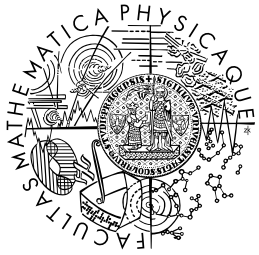


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Anna Mikeštíková

Analysis of meiotic spindle microscopy images captured with light-sheet microscope

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the bachelor thesis: Dr. Ing. Jan Schier
Study programme: Computer Science
Study branch: IPP5

Prague 2023

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

I am deeply grateful to my supervisor, Dr. Ing. Jan Schier, for all of his help, guidance, and especially patience during the writing of this thesis. Without his support and advice, I would not have been able to complete this work.

I would also like to thank RNDr. Michaela Schierová, Ph.D. for providing helpful feedback and suggestions regarding the biological background of the problem. A special appreciation goes to the research team from the Institute of Animal Physiology and Genetics CAS for providing the data and for their pleasant cooperation.

Finally, I would like to thank my family and friends for their continuous support and encouragement.

Title: Analysis of meiotic spindle microscopy images captured with light-sheet microscope

Author: Anna Mikeštíková

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Dr. Ing. Jan Schier, Department of Software and Computer Science Education

Abstract: Analysis of the meiotic spindle is an essential but time-consuming step in the research of the effects of the oocyte-meiosis regulating proteins. In this thesis, we propose a method for automated computation of the spindle volume as a step towards a more efficient analysis pipeline.

The proposed method is based on convolutional neural networks. It consists of two steps: segmentation of the spindle on a microscopic image and calculation of its volume. The segmentation is performed by a modified 3D U-Net architecture, which is trained on an augmented dataset of volumetric images.

The choice of architecture is supported by an in-depth analysis of the current state-of-the-art methods for image segmentation with a focus on biomedical images and volumetric data. The hyperparameters are tuned for the best performance on the dataset.

The model is evaluated on the testing dataset with respect to the segmentation quality and the volume estimation accuracy. The results demonstrate the feasibility of the proposed approach.

Keywords: meiotic spindle, microscopy imaging, image processing, image segmentation, deep learning methods

Contents

Introduction

The spindle apparatus is an intracellular structure responsible for chromosome separation during the act of nuclear division of a cell. The process of spindle formation is very complex, and it is regulated by various proteins. The female oocytes, which are responsible for reproductive cell production, are very sensitive to errors in the process of spindle formation. Any malfunction can lead to severe embryo damage or even miscarriage.

Because of these reasons, the processes of spindle formation and its development and the regulatory mechanisms in mammalian oocytes are being thoroughly studied by the scientific community. This results in large amounts of microscopic image data, which need to be processed by a human expert in order to extract relevant information. One of the key metrics used for evaluation is the volume of the spindle apparatus over time currently computed by manual process, which is a very time-consuming task. This makes the analysis process a good candidate for at least partial automation.

The automation of biomedical image analysis is a very researched topic because of the vast amounts of data generated in this domain. Manual data processing is very laborious and costly process, which requires the expertise of highly qualified medical or scientific personnel. High-quality results are required, while the human factor in processing inevitably introduces inconsistencies and errors.

Researchers have been investigating biomedical image processing since the end of the 20th century, initially using traditional computer vision techniques relying on low-level pixel processing. In recent years, deep learning methods have been introduced to this field and quickly superseded the traditional methods in accuracy, speed and scalability. Convolutional neural networks, in particular, are very effective with image data.

However, using deep learning methods for bioimaging brings challenges of its own. Contents of the image can be crowded, noisy, and difficult to distinguish. Additionally, the need for large amounts of well-annotated data is often unmet in this domain due to the high cost of manual annotation.

This thesis is focused on automating the analysis of the spindle apparatus with convolutional neural networks. The goal is to create a model which will be able to accurately compute the volume of the spindle from given volumetric images of mice oocytes.

The thesis consists of 6 chapters: The importance of biological research on spindle development is introduced in the first chapter. It is followed by the second chapter describing the data used for the training and evaluation of the models. The third chapter focuses on formulating the problem from the perspective of deep learning and computer vision, and the fourth chapter provides an overview of current state-of-the-art methods in biomedical image analysis with convolutional neural networks. Chapters five and six describe our implementation of the models and the results and evaluation of the experiments, respectively.

