and distinguishable from other structures. Different microscopic modalities can deliver fluorescent volumetric data such as Multiphoton Microscopy or Confocal laser Microscopy.

Typically, cells are nonplanar and live in three dimensions. The examination under the coverslip is insufficiently precise and can conceal some cellular behavior. In addition, 2D in vivo imaging is not always possible, particularly when underlying cell cultures are concealed beneath the surface. To obtain more information from the tissue, volumetric images, also known as Z-stacks or stacks, must be captured. Multiple color channels are present in this three-dimensional image.

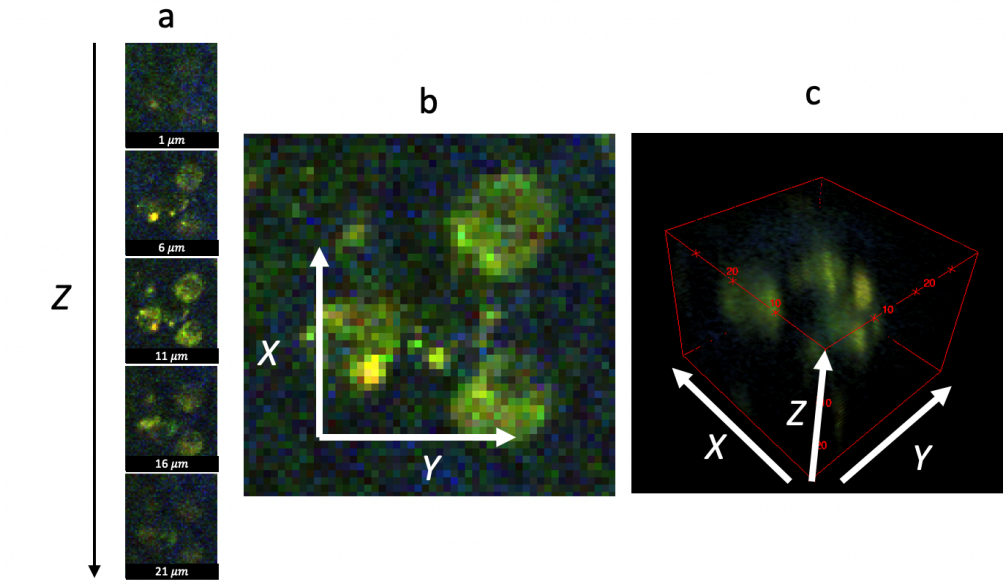

Figure 1: Representation of volumetric data. a: The montage of images acquired at different depths Z. b: Zoomed in a single image. c: 3D representation of a stack.

Stack allows seeing the cells in volume, which is oftentimes necessary, for example, direct observation of skin cells together with immune cells lying under skin layer (1). Volumetric data is more difficult to work with, due to the high memory consumption and visualization tradeoffs that we must do to observe the data.

2.1 Motivation

Any experiment in biological research consists of multiple steps, including hypothesis, subject preparation, acquisition, and analysis. In this section, the concepts of experiment subject, data acquisition method, and general data analysis will be explained.

2.1.1 Human immune cells in colon tissue

To a large extent, immune cells can be found everywhere; however, if inflammation takes place in a particular region, immune cells will congregate there, causing a shift in both the concentration and composition of immune cells in that region. The doctor can learn a lot about the inflammation process by looking at the different types of cells that are present in the affected area as well as the relative concentration of those cells.

It is undeniable proof that inflammation exists in the human colon tissue when there is a high number of immune cells present. However, information about the inflammatory process can be gathered not only from the presence of immune cells, but also from the cell type composition, concentration of those cells, and locations.

One of our groups (MBT FAU Erlangen) aims is to record three dimensional stacks in vivo. That also means investigation no stains could be applied to tissue, but natural fluorescence of FAD coming from cells cytoplasm could be used. It gives information about cell size, shape, and its nucleus size. This information has a potential to become useful for cells counting and classification.

After the fluorescence signal has been produced, it is essential to avoid conflating it with signals that are produced by other objects in observation volume that are not of interest. These other signals come from different objects. As a result, the location of immune cells is an essential component in the measurement of the immune infiltrate.

Imaging techniques such as multiphoton microscopy may be useful for determining the presence, quality, and quantity of immune cells in tissue.

2.1.2 Multiphoton Microscope

Multi-photon microscopy (MPM) is an efficient method that enables three-dimensional mapping of materials having a detectable nonlinear optical response, such as second harmonic generation (SHG), third harmonic generation (THG), or fluorescence caused by multiphoton absorption. MPM allows for the 3D observation of stained tissue and single cells with high resolution. By only exciting the focal plane, it reduces scattering from non-focal planes.

Fluorescence is the process that occurs when the fluorophore's electron absorbs the excitation photon and settles on a higher energy level within the molecule. The atom then returns to its ground state by emitting a photon of a different wavelength (**Figure 2A**). Two-photon excitation operates in the same manner as single-photon excitation but requires two incident photons with half the energy of the required photon. For this effect to occur, these photons must simultaneously strike the same atom. To achieve this infrequent occurrence, the photon density must be high. This condition is exploited to achieve a very small excitation volume without scattering from non-focal planes, which results in accurate volumetric data.

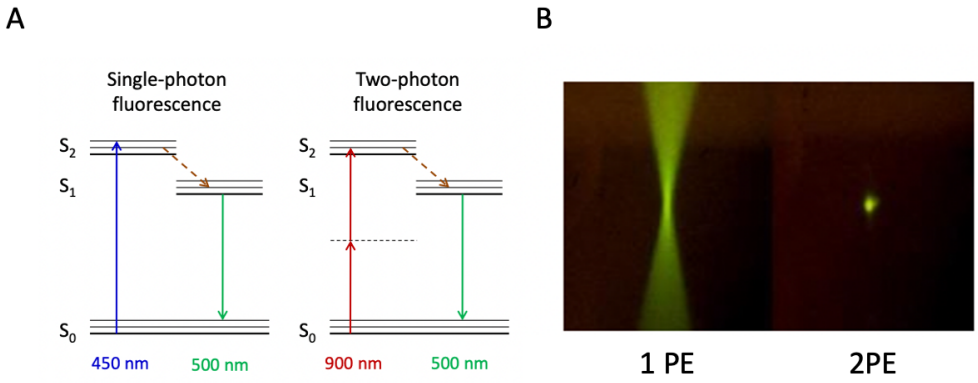

Figure 2: A: S. Schürmann, Department MBT, Multiphoton fluorescence energy diagram comparison of a single-photon (1 PE) fluorescence and two-photon (2 PE) fluorescence. In two-photon fluorescence, 2 photons of the energy twice lower than required are exciting electrons together. This is achieved by higher energy density which results in a higher probability of excitation events happening. B: The comparison of the excited volumes – 1 PE has a lot of exciting molecules out of the focal plane which results in worse image quality, compared to 2 PE.

The principle of this modality differs from the fluorescent microscope. In a fluorescence microscope, the excitation response of the fluorophore is linear, more excitation light - more fluorescence response. But with the two-photon excitation, it is different (**Figure 2B**). Excitation is visible only at the focal point – the place where the photon density will be the highest. MPM has been shown to have useful applications in areas such as the characterization of nonlinear materials, biological research, and the diagnosis of medical conditions.

2.1.3 Aims of Image Analysis

For the experiment conduction, the data is acquired by any given sensor (photo-multiplier tube in our case). This raw information provides a good view on the processes, structure, intensity, but to reveal a full potential and extract more information from them some analysis is required. To prove the hypothesis this information must be cleaned, distilled, and processed. All these tasks are faced by image analysis – the field of science and the mathematical toolset. Image analysis involves processing images into fundamental components to extract important information. It may involve tasks such as finding shapes, detecting edges, removing noise, counting objects, texture analysis, etc. achieved by means of mathematical operations, or dynamically, using machine learning approaches. In this Thesis, only the segmentation procedure for human immune cells in three-dimensional data will be explained.

2.1.4 Machine learning in image processing

Machine learning is a set of sophisticated mathematical operations performed on data to receive an expressive result from it. These algorithms are conventionally generalized in form of a pattern recognition pipeline (**Figure 3**). An electronic device, such as a camera, microphone, or microscope, is initially used to collect the data. The data is then saved and preprocessed. In the preprocessing step, data is filtered, improved, and prepared for subsequent processing. Next is a feature extraction, which uses a series of mathematical operations to extract representative features from the data. These characteristics may represent a streamlined version of the original data or generate new data, enabling algorithms to locate intra-data correspondences. For image processing these characteristics could be obtained using convolutional filters such as Gaussian, Laplacian, and Gradient, among others. Later, these characteristics are utilized for the so-called learning or training step. During training, the Algorithm attempts to identify the optimal split for classifying data with minimal error, based on extracted features.

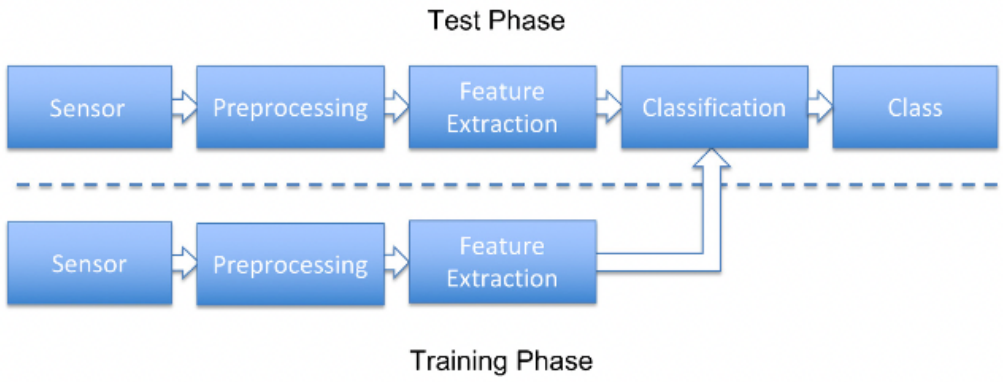

Figure 3: Typical pattern recognition pipeline (2). It is divided into two parts: The test phase and the Training phase. Sensor, preprocessing, and feature extraction steps are common for both phases.

These algorithms in image processing are conventionally divided into two groups: Classical and Deep Learning approaches. Differences between those groups can be compared using the pattern recognition pipeline (**Figure 3**):

- Classical machine learning approaches follow this pipeline. They require a small amount of data and small computational costs. Lack of generalization and precision.

- Deep Learning approaches bypass part of this pipeline by combining feature extraction with classification. There are no predefined feature extraction procedures in neural networks, and these procedures are estimated, during the training process. They require a substantial amount of labeled data, high computational costs, and time. They are good at generalization and most of the state-of-the-art methods now are using neural networks.