1. Biology background

1.1 Introduction

During sexual reproduction, the genetic information of two parent organisms combines to produce offspring genetically different from both parents. This process introduces variability into the population and is crucial for evolution.

Organisms produce cells partaking in sexual reproduction, gametes, in the process called *meiosis*. Meiosis is highly regulated by a number of proteins, which are responsible for meiosis initiation, correct genetic material distribution and genome integrity maintenance.

In this chapter, we will briefly describe meiosis and the importance of the spindle apparatus. Additionally, we will introduce a family of regulatory proteins and discuss their role in spindle formation based on current research.

1.2 Meiosis

Mitosis and meiosis are two distinct types of nuclear division that differ in their products and purposes.

Mitosis occurs in all body cells, and its purpose is to increase the number of cells, either for growth or repair. The two daughter cells generated by mitosis are identical to their mother cell.

Meiosis, on the other hand, is unique for germ cells, and the products are haploid gametes used for reproduction. It consists of two steps, Meiosis I and Meiosis II. During Meiosis I, two haploid daughter cells are created from one mother diploid cell, each inheriting one set of chromosomes. During Meiosis II, these daughter cells divide again, but this time their chromosomes are split in half because each half will belong to one of the next-generation daughter cells. This way, four haploid daughter cells are created from one parent diploid cell.

Furthermore, meiosis products differ depending on sex. Male meiosis, spermatogenesis, is even and produces four haploid sperm cells, but female meiosis, oogenesis, produces only one egg cell, which contains the majority of organelles and cytoplasm, while other daughter cells (polar bodies) perish.

1.3 Role of spindle apparatus

Mitosis and meiosis both rely upon the *spindle apparatus* to ensure DNA in the form of chromosomes will be separated evenly between daughter cells. In the case of Meiosis I, the spindle apparatus is responsible for pulling apart a homologous pair of chromosomes to facilitate cell division.

A spindle is a structure made from microtubules, protein molecules with tens of micrometers in length, constructed from small subunits of the protein tubulin. It has an oval shape with two poles. In animal cells, the spindle poles are generally organelles called *centrosomes* which contain *Microtubule-organizing centers (MTOC)* made of *centrioles* and pericentriolar material (PCM). From MTOCs,

microtubules grow towards the center of the spindle, where chromosomes are located, but also to the cell's membrane to anchor the complex and provide support.

During division, microtubules from opposite poles attach themselves to protein structures *kinetochores* located on the chromosomes, align all the chromosomes in an equatorial plane and pull them apart towards the poles.

1.3.1 Problems with spindle formation and maturation and their consequences

Correct spindle formation is crucial during meiosis, as the spindle ensures an even separation of a homolog pair of chromosomes into two daughter cells. An absence of a bipolar spindle or its malformation during meiosis in female oocytes can lead to aneuploidy, a genetic state in which the cells of an individual contain a non-standard number of chromosomes. Most cases of aneuploidy are incompatible with life, except for a few, e.g., trisomy of the 21st chromosome, more commonly known as Down Syndrome. Aneuploidy is the leading cause of miscarriage for the human species (Nagaoka et al. [2012]).

1.3.2 Spindle features of oocytes

Spindle formation in oocytes is different from other cells in the body because oocytes lack centrioles which are eliminated during their development. Consequently, the two centrosomes are not present to form the spindle. Instead, multiple *aMTOCs* (acentriolar Microtubule-organizing centers) are responsible for microtubule nucleation and spindle creation.

The aMTOCs undergo many changes during the maturation of the oocyte, including fragmentation, clustering into two poles, and migration to the cell's membrane. All of these processes are highly regulated and activated by appropriate proteins (Blengini et al. [2021]).

1.4 Aurora kinases

A family of three kinases, the *Aurora kinases*, is involved in the regulation of the spindle formation and development in mouse oocytes.

The first member, *Aurka*, is suspected to be responsible for maintaining the structure, number, and location of the aMTOCs at the poles of the spindle. Its absence results in short and disorganized spindles with over-clustered aMTOCs. Oocytes with such defects terminate their division process in the metaphase of Meiosis I. Consequently, the individual lacking *Aurka* would be sterile.

Multiple aMTOCs are involved in microtubule nucleation and spindle creation, and *Aurkc* plays a role in the process of clustering them into two poles. Failing in this task would result in a multipolar spindle and a higher risk of aneuploidy.

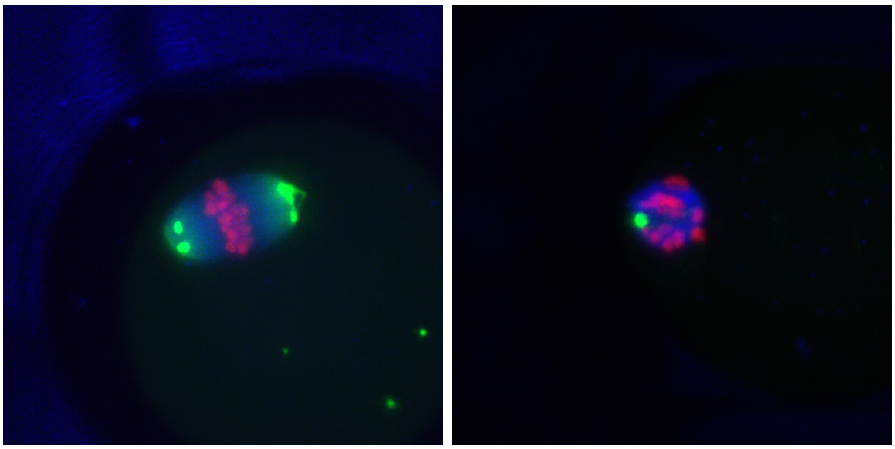
The precise function of the AURK family members and their ability to replace each other are under recent research (Blengini et al. [2021]). To be able to diagnose and explore the compensatory abilities and functions of the members of this family, it is crucial to examine specific spindle properties.

1.5 Evaluation methods

In order to discover the role of *Aurka* and diagnose whether other kinases would compensate for its loss, experiments on mice were conducted. Blengini et al. [2021] compared a strain of mice lacking this kinase in oocytes with a control mouse strain.

Oocytes harvested from both types of mice were observed and compared in various stages of maturation with an emphasis on the detection of certain events. The first event which is observed during analysis is the germinal vesicle breakdown (GVBD) — the disassembly of the nuclear envelope and the release of the chromosomes into the cytoplasm. After GVBD, the volume, shape and bipolarity of the spindle are measured and compared because they serve as markers for detecting if the oocyte is progressing through meiosis correctly.

In the absence of *Aurka*, some oocytes might not form spindles at all, other have defects such as mono or multipolar spindles or bipolar spindles of short lengths.



(a) (b)

Figure 1.1: (a) Spindle apparatus (blue) in a wild-type oocyte. The spindle is bipolar, aMTOCs are clustered at the poles (green), and chromosomes are aligned in the equatorial plane (red). (b) Spindle apparatus in an Aurka-deficient oocyte. The spindle is monopolar, aMTOCs are over-clustered at one pole, and chromosomes are disorganized.

2. Data collection and characteristics

During experiments with mice oocytes, their development is recorded over the course of several hours. The data is collected in the form of a time series of volumetric images.

The data we use in this thesis was collected with a light sheet fluorescence microscope system Vivents LS1 Live. In this chapter, we explain the advantages of light sheet microscopy for observing live cells and describe the properties of data collected by this system.

2.1 Fluorescence microscopy

Fluorescence microscopy is an optical microscopy technique using the fluorescent properties of molecules to visualize structures within a sample. Typically, antibodies labelled with fluorescence dye are introduced into the sample; they bind to molecules of interest and effectively label them. The prepared sample is then illuminated with ultraviolet light, which is absorbed by the fluorophore. The fluorophore emits emission light of a longer wavelength which is collected by a detector.

All current methods of microscopy have to make some trade-offs between image quality (resolution and contrast), acquisition speed, and light intensity. For long-term live-cell imaging, light intensity is a crucial factor because continuous exposition to light may result in the release of reactive oxygen, which is dangerous to organelles and other cell structures and can even cause cell death. This phenomenon is known as phototoxicity. Another inconvenience of light over-exposure is photobleaching, a process during which the structure of fluorescent molecules is gradually altered until they are no longer capable of emitting light.

Even advanced techniques, such as the laser-scanning confocal microscopy, suffer from the detrimental effects of heavy light exposure in the conditions of long-term live-cell microscopy. The reason is that the light travels through the sample in the directions perpendicular to the plane of interest, so for each image of a single plane, the whole sample is illuminated.

2.2 Data acquisition with light sheet microscope

The data used in this work was obtained with the light sheet microscopy technique. As follows from the description given below, imaging the whole slice in one exposure reduces the total exposure time and drastically reduces light dosage. These features make it suitable for fast 3D imaging of larger biological specimens.