Using machine learning (ML), a subset of artificial intelligence (AI), software programs can more accurately predict outcomes without being explicitly programmed to do so. In order to forecast new output values, machine learning algorithms using previous data as input. There are concepts like Decision Tree and Random Forest that allow to map several values from one domain to another by classifying them with simple rules, based on few annotated examples. Those can be used for image processing for automated pixel-wise classification, which is also known as segmentation task.

These procedures are lacking in a few areas. The first is that the user must design and select the feature extraction stage based on their own experience. Secondly, inefficient computation. Using traditional machine learning algorithms for three-dimensional stacks becomes very slow. Thirdly, these methods cannot be applied effectively to unknown data. Neural networks are developed to provide solutions for all three of these issues.

2.2 Neural Networks

Artificial neural networks are a subfield of machine learning that is partly based on how neurons in the human brain function. In the last decade, neural networks have had a tremendous rebirth because to the immense availability of data and the great growth in computing power, mainly due to use of Graphical Processing Units (GPU) availability. There are a variety of neural network types (Feedforward, Convolutional, Recurrent, etc.). A neural network is not an algorithm, rather it is a computational structure that approximates an algorithm by being built on learnt mapping and being calculated via the process of training. Algorithms use is commonly referred to the simplest formula:

$$\text{Algorithm} = \text{Personal experience; Question} + \text{Algorithm} = \text{Answer}$$

An algorithm is a sequence of computations, that is defined by human, and usually algorithm is not a self-improving system. Neural network on another hand is being trained by following this formula:

Algorithm = Question + Answer; Question + Algorithm = Answer

By displaying the network's input and output, it is feasible to educate it to produce a new output given a new input.

2.2.1 Feed forward network

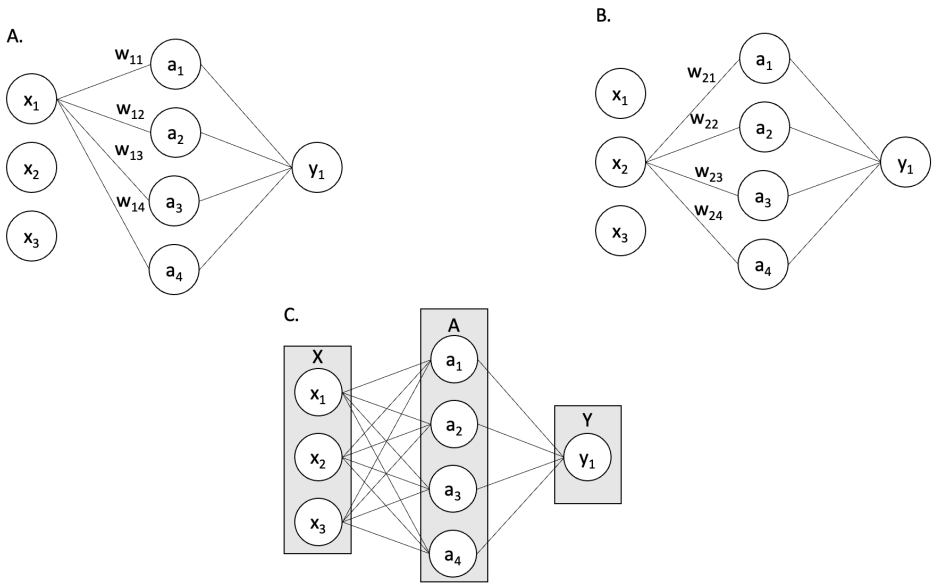

Figure 4: Schematic representation of feed forward network; A. Forward pass from neuron x_1; B. Forward pass from neuron x_2; C. Forward pass from all input neurons.

Feed forward network is one of the earliest forms of neural networks. It allows to approximate mapping of complex domains and solve problems like, classification, regression. On Figure 4 as an example illustrated a feed forward network with one input layer X one hidden layer A and output layer, that consists of one neuron Y. Every node or "neuron" is considered as a variable for x_i or as linear combination a_k of weights w_{ij} and inputs x_i. Considering the first case on Figure 4A the so called forward pass is performed from input layer to hidden one. It is represented by lines. Each input neuron has "connection" with all neurons in the next layer. Each connection has a weight assigned to it. It can be written as:

$$a_k = \sum_{j=1}^{A} w_{ij} \cdot x_i$$

This operation is repeated for all input neurons, Figure 4B:

$$a_k = \sum_{i=1}^{X} \sum_{j=1}^{A} w_{ij} \cdot x_i$$

All neurons in hidden layer A represent nothing more than a linear combination. Linear combination cannot approximate mappings more complex than linear ones. To solve this issue every layer output has a non-linearity function σ, that allows to approximate more complex distributions.

There are multiple functions that are used: Rectified linear unit, Sigmoid, Softmax etc. Once the non-linearity is added to every layer's output, it can be forwarded to next layer until the end.

For computational efficiency and simplicity computation from one layer to another can be rewritten in matrix form:

$$\hat{y} = \sigma(W_a a)$$

$$\begin{pmatrix} \widehat{y_1} \\ \widehat{y_2} \end{pmatrix} = \sigma \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \end{bmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix}$$

And because a values were computed the same way as $\hat{y}$ via matrix multiplication:

$$a = \sigma W_x x$$

But how exactly do neural networks compare their output to the training samples? It is done by loss functions. They compare the network's output with ground truth and measure how different they are. Once a loss is calculated, it is backpropagated. Backpropagation is the process of sequential gradient computation from the loss value back to the input layer. A gradient of loss with respect to input and with respect to weights is computed and used for gradient descent weight optimization. This method force weights change to minimize loss or difference between output and target values.

2.2.2 Convolutional neural networks

Feed forward networks operate well but have limitations that prevent them from working with large data such as photos, video, and audio. There are two primary reasons for considering an alternative to feed-forward networks.

Firstly, feed-forward networks are not scalable. Mainly, because they are computed by matrix multiplication which has a complexity of $O(n^3)$. Imagine the

example with a 512-by-512-pixel picture. For a single hidden layer with 8 neurons, it results in $(512 * 512 + 1) * 8 > 2\,000\,000$ trainable weights. Memory consumption for a single operation is very high, it is not feasible to train a network with so many weights on average personal computer.

Second, pixels are a poor representation from the standpoint of machine learning. They are highly correlated, scale dependent and vary in intensity. Hence pixels on their own do does not carry any spatial information about object, it is just a floating-point value.

For images, feed forward networks become computationally difficult and worthless. Convolutional neural networks (CNN) were constructed as a result. They address both issues simultaneously. The size of a convolutional kernel is typically between 1 and 81 pixels, and kernel values are trainable weights. This provides for significant memory savings, while the convolutional kernel enables the manipulation of spatial features through image convolution. Instead of forwarding the whole image, CNN convolves it using kernels and only transmits the output of the convolution **Figure 5**.

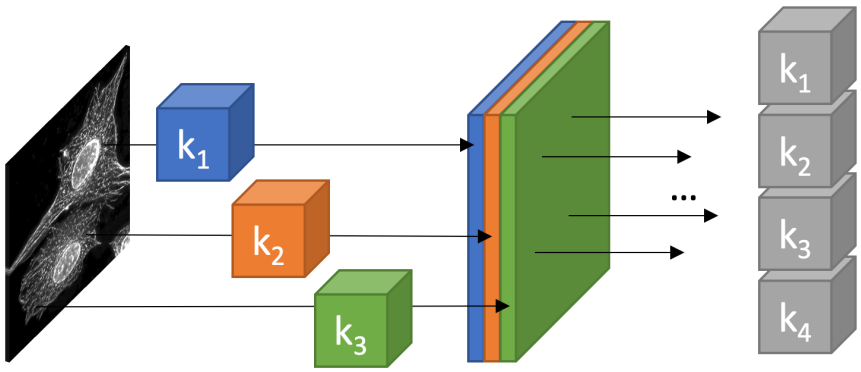

Figure 5: Representation of CNN work. Image is convolved with three kernels k_1, k_2, k_3, to produce a tensor with three channels. Convolution is repeated with a new set of kernels, using the output feature maps from the previous stage as inputs.

CNNs are very powerful networks, and they can solve problems with different approaches. To satisfy the requirements of this project it is necessary to detect, classify and count cells from three dimensional images. Two common approaches are used to accomplish that task: object detection and image segmentation.

2.3 Object detection vs. Image Segmentation

Image classification is one of the most prevalent applications of CNNs; it takes an image as input and predicts the probability of object existence. State of the art networks like in (3) can define presence of object on the image. For an image containing 100 cells it would predict that “cell” is there, but it does not provide information about cells number. For predicting cell number, first those cells must be localized on the image and then, it is possible by postprocessing means to determine their number and class. One of the approaches is called object detection.

Figure 6: Example of algorithm output difference. From left to right: Classification – allows to predict content on image; Detection – allows to localize objects on image with bounding boxes and to predict their class; Segmentation – allows to assign every pixel on image to class.

Object detection enables the prediction of an object's position using a bounding box — a rectangular frame that encompasses the object's perimeter. There is multiple of models, that exploit different architectures to solve this problem. One of the most popular networks YOLOv3 (4) sort of divided image in grid and predicted bounding boxes coordinates and class probability for every image chunk. And then by using non-maximal suppression technique leave only boxes high class probability. There are other good alternatives like SSD – Single Shot MultiBox Detector (5), Faster R-CNN (6), or RetinaNet (7). These networks might be utilized for cell detection, however for three-dimensional data they have not yet been implemented.

In contrast, there are algorithms designed for segmentation purposes. They enable class assignment for each pixel in an image. When morphological examination of cell shape is necessary, this capability is used. Commonly used networks are U-net (8) an encoder-decoder shaped network with residual connections, usually it has output of the same shape as input, that allows to map pixels directly without rescaling back on input. Mask R-CNN (9) is another network design, that augments Faster R-CNN architecture with a segmentation branch, it basically takes each predicted boundary box and performs segmentation inside it.

Both network design branches are efficient and functioning well, but research articles consistently overlook a challenge that is seldom considered: labeling. For boundary box labeling, it is required to label two points on image and for

segmentation it is one contour. When pictures are three-dimensional instead of two-dimensional, however, boundary boxes need three points and segmentation requires contours multiplied by image depth in pixels. For cellular data both boundary box labeling and segmentation become tedious time consuming.

What if it is not necessary to label whole cells with contours or boundary boxes, but only their centers with a single point? Out there That would make labeling easier. At current state-of-the-art there are hundreds of image segmentations networks for planar images, few for volumetric cell segmentation and none for localization and classification with point notation – the easiest and fastest labeling strategy. In this results section the applicability of point notation on fluorescence stacks will be tested as well as cell simulator, that will support this testing.

3 Methods

In this section two parts will be explained: the cell simulator for training dataset creation and MBT-net for using simulated data for cell detection and classification.

Commonly deep learning project require three major parts: data, annotation, and network. We want to design a network that could localize and classify cells. This needs data and annotation, but typically, to gather and label data, one must be aware of the required resolution, imaging depth, signal-to-noise ratio, etc. If data was collected with incorrect or inefficient parameters, it may be impossible to utilize it for network training, or the network may fail to detect cells. Therefore, it is usually preferable to begin with a good estimate of the acquisition parameters for images that a network may identify and then determine whether the design method is correct. To test network architecture, it is not necessary to start with real annotated data, labeling cells on real stack might be substituted by synthetic data.

A simulation software framework of autofluorescence in colon tissue was developed and utilized for training data generation in section 3.1. The generated training dataset is used to train network and to test different network architecture, loss functions and training strategies in section 3.2.

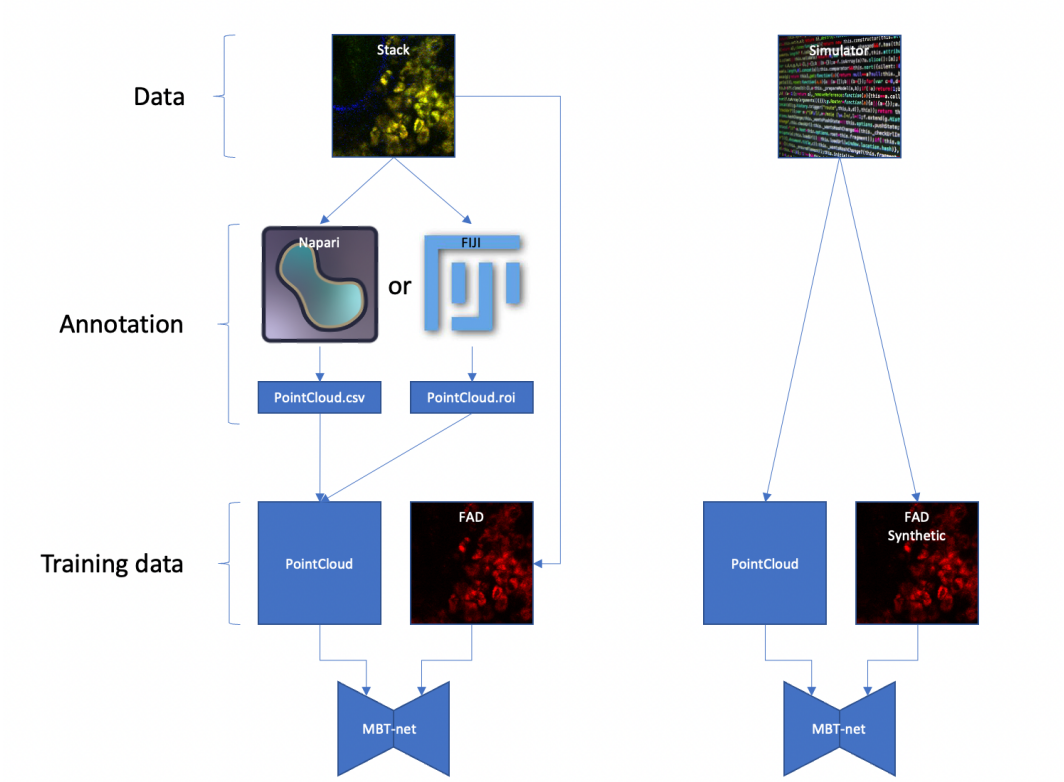

Figure 7: Chart visualizing two separate workflows for cell counting and classification. First column represents workflow with real annotated data: Stack is annotated by FIJI or Napari, and then used to train MBT-net. Second column shows a synthetic workflow, where labeling part is skipped, because simulated data is annotated by default.

3.1 Cell simulator

Deep Learning approaches require a lot of labeled data like hundreds or even thousands of images. Due to that, it is always preferable to find a way to minimize the labeling time because it is tedious and expensive. This issue intensifies when dealing with three-dimensional data. For this purpose, the Cell Simulator was created. It simulates image stacks similar to those seen in the real world like in **Figure 8**. It replicates the fluorescence of immune cell images in the FAD channel. It accomplishes that by creating three dimensional figures, to mimic shapes of cell nucleus and cytoplasm, and populate image volume with them.

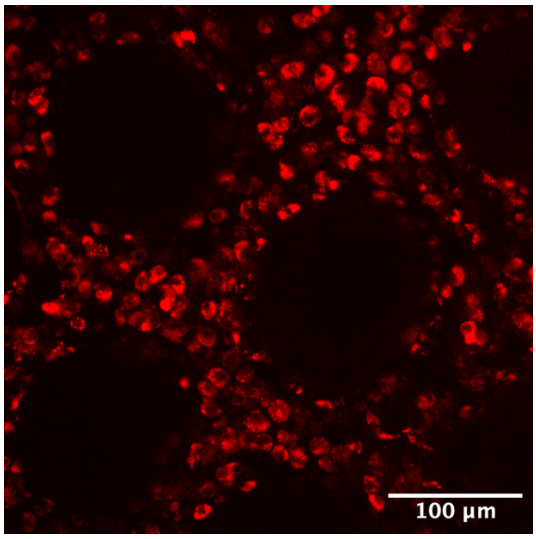

Figure 8: Example of stack layer recorded in human colon tissue. Acquired with multiphoton microscope, excitation wavelength 810nm, emission in FAD emission spectrum. MBT FAU Erlangen.

3.1.1 General approach

FAD fluorescence from immune cell looks as following: a bright spheroid with dark spheroid inside of it. Both shapes are not perfect spheroids but have continuous deviation from perfect sphere. Images have noise, and intensity variations **Figure 9**.

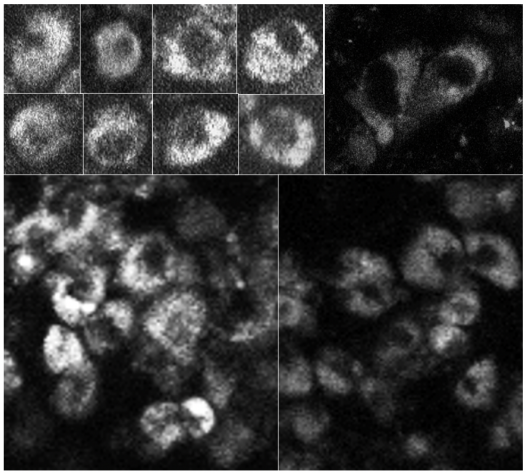

Figure 9: Examples of immune cells recorded at FAD fluorescence channel. Top left images of individual cells. Other pictures – examples of cell clusters. MBT FAU Erlangen

As a starting point for cell simulation the simplest spheroid shape could be modeled - a sphere:

$$S(x, y, z) = (x^2 + y^2 + z^2)$$
$$S(r, \theta, \varphi) = r$$

But in practice cells are often not perfectly regular and have imperfections, that change their shape. In spherical coordinates this deviation could be simply added to modulate spherical shape.

3.1.2 Cell shape modulation

Shape change could be simulated via adding random values for each shape radius r. But random noise has no "continuousness", every entry in random vector tends to have no connection to its neighbors. Cell made with this noise could appear unnatural – artificial. The same questions were addressed by Ken Perlin in his works (10), (11). He developed a well-known noise generation algorithm that creates a gradient noise. Gradient noise is a sort of noise often utilized in computer graphics as a procedural texture primitive. Conceptually distinct and sometimes confused with value noise. This approach involves the development of a lattice of random (or often pseudorandom) gradients, whose dot products are then used to interpolate values between the lattices. Unlike value noise, gradient noise has a greater amount of energy at high frequencies. Modern gradient noise implementations were later developed, that improved some artifacts of original method like Simplex (12) (patented till 08.01.2022) and its free alternative OpenSimplex (13).

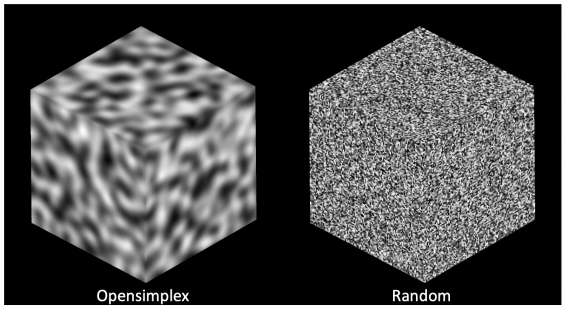

Figure 10: Example of three-dimensional noise domains. Left – OpenSimplex noise, right – Random gaussian noise.

OpenSimplex noise can be generated at different scales and can be exploited the same way as Fourier transform. By adding OpenSimplex noise of different frequency and amplitude different textures can be generated **Figure 11**A. To

define noise amplitude composition a weighting function with parameter γ was used **Figure 11**B. Complete formula for gradient noise space generation was developed:

$$N(\gamma, \alpha, x, y, z) = \sum_{\alpha=1}^{A} \mathbb{N}(\alpha, \mathrm{x}, \mathrm{y}, \mathrm{z})\, e^{-\gamma\alpha}$$

Where $\mathbb{N}$ is an OpenSimplex gradient noise function, α scale parameter, A maximum scale, γ amplitude attenuation parameter. This weighting could also be done with any other function of choice, but with gamma it is more representative to adjust frequency composition in cell shape formation.

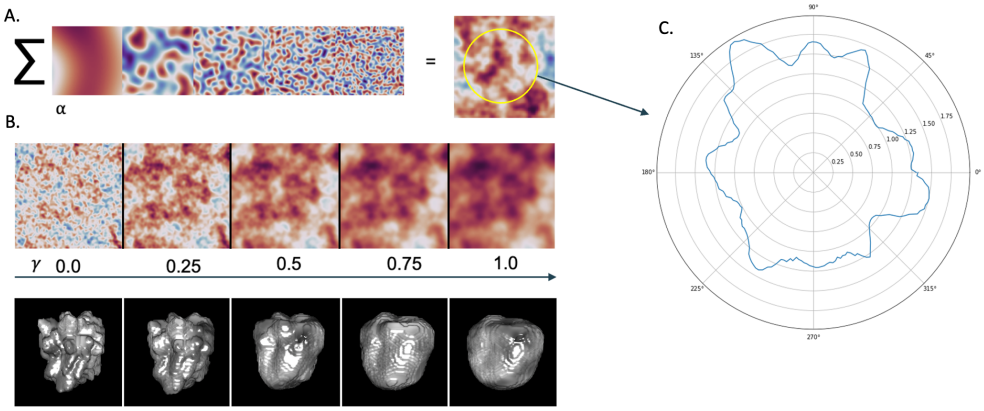

Figure 11: Texture generation: A. Explanation of how the superposition of many frequencies of noise may produce a texture. B. Representation of parameter influence gamma. C. Result of sampling circle in noise domain.