In a light sheet microscope (see Figure 2.1), a laser beam goes through a cylindrical lens, which shapes it into a several-micrometer-thin sheet. This sheet of light then passes through the sample, illuminating only a slice without unnecessarily exposing the surrounding volume. A detection device is positioned perpendicularly to the illuminated plane to collect the light emitted by the fluo-

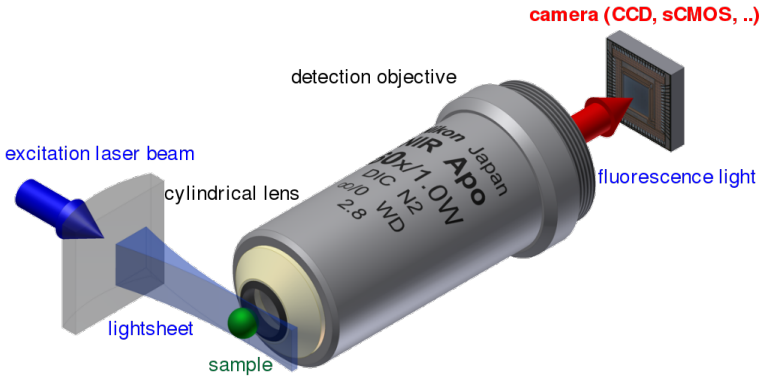


Figure 2.1: Light sheet microscopy
Source:https://en.wikipedia.org/wiki/Light_sheet_fluorescence_microscopy,
Licence: CC BY-SA 3.0

rophores. The sample is moved by a motor drive so that the light sheet illuminates multiple depth layers (slices), creating a 3D image. The process is then repeated in intervals, allowing to capture sample development over time.

2.3 Data characteristics

The dataset consists of data from 6 experiments, each obtained from a different mouse strain. From each experiment, we were provided with 5 to 20 oocyte observations. This resulted in a total of 70 oocyte observations (see Figure 2.2).

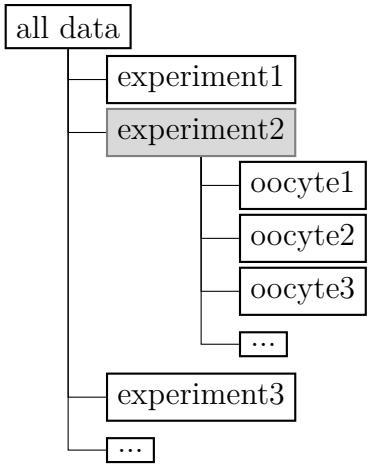


Figure 2.2: Structure of the dataset

Each oocyte was observed for several hours, with its image being captured and stored at an interval of 10 minutes. The resulting data for one oocyte observation is a time series of more than 100 volumetric images, each containing 31 slices (images) of 750×750 pixels. The distance between slices corresponds to 2 µm and the distance between individual pixels to $0,17 \times 0,17$ µm.

Samples were colored with three fluorescent dyes to label distinct structures within the cell. The emitted light was captured as three separate channels: chromosome channel, microtubule channel, and aMTOC channel (Figure 2.3).

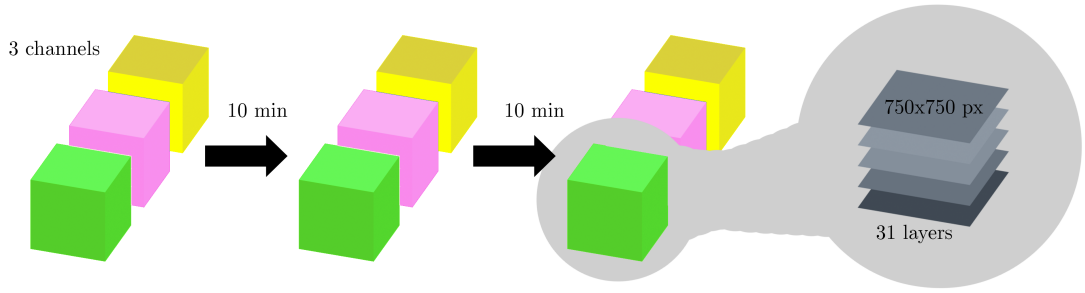


Figure 2.3: Diagram of a single oocyte observation. A 3-channel volumetric image is captured in an interval of 10 minutes. Each image has 31 slices of size 750×750 px.