Gradient noise is smooth, but it is not cyclic, which is required for spherical shapes. How to preserve smooth connection for cell surface? To receive a cyclic distribution a closed loop must be sampled from given domain. This issue was solved by sampling noise values with the simplest three-dimensional loop – sphere. This sampled noise we call a noise patch N_p:

$$N_p = N(\gamma, \alpha, x, y, z) = N(\gamma, \alpha, r\sin\theta\cos\varphi\,, r\ \sin\theta\sin\varphi\,, r\cos\theta);\ r = const$$

OpenSimplex noise could be used to generate continuous deviation for cell spherical shape K, while being added to it in spherical coordinates:

$$\mathrm{K}(r, \theta, \varphi) = c * r + N_p$$

Variable c is multiplied with sphere radius to modify relative influence on final shape. Larger values of c let spherical component to be more prominent than noise patch N_p. By varying values c and γ different forms could be achieved **Figure 12**.

FAD fluorescence signal is not ideal in reality and cannot have the same intensity over the image – it has a smooth intensity distribution over the cell which is also can be simulated by proposed gradient noise space. It could be applied to generated cell shape as texture.

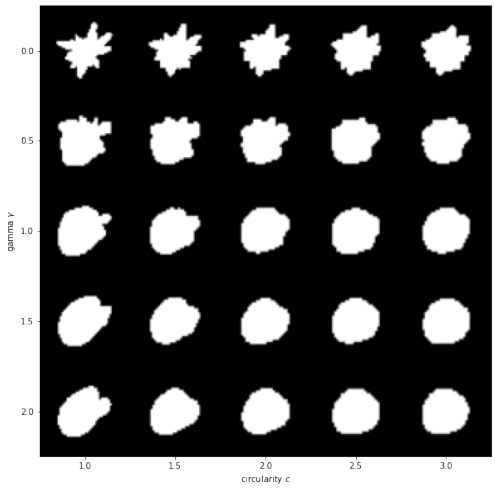

Figure 12: Visualization of weighting constants influence on resulting cell shape. Vertical axis γ, horizontal axis c. Shape has higher frequency features with smaller γ and has less noise amplitude with increasing c.

Overall cell simulation algorithm is shown on **Figure 13**. At first, noise space is produced $N(\gamma, \alpha, x, y, z)$ and sampled to make a noise patch N_p, which is then added to a sphere in spherical coordinates to form a cell shape mask $\mathrm{K}(r, \theta, \varphi)$. A noise domain is then applied to this mask in cartesian space to give a realistic intensity distribution. Two blobs with distinct parameters are generated: the cell body and the nucleus mask. The nucleus mask is then subtracted from the cell body to simulate the absence of FAD fluorescence in the nucleus. This produces a simulated cell.

This shape generation allows to modulate cell appearance with few parameters and generate an infinite number of cells with similar properties, that could be exploited for simulation of cell classes. In this work two cell classes were generated: Macrophage and T-Cell. Their size parameters are like real cells (14). Parameters can be found in Table 3.1.

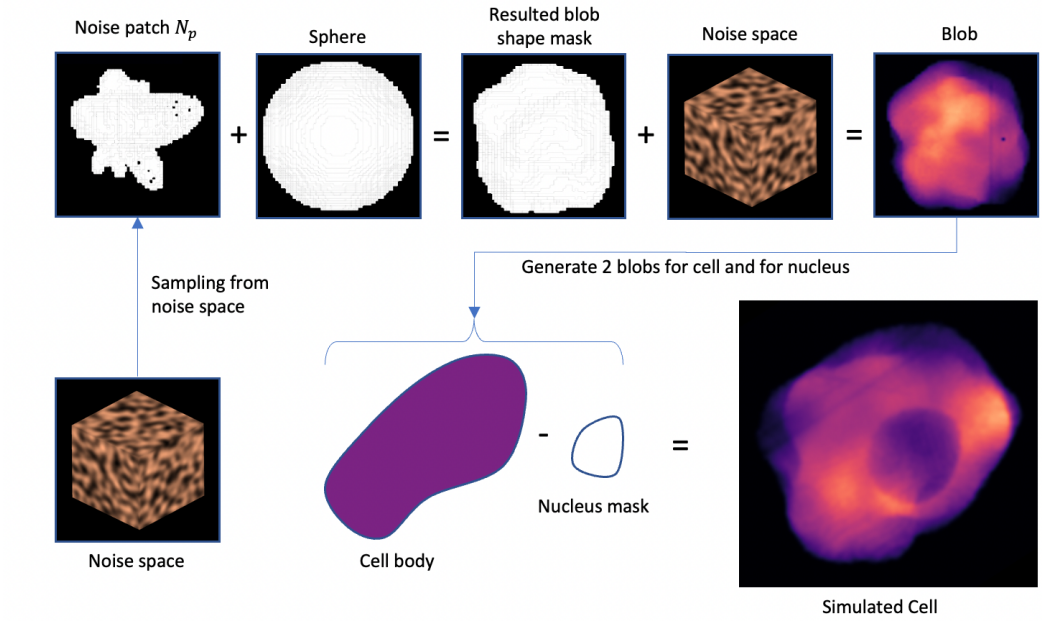

Figure 13: Algorithm of cell simulation: noise space is generated and sampled to create a noise patch, then it is summed with a sphere in spherical coordinates to create a cell shape mask, then a noise domain is applied on this mask to create a natural intensity distribution. Two blobs with different settings are created - cell body and nucleus mask. Then nucleus mask is subtracted from cell body to imitate no FAD fluorescence in nucleus region. That results in a simulated cell.

3.1.3 Stack generation

Simulation of a single cell is a starting point of stack creation. A canvas full of cells and obstructions could mimic realistic images of colon tissue. In this simulation framework cell placement is done as in this algorithm:

1. Generate a cell.
2. Cell mask is placed in random place on the stack
3. If cell mask overlaps with stack mask, then go to 2.
4. Place a cell image in the stack and add cell mask to stack mask.
5. Repeat with a new cell until number of cells or there will be no place to put a cell or number of tries exceeds the limit.

To make cell distribution more realistic, crypts were added to simulation. Those structures usually do not contribute to FAD fluorescence channel, and only displace cells from image. Simulation of this obstruction is simple and may be reached in the same manner as simulated cells – by creating a blob, but in two-dimensional case, because usually crypts are distributed along depth axis **Figure 14**. Afterwards this blob is used as mask to restrict placement of cells. By following this algorithm, immune cell distribution on simulated stacks was achieved.

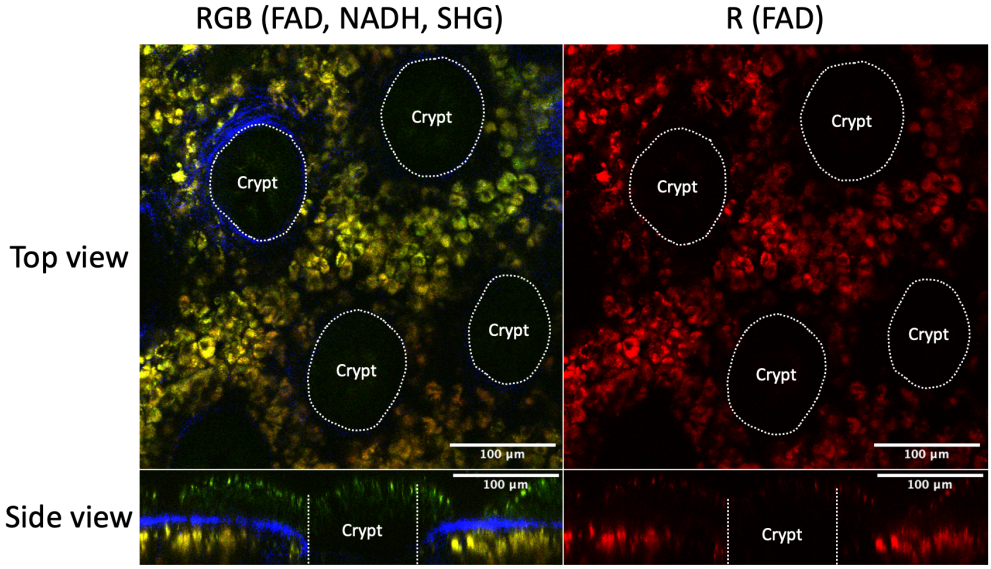

Figure 14: Real stack acquired at MBT FAU Erlangen. Human colon tissue sample, red – FAD, green – NADG, blue – SHG. Dashed lines represent crypts in colon tissue. Blue channel shows collagen matrix. Yellow (combination of FAD and NADH) shows immune cells.

3.2 MBT-net

Another important part of this project is a creation of neural network that could localize and classify cells in a three-dimensional stack. In this work we introduce and MBT-net **Error! Reference source not found.**, a network for segmentation and classification purposes that exploits efficient strategies proposed by community and put them all in a single network. MBT-net is built around a popular network architecture called LinkNet (15). That is an encoder decoder network that outputs a tensor of the same resolution as an input tensor but multiplied by number of classes. It uses summation as a skipped connection to save memory and speed-up training process.

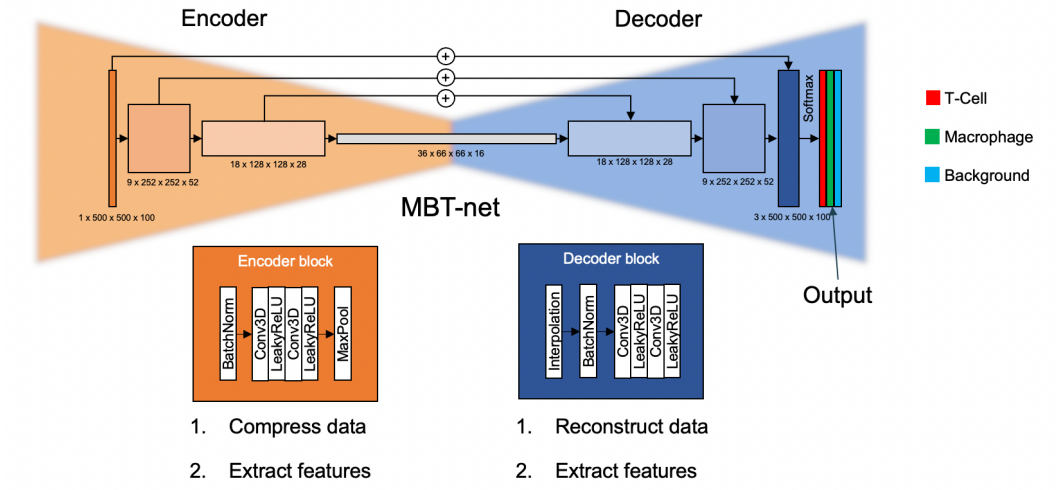

Figure 15: MBT-net architecture. Orange – encoder blocks, Blue – decoder blocks. For simplicity reasons visualization is done in two dimensions. Every block is at scale to tensor shape.

Every encoder block is structured as following: Batch normalization layer, 2 x (Convolution3D, LeakyReLU), MaxPool3D. Batch normalization is required at every encoder block input to reduce effect of numerical instability (16). Then comes two times convolution layers with three-dimensional kernels 3 x 3 x 3 and padding 2 stride 1. After each convolution a non-linearity function Leaky Rectified Linear Unit was added. It was used instead of well-known ReLU because of better accuracy and no issues like dying neurons (17).

Every decoder block is structured as following: trilinear interpolation layer for input up sampling, Batch normalization layer, 2 x (Convolution3D, LeakyReLU). For up sampling purposes the interpolation approach from (18) was used.

Every encoder has a single skipped connection to decoder as on the image. As explained previously skipped connection are done via summation. As a final activation function SoftMax was used:

$$Softmax(x) = \hat{y}_k = \frac{e^{x_k}}{\sum_{j=1}^{K} e^{x_j}};$$

$$Properties: \quad 1. \sum_{k=1}^{K} \hat{y}_k = 1; \quad 2. \ \hat{y}_k \geq 0$$

It allows to output a probability vector for multi-class problem. Now for each pixel in input tensor there are 3 pixels in output tensor that predict class probability.

Training of neural network needs a suitable loss function. For training a weighted Cross Entropy Loss was used:

$$Weighted\ Cross\ Entropy\ loss: L(y, \hat{y}) = -\frac{1}{f_c}\log(\hat{y}_k)\,|_{y_k=1}$$

It calculates a negative logarithm only for output $\hat{y}_k$ which must be equal to 1. f_c is a relative class frequency. It is calculated based on the training dataset, where for each class, number of entries is counted and then divided by the number of all entries. More frequent class must experience less loss when misclassified.

Cell labeling was performed automatically by simulation software as a point-notation. Each cell center was marked as single pixel with value "1". But network cannot output perfect result, but instead a probability density function of each cell location and class **Figure 18**. To address this issue, a counting algorithm was used.

Following that network output is fed through Softmax function output is following a property (1). Following probabilistic output (2), we can predict Cell probability P_{Cell}. Then we could find for each region P_{loc} with probability above threshold a local maximum (3). And, sequentially use these coordinates to classify the cell (4). So, at location of the highest probability P_{loc} we compare classes probabilities to find the highest one.

1. $\hat{y}(x, y, z) = P_M(x, y, z) + P_T(x, y, z) + P_B(x, y, z) = 1\ \forall\ x, y, z$
2. $P_M + P_T = P_{Cell}; \quad P_{Cell} + P_B = 1; \quad P_{Cell} = 1 - P_B$
3. $\dot{x}, \dot{y}, \dot{z} = \mathrm{argmax}(P_{loc})$
4. $\max_{x,y,z}(P_M(\dot{x}, \dot{y}, \dot{z}), P_T(\dot{x}, \dot{y}, \dot{z}))$

3.3 Software packages

For development of mentioned software packages, a list of following tools was used. The whole project was written entirely in Python 3 (19). At the time of writing this Thesis it is the most popular programming language according to TIOBE rating (20). As well as for other languages there exist libraries or modules, of open-source software and frameworks that could be used to build sophisticated programs. Main modules, that were used for development of Cell Simulator are:

- NumPy (21) – scientific package for calculations and operations with matrices
- Pandas (22) – package similar to NumPy that works with table data
- Napari (23) – visualization software for volumetric and planar images. It is an alternative to ImageJ.
- Scikit-image (24)– a collection of computer vision functions and algorithms
- OpenSimplex (25) – module, that generates gradient noise like commercial simplex noise.
- Pytorch (26) - an open-source deep learning framework that enables GPU utilization with simple interface.
- Pytorch-lightning (27) – research framework. Allows convenient manipulation of data during network training.
- Git (28) – version control software. Allows to track changes of software over time
- GitHub (29)– a developer platform that was used to host repository with developed software

4 Results

As the result of this Master's Thesis, two software packages were developed:

1. Cell Simulator framework. It allows to simulate infinite number of cells that share same appearance parameters in 3D and mimic their look as if they were acquired with MPM at FAD channel. Also, it allows to generate stacks filled with simulated cells and crypts.
2. MBT-net neural network for classification and localization of immune cells infiltrates in colon tissue. It was designed, trained, and tested using simulated data.

And as the last result neural network was trained on simulated stacks and its performance was evaluated on test stacks. Developed software is documented and is available as repository on MBT server as well as on GitHub here: *https://github.com/dobrik4ever/Master-Thesis.git*

4.1 Cell simulator

Cell simulator is not static program but an extensible and modifiable framework. It was developed with aims of scalability and usage simplicity. It is based on two main classes – "CellSimulator3D" a class, that enables easy interface for cell configuration and tuning, as well as visualization. It builds a volume that consist of cell image, cell mask and cell center. And "Simulator3D" class is meant for creation of a stack that is populated by cells. It has input parameters like stack size, sampling size, types of cells to be used, their number and noise parameters. Both classes were used to training data.

The shape generation from CellSimulator3D allows to modulate cell appearance with few parameters and generate an infinite number of cells with similar properties that could be exploited for simulation of cell classes. In this work two cell classes were generated: Macrophage and T-Cell. Their size parameters are taken from literature [30]. Parameters are listed in **Table 4.1**. Examples of generated cells in training dataset can be seen on **Figure 16**. Comparison of simulated stacks and real stacks can be seen on **Figure 17**. Here crypts were simulated in the same fashion as cells and used as masks. The simulated stacks differ from real ones, because they don't consider fluorescent debris, point spread function effects, absorption.

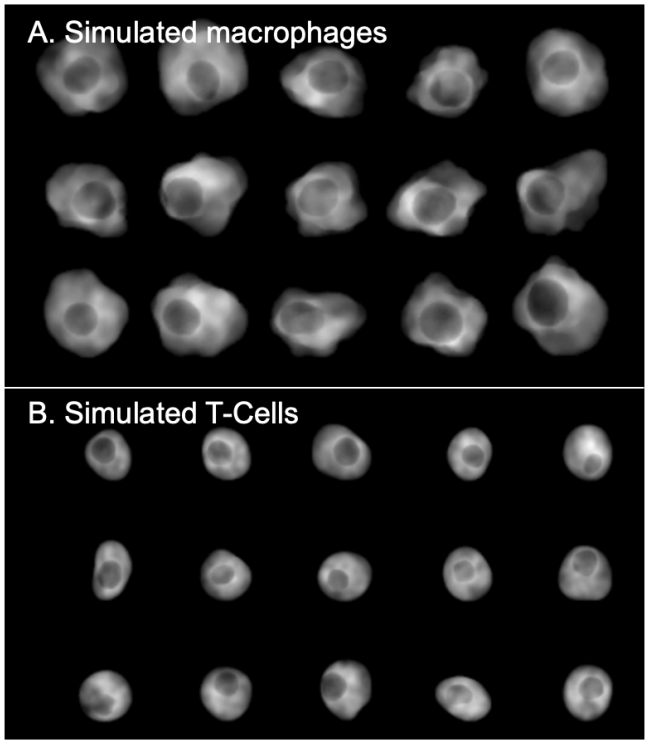

Figure 16: Examples of simulated cells with parameters given in the table 3.1. A. Simulated macrophages B. Simulated T-cells. The shape generation from CellSimulator3D allows to modulate cell appearance with few parameters and generate an infinite number of cells with similar properties that could be exploited for simulation of cell classes.

Table 4.1 A cell simulation parameters table.

	Cytoplasm size, µm	Nucleus size, µm	γ, nucleus shape	c, nucleus shape	γ, cytoplasm shape	c, cytoplasm shape
Macrophage	15	6	1.5	2.0	0.5	1.5
T-Cell	9	4	1.5	2.0	1.5	2.0

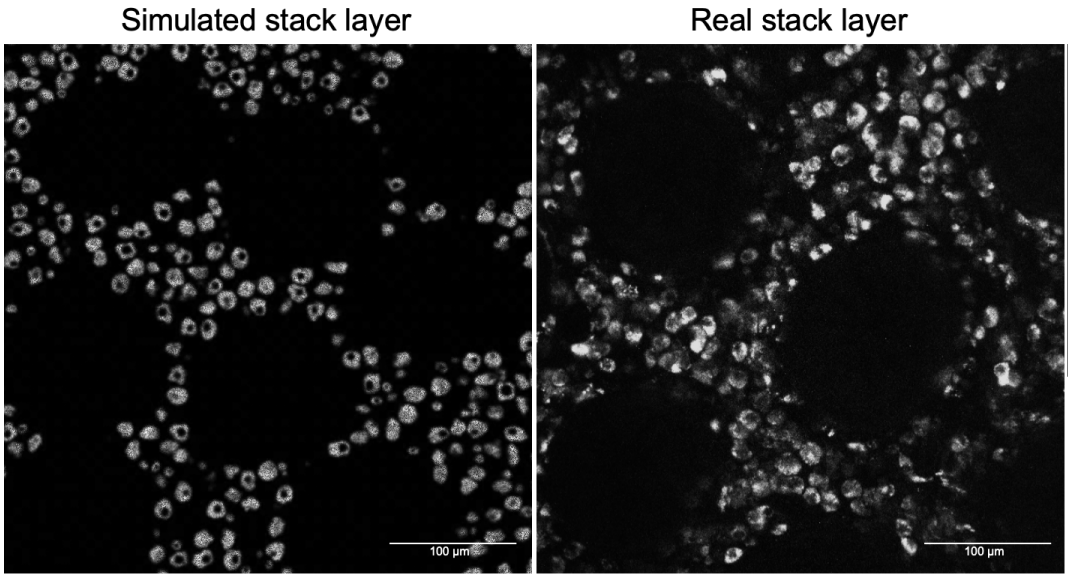

Figure 17: Comparison of simulated stack (left), and real stack (right). Simulated stack differs from real, because of fluorescent debris, scattering, PSF and signal attenuation at depth. The variance in cell size and shapes is currently also smaller in simulation, but can be further adjusted with simulation parameters.

4.2 MBT-net

The simulated data was used to train MBT-net. For training of MBT-net we used the following parameters Table 3.2 and training, validation losses can be seen on the **Figure 18**. Loss curves describe how difference between a ground truth and network output minimizes over time and saturates. The network output was rendered in 3D with green and red channels for network's output (probability density function for each cell class) and gray for the stack. As can be seen each cell is represented by a small, segregated region. Each region was used to identify cell location and class. This is what was planned to achieve in the beginning.