In this thesis, only the microtubule channel is used for the task of predicting the volume of the spindle. The dataset is partially labelled – it contains information about the time of GVBD for each experiment and the volume and shape of the spindle for every sixth frame starting at GVBD. The label is provided as a segmentation mask of the spindle (Figure 2.4).

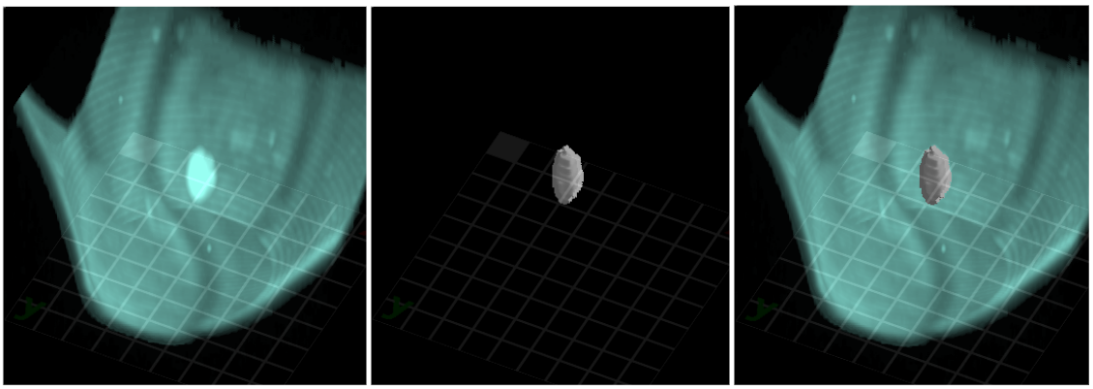


Figure 2.4: Microscopic image of the spindle with its segmentation mask. The microtubule channel (left), segmentation mask of the spindle (middle), both images overlaid (right)

3. Problem formulation

Because of the large amount of 3D microscopic images which have to be analyzed during Aurora kinase research, there is a strong need for automated data processing, especially in the automation of spindle volume estimation.

Values of the spindle volume during cell division represent an essential indicator of the cell's development and health. They can be used to determine the effect of Aurora kinases on the spindle development as described in Chapter 1. Since each experiment contains over a hundred images for each cell, evaluating the spindle volume is tedious. This is well illustrated by the fact that to make analysis reasonably feasible, only every sixth spindle image is analyzed now.

In this chapter, we will introduce our data in the context of various 3D data types. We will describe the most common tasks of computer vision and consider them in the context of predicting the spindle volume. We will also introduce deep learning as a suitable approach for solving our task. Finally, we will describe the challenges of working with biomedical data, with a focus on volumetric data and scarce annotations.

3.1 3D data types

3D image data come in many forms, each with specific properties, use cases and processing methods. In the industrial domain, the technologies used to capture images are only capable of scanning the surfaces and cannot penetrate the internal structure. The most common representations are point clouds, polygon meshes, and RGBD (RGB-Depth).

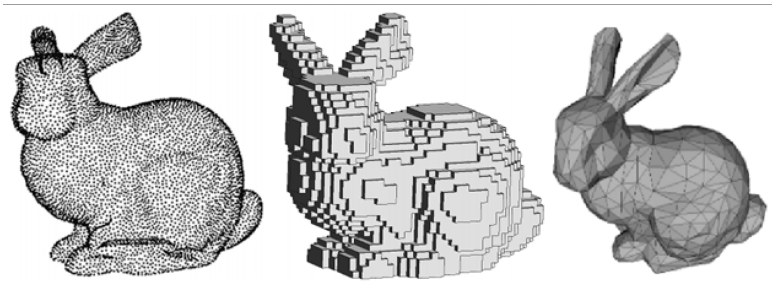


Figure 3.1: 3D data types visualized on the example of the Stanford bunny model. Point cloud (left), voxel (middle, ours), and 3D triangle mesh (right). Source: Hoang et al. [2019]

A remarkable property of biomedical image data is that we are able to capture the internal structure, and often it is also necessary to do so for the task at hand. The common modalities used to capture the internal structure of an organism are X-ray technology, ultrasound or magnetic resonance in the case of bioimaging or medical imaging and special microscopic techniques in the case of microscopy imaging. These technologies typically capture multiple 2D layers of the object and then stack them together to create a 3D volume, resulting in volumetric image data represented by voxels. This is also the case with the data used throughout this thesis.