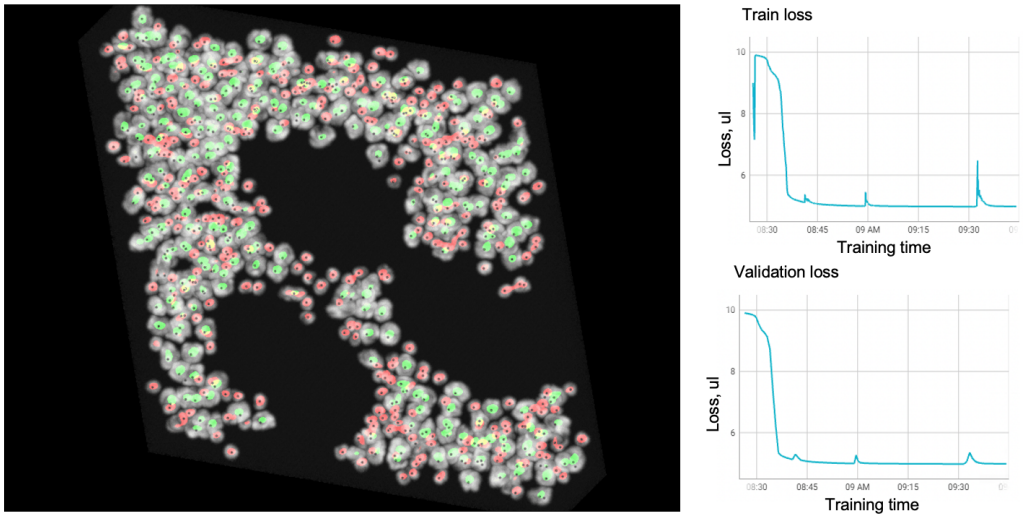

Figure 18: Overlay of network output and network input for given test stack. Green regions is a probability density function (PDF) of Macrophages and red ones of T-cells. Every cell is localized by a segregated region. Train and validation losses are plotted on the right.

Table 4.2 Training parameters for this project

Parameter	**Value**
Output function	Softmax: $\hat{y}_k = \frac{\exp(x_k)}{\sum_{j=1}^{K} \exp(x_j)}$
Loss function	Weighted cross entropy loss: $L(\hat{y}, y) = -\frac{1}{f_c} \sum_{k=1}^{K} \log(\hat{y}_k)\,|_{y_{k=1}}$
Batch size, learning rate	1, 10^{-4}
Optimization	Adam
Stack size	500 x 500 x 100 pixels 250 x 250 x 50 µm 0.5 µm / pixel
Training samples	15 train / 5 validation / 10 test
Epochs	100

Stack #	Ground truth		Prediction		Misclassified %	
	Macrophage	T-Cell	Macrophage	T-Cell	Macrophage	T-Cell
0	336	364	312	366	7%	1%
1	314	301	287	298	9%	1%
2	285	313	262	317	8%	1%
3	327	347	293	373	10%	7%
4	329	353	299	364	9%	3%
5	325	302	294	318	10%	5%
6	340	308	302	325	11%	6%
7	237	237	217	245	8%	3%
8	324	308	301	317	7%	3%
9	303	329	279	347	8%	5%
				Deviation	1%	2%
				Average	9%	4%

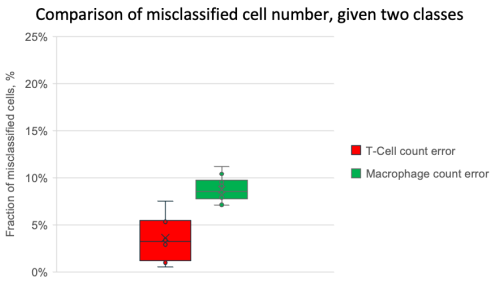

Figure 19: Evaluation statistics of MBT-net performance. Table on the left demonstrates percentage of misclassified cells. Box plot on the right demonstrates deviation and mean of misclassified cells number along 10 test stacks

For evaluation of network performance, we used 10 test stacks – stacks that have never been seen by network. Now we can feed these stacks through trained network and calculate number of identified cells. Then a comparison between ground truth number and predicted number will be checked.

Network can classify and count on average 90% of cells on previously unseen data correctly. This is a comparatively good result. But it is important to consider – test data was generated by the same algorithm, as well as the training data. Despite of being tested on unseen set, it still has bias of simulator. That means, that the same algorithm can perform worse on real data, because it has higher variance and complexity. The misclassification comes from network-related artifacts. When outputting the probability density functions cells can be misidentified due to two reasons: Double prediction for macrophages and dense cell clusters. Examples of misidentified cells could be seen below:

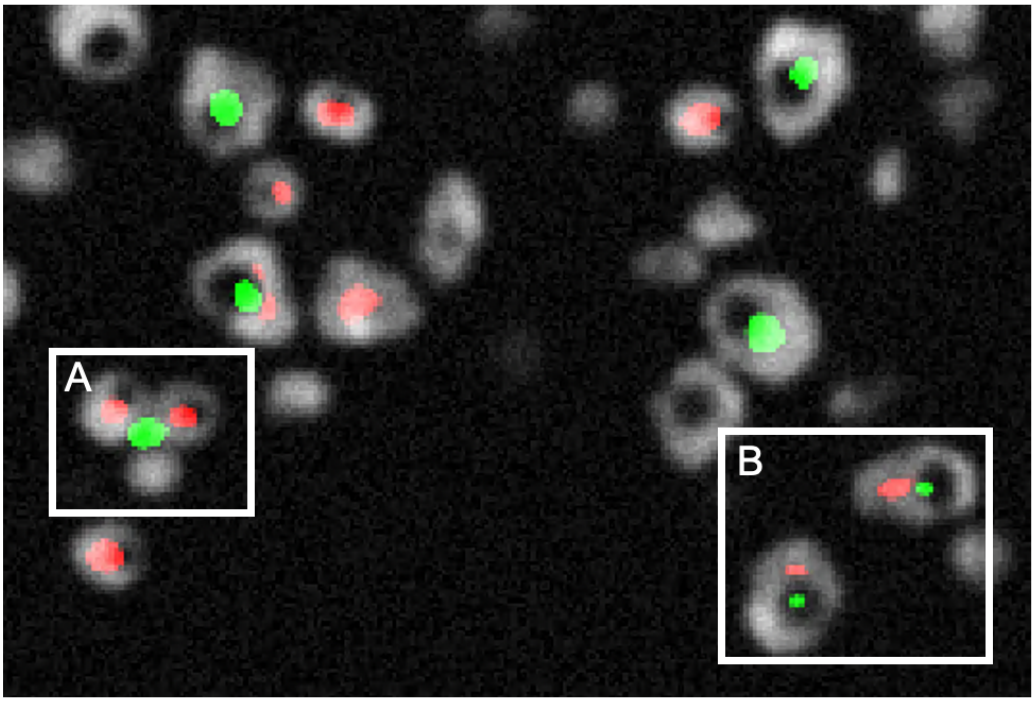

Figure 20: Overlay of generated stack layer and produced output from MBT-net. **A**. A cluster of T-cells reminded network of macrophage. **B**. Macrophages were identified correctly, but part of them experienced double prediction of T-cells.

5 Discussion

An automated quantification of cell populations in three-dimensional tissue is a very challenging task, which is not easily solved with classic image processing routines. Modern Deep Learning approaches are promising route to address the task, but require many well-annotated 3D datasets, which is a difficult and time-consuming manual procedure. In this thesis, the potential of neural networks for cell quantification in label-free stacks was evaluated. In this section an explanations of model selection, as well as applicability and faced problems will be discussed in detail.

5.1 Model selection

MBT-net is built around a popular network architecture called LinkNet (15) which was developed after famous and widely used U-net (30) introduced in 2015 as a solution for medical image segmentation purposes, and then have been improved to work with three-dimensional data (31).

These architectures main idea is built around two network modules – encoder and decoder. First one encodes input information in more compact form. Usually, any image classifier network could be called an encoder because it compresses input image in vector of class probabilities (for example 512 x 512 image in vector 1 x 3). Second module, decoder, is doing an opposite action, it decompresses information. Combination of two modules can result in efficient mapping from input image to output segmentation map with the same resolution as input, which is essential in segmentation tasks. But mentioned networks use a clever trick to allow for better backpropagation and information preservation called skipped connections. This is a connection (summation (LinkNet) or concatenation (U-net)) that connects output from one encoder block to input of decoder block at each scale **Figure 21**.

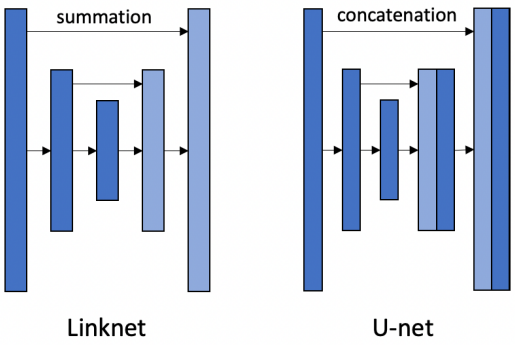

Figure 21: Comparison of LinkNet and U-net residual connections.

As described in (15) LinkNet has comparable segmentation quality to U-net, but with fewer memory consumption and higher computation speed. That's why in this project skipped connections like in LinkNet were utilized.

5.2 Real data

In this project an experiment was conducted. An attempt to feed a real stack to MBT-net. The outcome can be observed on **Figure 22**. As can be seen, it finds very few cells on this stack. MBT-net was trained on simulated data. And as any simulated data, it has bias, and it cannot adequately replicate the complexity of real data to develop networks capable of handling real data. But a fact that it marks few cells, is telling us that simulated cells from simulator were at least close to ones observed in reality. Few ways to improve could be considered. Firstly, a real data could be annotated and used for training. Secondly the cell simulator may be improved to make stacks look more like real data.

Another strategy would be to use trained on simulated data network, and then fine-tune it on real stacks. This strategy is not novel, and it is called transfer learning. Cell Simulator could help to pre-train MBT-net before it will be fine-tuned on real data. That can help to save time and reduce the power consumption, to help environment.