3.2 Computer vision task definitions

Problems targeted by computer vision research cover many fields of application. In this thesis, we will focus on the tasks related to image analysis based on the content of the image.

Such tasks are typically divided into these categories:

- *image classification* – assign an image to one of the predefined classes, based on which one is the most likely to represent.
- *object detection, localization* – determine a position of an object in the image and mark it with a bounding box.
- *semantic segmentation* – find the image mask, i.e. assign each pixel as belonging to the class of the object it represents.
- *instance segmentation* – assign each pixel to a specific instance of an object.

These tasks are illustrated in Figure 3.2.

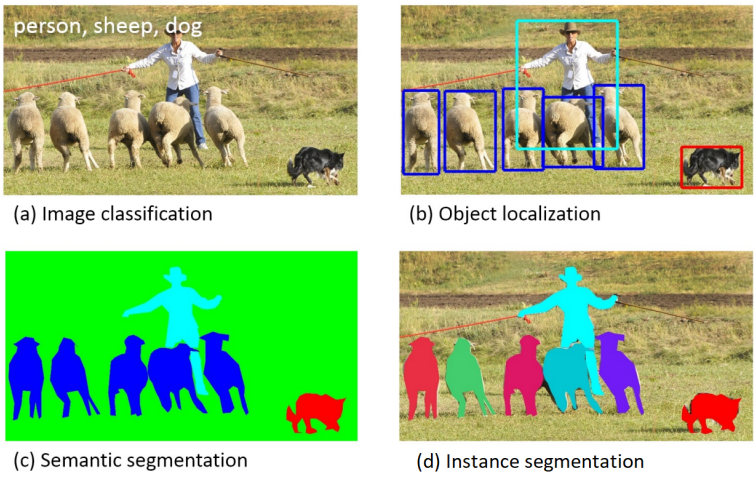


Figure 3.2: Computer vision tasks. Source: He et al. [2017]

3.2.1 Spindle volume estimation with semantic segmentation

For predicting the spindle volume, we decided first to perform the semantic segmentation of the spindle and then use the segmentation mask to determine the spindle volume. This is the same strategy used to compute the spindle volume manually. The volume will be calculated by counting the number of voxels belonging to the spindle and multiplying it by the volume of a single voxel.
The volume of one voxel is

$$V_{voxel} = 0,17 \times 0,17 \times 2 = 0.0578\,\mu m^3$$

The total volume of the spindle equals to:

$$\begin{aligned} V_{spindle} &= \sum_{i=1}^{N} 1_{\{x_i \in S\}} \cdot V_{voxel} \\ &= \sum_{i=1}^{N} 1_{\{x_i \in S\}} \cdot 0.0578\,\mu\text{m}^3 \end{aligned}$$

Because there is only one object of interest in the image, the task of semantic segmentation is equivalent to the task of binary semantic segmentation, i.e. to assign each pixel to one of two classes:

- *foreground* – the pixel belongs to the spindle
- *background* – the pixel does not belong to the spindle

3.3 Deep learning for medical image analysis

Litjens et al. [2017] describes the progression of the medical image analysis field from the early 1970s to the present times.

Initially, the methods were based on low-level pixel processing, mathematical modelling and rule-based systems. With the increasing popularity of supervised methods in the 1990s, the field had shifted towards the use of algorithms *learning* from the training data to solve the tasks. In these *pattern recognition* and *machine learning* systems, the computer learns to work with patterns and statistics based on example data. The data is in the form of feature vectors extracted from the images by hand by the domain experts.

To fully use the potential of the data, the *deep learning methods* were introduced for automated feature extraction. Deep learning methods are based on artificial neural networks, and with each layer of the network, they extract higher-level features from the training data. Consequently, the models based on deep learning, especially convolutional neural networks, have become a very successful approach to medical image analysis.

We decided to use convolutional neural networks for the task of semantic segmentation based on this success and their omnipresence in medical image analysis literature.

3.4 Challenges of 3D biomedical image segmentation

Biomedical data brings many challenges that need to be reflected in the methods used for processing it. The challenges are sometimes similar to the ones of non-medical data, but they affect the process to a considerably higher degree. Volumetric biomedical data take these problems even further, as described in Niyas et al. [2022]. Following are the key challenges they identified:

- **Expensive annotation process** – Training a deep learning model requires a large amount of labelled data. Biomedical data is difficult to annotate

and often requires a domain expert for the task, which is time-consuming and expensive. In the case of volumetric images, the annotation process consists of annotating each slice of the volume separately so the work for one data point is multiplied by the number of slices in the volume.

- **Dataset limitations** –For the reasons mentioned above, the datasets are often not labelled perfectly. The two groups of imperfect labels mentioned in Tajbakhsh et al. [2020] are *scarce annotations*, where only some observations are (fully) annotated, and *weak annotations*, where only part of the image is annotated. The latter includes sparse annotations (e.g. only some slices in a 3D image are annotated), scribble and point annotations, noisy annotations, etc.

- **Class imbalance** – Some biomedical data is highly imbalanced in terms of the number of voxels belonging to the different classes. Many methods developed for non-medical images do not take this into account, which may result in poor model performance due to "over-segmentation of classes with a high voxel share" (Niyas et al. [2022]).

- **Noisy data** – Due to the specifics of the acquisition process, the data in bioimaging and fluorescence microscopy contains more noise than non-medical images, often with rather complex structure (see Luisier et al. [2011]).

- **Computational cost** – The final challenge presented in Niyas et al. [2022] is the computation cost and substantial memory requirements of the models designed for volumetric data.

3.4.1 Challenges specific to our data

Of the challenges mentioned above, many are present in our data. First, we face the problem of scarce labels, as only every sixth image is annotated, resulting in around 600 labelled data points in total. The spindle is also very small compared to the size of the image, resulting in a very imbalanced dataset. The images are high resolution and volumetric, so processing them is both computationally- and memory-expensive.

Methods designed for biomedical image data need to be able to deal with these imperfections and very active research is being conducted in this area. We reflect on these challenges in the analytical part of our thesis in Chapter 4 as we introduce common methods used in deep learning and analyze the properties of the current state-of-the-art models. We will also try to address these challenges in the design of our model in Chapter 5.

4. Convolutional architectures

In this chapter, we introduce Convolutional neural networks, which are among the most popular methods for solving image segmentation problems nowadays. We will then introduce two approaches to image segmentation using CNNs: Top-down and bottom-up, along with respective examples and in the context of biomedical data. Finally, we will mention two models suitable for segmenting volumetric images with scarce or sparse annotations.

4.1 Convolutional neural networks

Deep convolutional neural networks (CNNs) are state-of-the-art for solving many computer vision tasks. They are a type of neural network specialized for data with grid-like topology, which includes images and volumetric data.

4.1.1 Overview of architecture

CNNs are composed of subsequent feature-extracting layers. Given an image, the network extracts more complex features with each layer and uses the final features to produce an output appropriate for the given task.

Each layer consists of three types of operations: convolution, non-linear activation and pooling. Usually, multiple convolutions are performed on the input in parallel to produce a set of linear activations. The activations are passed through a non-linear activation function to obtain a set of feature maps. The feature maps, in turn, are pooled and enter the next layer of the network.

In some works, all convolutions, non-linear activations, and pooling are considered individual layers of the network.

Convolution

Convolution is used to extract activations from an input. During this operation, a small tensor of trainable weights called a *kernel* or a *filter*, is applied to each position of the input tensor to detect a specific local feature.

In the mathematical context, discrete convolution is a binary operation which takes two input functions of discrete arguments and outputs another function. For two discrete, binary functions x and w, the convolution is defined as:

$$s(i_1, i_2) = \sum_{a_1=-\infty}^{\infty} \sum_{a_2=-\infty}^{\infty} x(a_1, a_2) w(i_1 - a_1, i_2 - a_2)$$

In CNNs, the inputs of convolution are finite multidimensional arrays, which behave as functions in a way that we can index them (input) to retrieve a value on a specific position (output). The values are defined as zero for indices out of the bound of the arrays so that the convolution can be computed as a sum over a finite number of elements. In the case of two-dimensional arrays, the convolution can be written as:

$$
\begin{aligned}
S(i,j) &= \sum_m \sum_n I(m,n)K(i-m,j-n) \\
&= \sum_m \sum_n I(i-m,j-n)K(m,n)
\end{aligned}
$$

where I is the input tensor (the input image or output of the previous layer), K is the convolutional kernel, and the output tensor S is one layer of the linear activation. Because the kernel is small, the second equation more intuitive as it represents the kernel sliding over the input image. It is valid because the convolution is commutative.