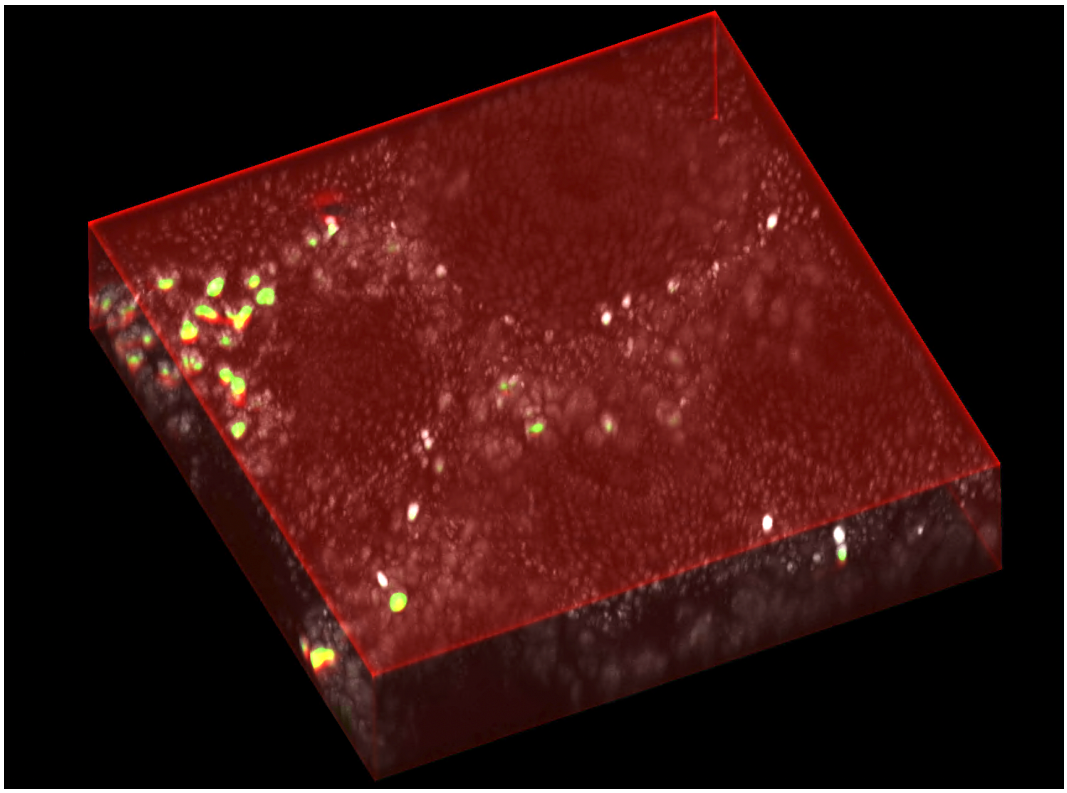

Figure 22: MBT-net output, given a real stack. Only few cells were found on this stack (green regions).

5.3 Occurred difficulties

5.3.1 Slow computation of spherical to cartesian transform

For training of the MBT-net twenty stacks were simulated. Simulation process was taking a lot of time even on powerful computers. That's why this software required acceleration. An investigation was conducted to find, which part of the algorithm was slowing down the simulation process. By using cProfile (32), a python module to measure time of execution for python program, we found the slowest function of this software. It was a transition from spherical to cartesian coordinates. To optimize execution time, we tried to use mathematical tricks. Initially the function #1 was calculating transform exactly like it is stated in literature (see in **Code block 1** below **spherical2cartesian1**). Because $\sin(P)$ is anyhow computed twice, we can store first computation made for x and apply it for computation of y (**spherical2cartesian2**).

```
def spherical2cartesian1(R, T, P):
    x = R * np.cos(T) * np.sin(P)
    y = R * np.sin(T) * np.sin(P)
    z = R * np.cos(P)
    return x, y, z

def spherical2cartesian2(R, T, P):
    sinP = np.sin(P)
    x = R * np.cos(T) * sinP
    y = R * np.sin(T) * sinP
    z = R * np.cos(P)
    return x, y, z

def spherical2cartesian3(R, T, P):
    sinP = np.sin(P)
    sinT = np.sin(T)
    cosT = np.sqrt(1 - np.square(sinT))
    cosP = np.sqrt(1 - np.square(sinP))
    indcs_T = (T > np.pi/2) & (T < 3*np.pi/2)
    indcs_P = (P > np.pi/2) & (P < 3*np.pi/2)
    cosT[indcs_T] *= -1
    cosP[indcs_P] *= -1
    x = R * cosT * sinP
    y = R * sinT * sinP
    z = R * cosP
    return x, y, z
```

Code block 1: Illustration of three functions for computation of spherical to cartesian transform

Computation of powers is less computationally intense than sine and cosine. Given that we used trigonometrical identity $\cos T = \sqrt{1 - sin^2(T)}$ but square of variable always output positive values, which does not work for cosine function. To fix that we multiplied by -1 regions of $\sin T$ and $\cos T$ that belongs to $(\frac{\pi}{2} < T < \frac{3\pi}{2})$ resulting function can be seen here (**spherical2cartesian3**). But not only that was done to improve computational efficiency, and hence computational speed. Different python compiler was used to accelerate computation of matrices. It is called Numba (33) and it allows to speed up computations to the level of C and Fortran languages in some cases. We used this compiler with mentioned functions, and the results can be seen here on **Figure 23**.

With approaches mentioned above we were able to accelerate simulation time in more than two times, eighteen seconds vs. 46 seconds.

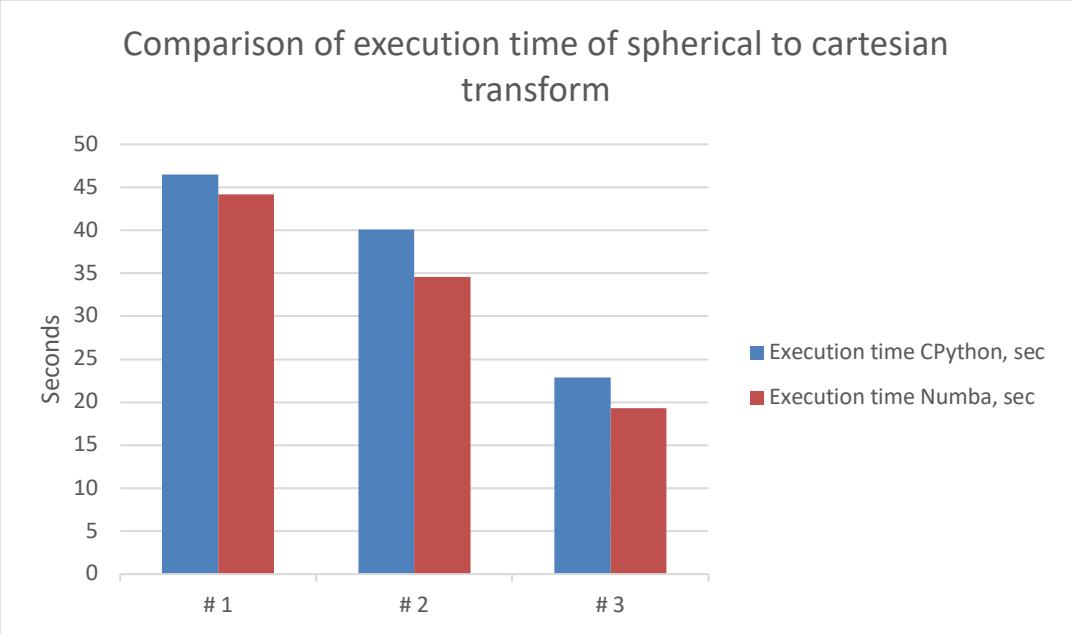

Figure 23: Comparison of execution time of spherical to cartesian transform. Blue – execution time using default CPython compiler. Red - execution time using Numba compiler.

5.3.2 Checkerboard artifact

Encoder decoder networks such as U-net and LinkNet networks, as well as Generative Adversarial Networks (GAN) usually are suffering from the so called "checker-board" artifacts. This image artifact could lead to disadvantageous results during and after training. As nicely explained in blog (34) layers like transposed convolution and max-pooling, max-unpooling cause these artifacts at network's output. To avoid a bilinear interpolation was suggested as a solution to that. But it was applied to two-dimensional images. But in this work this approach was adopted to three-dimensional images by means of trilinear interpolation. According to classification results and observance of network output it was proven to work well.

5.3.3 GPU memory limitation

As in the frame of this project we are dealing with volumetric images, the problem of hardware limitations is getting more noticeable. Neural networks are usually used and trained not on conventional CPUs but on Graphical Processing Units also known as GPUs. These devices were built to perform processing of graphical content, that most of times require matrix multiplication. In this regard, neural networks are no different from graphic content, because

they also require matrix multiplication. GPUs can perform matrix multiplication significantly faster than CPUs. The reason for that is their ability to build a computational graph that allows to perform matrix multiplication in parallel, rather than on CPU, that need to perform it sequentially. This parallelization speeds up the processing time but puts a limitation on ability of neural network to function. GPU memory limits the size of network; hence network must be optimized to be hosted by GPU.

In case of this project a training was performed with batch size equals to one. That's not a commonly used batch size, but it was necessary. When MBT-net forwards a single stack through its layers 18 gigabytes of GPU memory are being allocated. With GPU RTX 3080ti with 24 Gb of memory that was a limit. To train MBT-net on lower-end GPUs or to forward larger stacks it is needed to exploit an overlap-tile approach (35). It could solve issue with GPU memory consumption by dividing an input stack in a grid of equal sizes. Each element of this grid could be fed through network separately and then assembled back in the end. This approach isn't straight forward and requires to almost double number of operations required. The reason is that most of networks tend to lose information on image boundaries. Simply speaking it is challenging to classify object, half of which is lost on image boundary.

5.4 Failed approaches

5.4.1 Weka Segmentation plugin

One of the attempts to count immune cells in three dimensional stacks was by means of classical machine learning. One of the promising approaches was a random forest. That's a classification algorithm that could perform pixel-wise classification. To make it aware of any pixel surroundings handcrafted features were designed. For classification task several convolutions were used to acquire feature maps like gaussian, difference of gaussians, Laplacian, mean, max, variance etc. Despite of simplicity and effectiveness of how random forests work, no combination of parameters was successful enough to perform cell-wise segmentation.

5.4.2 CellPose

One of the promising approaches from deep learning field was CellPose. This project is very interesting because it does not deliver a neural network but graphical user interface, pretrained weights and superior generalization in

segmentation field of fluorescence data at that moment. One of its main benefits was no need of training network. It was already working well – we tried to feed through it layers of our data. But unfortunately, it was only developed for 2d images. In the scope of this project, we tried to feed our data through CellPose network layer by layer, and then tried to merge resulting mask in depth direction. Merging of masks in 3d didn't give us a good result either. So, we moved on with development of our own network and simulation framework.

5.5 Outlook

This project was done in a limited timeframe, and as any project can be improved in future. In this section will be explained a set of considerations and ideas, that can be improved to present a better performance.

5.5.1 Modification of ground truth representation

In the scope of current project ground truth stacks used for annotation the point notation. It is the simplest annotation form in segmentation stacks, but also an unstable one. In paper (36) a method for better data preparation was developed. At every cell location they apply a gaussian kernel that smoothens the transition from cell centroid to background. Potentially this approach will help to get rid of artifacts on MBT-net output.

5.5.2 MBT-net architecture improvement

Not only that can improve MBT-net prediction. Also, number of kernels used for it is not a lot. For this network the highest number of kernels per layer is only 36. It is preferable to minimize networks size to fit in in GPU memory but given hardware at MBT lab number of kernels can be increased. But not only that, but also depth of network. In MBT-net there are only three encoder and three decoder layers, but original U-net and LinkNet both have four encoder and four decoder layers.

5.5.3 Cell Simulator proposed features

Simulated data, generated by the Cell simulator, is distant from real images because this method does not include physical effects that define how image is formed. Adding those effects to Cell Simulator model could improve simulated data and make it more realistic. List of proposed features is:

1. Convolution with point spread function. It is clearly observable in real data.

2. Improvement of noise. At current state gaussian noise is used. But histogram of real images resembles Poisson distribution.

3. Adding debris. In original stack there always fluorescence debris around cells.

4. Attenuation simulation. For pixels acquired at greater depth there is less light intensity due to Beer-Lambert law of intensity attenuation.

6 Conclusion

Statistical analysis and quantification of cells is an important part of current research. An automated classification and localization of immune cells in colon tissue, imaged by label-free multiphoton microscopy, would be a very valuable tool. Within this thesis, a method for reliable cell quantification was developed: MBT-net, a neural network for cell localization and classification in 3D image stacks. To evaluate MBT-net, a simulator framework, called Cell Simulator, was developed that could mimic real 3D imaging data and provide an unlimited number of training samples. As a result of research on modern solutions in the fields of microscopy, computer vision, and machine learning, two software packages were developed:

1. Simulator for creating three dimensional datasets is developed. It can be used for generation of arbitrary volumetric data and not only for immune cells in FAD channel.
2. Neural Network for quantification and classification of immune cells in volumetric images is developed

Lastly, after training on synthetic data, 90% counting/classification accuracy of immune cells was achieved in simulated image stacks.

In future work, stacks with specific cell type markers will be acquired and used for training MBT-net. MBT-net will be improved to count and classify cells even better, and after training on real data, it will be used for research purposes.

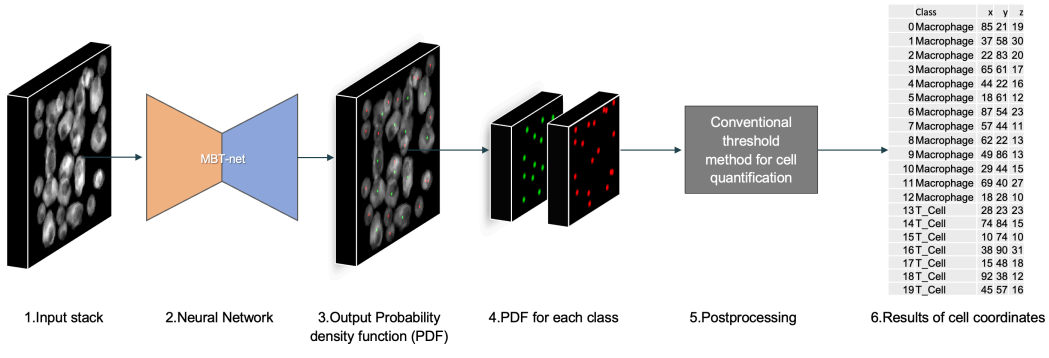

Figure 24: visualization of Cell Simulator and MBT-net workflow. Simulated stack is fed in MBT-net, it outputs PDF that is processed and used to count cells and expose their location.

7 References

1. Inhibiting Interleukin 36 Receptor Signaling Reduces Fibrosis in Mice With Chronic Intestinal Inflammation. Kristina Scheibe, Christina Kersten, Anabel Schmied, Michael Vieth, Tatiana Primbs, Birgitta Carlé, Ferdinand Knieling, Jing Claussen, Alexander C. Klimowicz, Jie Zheng, Patrick Baum, Sebastian Meyer, Sebastian Schürmann, Oliver Friedrich, Maximilian J. W. 4, 2019, Gastroenterology, Vol. 156, pp. A1-A22, e1-e18, 827-1224.

2. A gentle introduction to deep learning in medical image processing. Andreas Maier, Cgristopher Syben, Tobias Lasse, Christian Riess. 2, 2019, Zeitschrift für Medizinische Physik, Vol. 29, pp. 86-101.

3. Deep Residual Learning for Image Recognition. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. 2016. IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

4. Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. Washington : University of Washington, 2018.

5. SSD: Single Shot MultiBox Detector. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg. Amsterdam : Springer, Cham, 2016. Computer Vision – ECCV 2016.

6. Advances in Neural Information Processing Systems 28 (NIPS 2015). Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Montreal : s.n., 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.

7. Focal Loss for Dense Object Detection. Dollar, Tsung-Yi Lin Priya Goyal Ross Girshick Kaiming He Piotr. 02, 2020, IEEE Transactions on Pattern Analysis & Machine Intelligence, Vol. 42, pp. 318-327.

8. U-Net – Deep Learning for Cell Counting, Detection, and Morphometry. Thorsten Falk, Dominic Mai, Robert Bensch. 2019, Nature methods, Vol. 16, pp. 67–70.

9. Girshick, Kaiming He Georgia Gkioxari Piotr Dollar Ross. Mask R-CNN. [Online] January 24, 2018. [Cited: August 26, 2022.] https://arxiv.org/abs/1703.06870.

10. An Image Synthesizer. Perlin, Ken. San Francisco : s.n., 1985. SIGGRAPH.

11. Hypertexture. Perlin, Ken. 3, 1989, Computer Graphics, Vol. 23, pp. 253-262.

12. Perlin, Kenneth. Standard for perlin noise. US6867776B2 USA, March 15, 2005.

13. Uniblock. Noise! [Online] September 19, 2014. [Cited: August 29, 2022.] https://uniblock.tumblr.com/post/97868843242/noise.

14. 1. [Online] Faculty of Biological Sciences, University of Leeds. [Cited: August 31, 2022.] https://www.histology.leeds.ac.uk/blood/blood_wbc.php.

15. Abhishek Chaurasia, Eugenio Culurciello. LinkNet: Exploiting Encoder Representations for Efficient Semantic Segmentation. 2017 IEEE Visual Communications and Image Processing (VCIP). s.l. : IEEE, 2017, pp. 1-4.

16. Understanding Batch Normalization. Johan Bjorc, Carla Gomes, Bart Selman, Kilian Q. Weinberger. Montreal : s.n., 2018. Advances in Neural Information Processing Systems 31 (NeurIPS 2018).

17. Jain, Arun Kumar Vanita. Comparative Study of Convolutional Neural Netowrks Relu and Leaky relu activation functions. Lecture Notes in Electrical Engineering. Singapore : Springer, 2018, Vol. 553, pp. 873-880.

18. Image Super-Resolution Using Deep Convolutional Networks. Chao Dong, Chen Change Loy, Kaiming He, Xiaoou Tang. 2, 2016, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 38, pp. 295-307.

19. Welcome to Python.org. [Online] https://www.python.org/.

20. TIOBE index. [Online] https://www.tiobe.com/tiobe-index/.

21. NumPy. [Online] https://numpy.org/.

22. Pandas - Python Data Analysis. [Online] https://pandas.pydata.org/.

23. napari. [Online] https://napari.org/.

24. scikit-image: Image processing in Python. [Online] https://scikit-image.org/.

25. lmas, Alex. OpenSimplex Noise. [Online] https://github.com/lmas/opensimplex.

26. PyTorch. [Online] https://pytorch.org/.

27. PyTorch-lightning. [Online] https://www.pytorchlightning.ai/.

28. Git. [Online] https://git-scm.com/.

29. GitHub. [Online] https://github.com/.

30. O. Ronneberger, P.Fischer, T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. Medical Image Computing and Computer-Assisted Intervention (MICCAI). s.l. : Springer, 2015, pp. 234-241.

31. Özgün Cicek, Ahmed Abdulkadir, Soeren S. Lienkamp, homas Brox, Olaf Ronneberger. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. Medical Image Computing and Computer-Assisted Intervention (MICCAI). s.l. : Springer, 2016, pp. 424-432.

32. The Python Profilers. [Online] Python Software Foundation. https://docs.python.org/3/library/profile.html#module-cProfile.

33. Numba: a high performance Python Compiler. [Online] Anaconda Inc. https://numba.pydata.org/.

34. ODENA, AUGUSTUS. Deconvolution and Checkerboard Artifacts. [Online] Google research. Brain team, October 17, 2016. [Cited: August 30, 2020.] https://distill.pub/2016/deconv-checkerboard/.

35. U-Net: Convolutional Networks for Biomedical Image Segmentation. Ronneberger, O., Fischer, P., Brox, T. 2015. MICCAI 2015: Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015.

36. Deeply-supervised density regression for automatic cell counting in microscopy images. He S, Minn KT, Solnica-Krezel L, Anastasio MA, Li H. 2021, Medical Image Analysis, Vol. 68.

37. Random Forest With Learned Representations for Semantic Segmentation. Nguyen, B. Kang and T. Q. 7, 2019, IEEE Transactions on Image Processing, Vol. 28, pp. 3542-3555.

38. Fiji. [Online] Open Source. https://imagej.net/software/fiji/.

Curriculum vitae

Personal Details

Name	**Sergei Dobrovolskii**
Date and place of birth	03.02.1996 Moscow
Address	Koldestraße 10, 91052 Erlangen
Email	dobrik4ever@gmail.com
Phone number	+49 1516 3610896

Education

10/2019 – 09/2022	Master program Advanced Optical Technologies, Friedrich Alexander Universität Erlangen-Nürnberg
09/2017 – 07/2018	*Bachelor of Science (B.Sc.)* Bachelor Thesis: "Phase microscope based on multi-step phase calculation method"
09/2014 – 05/2018	Bachelor program Optical engineering, MIREA – Russian Technological University
09/2002 – 07/2014	School №875, Moscow

Professional Career

08/2020-09/2022	Interherence GmbH – Optical engineer

10/2019 – 07/2020	JSC "AMTEO M" - Software Developer. Developing the automated histopathology scanning brightfield microscope.
10/2018 – 10/2019	JSC "AMTEO M": • Chief Engineer. Developing the automated histopathology scanning brightfield microscope. • Field service engineer. Repairment of medical devices. • Crisis manager. B2B communication, factory processes repairment.
03/2016 – 08/2018	State budget vocational educational institution of Moscow "Polytechnic College No 47 named after VG Fedorov" – teacher (subjects: microcontrollers, electronics, robotics, programming)

Erklärung

Ich versichere, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegeben Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in dieser oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Declaration

I confirm that I have written this thesis without any external help and not using sources other than those I have listed in the thesis. I confirm also that this thesis or a similar version of it has not been submitted to any other examination board and has not been previously accepted as part of a exam for a qualification.

Erlangen, den 22.09.2022

Sergei Dobrovolskii