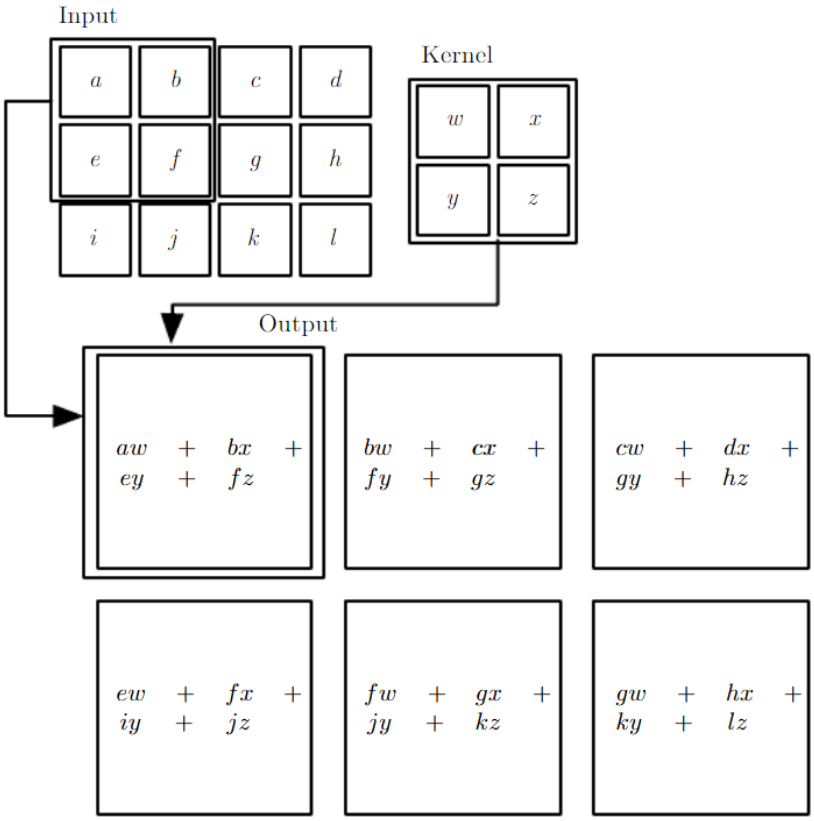


Figure 4.1: Example of convolution. The kernel is applied to each position of the input tensor to produce a linear activation output. In library implementations, cross-correlation is used instead of convolution, defined as $S(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$. Source: Goodfellow et al. [2016]

Activation

Because the convolution is a linear operation, the activations produced by the convolutional layers are always linear. To enable the network to learn non-linear features, we have to use non-linear activation functions. The most common activation function for the activation stage of CNNs is rectified linear unit (ReLU) defined as $f(x) = max(0,x)$. In the state-of-the-art networks, more versions of ReLu are used to reduce the so-called "dying ReLu problem", e.g. Leaky ReLU, ELU, PReLU, Swish.

Pooling

Pooling is used to summarize the features extracted by the convolutions. It shrinks the size of the feature map because it replaces certain regions of the map with their summary statistics to make the feature maps invariant to small translations of the input. The most common pooling operation is max pooling, which uses the maximum value of each region as the summary statistic and is used to determine if some feature is present in that region. Other options are average pooling or L2-norm pooling.

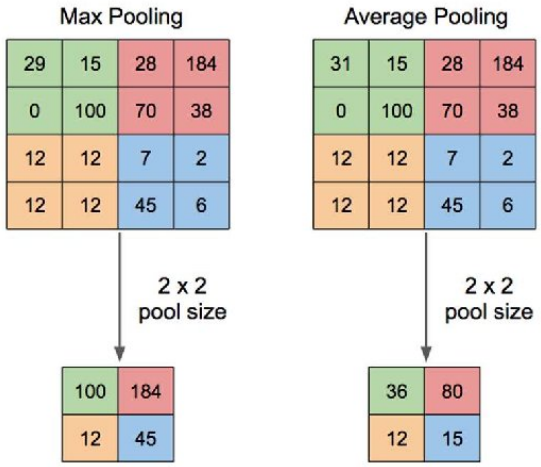


Figure 4.2: Two examples of pooling: Max Pooling and Average Pooling with a 2x2 pixel filter size. In max pooling, the maximum value of each region is used as the summary statistic, while in average pooling, the average value is used. Source: Yani et al. [2019], Available via license: CC BY 3.0

Backbones

The part of the convolutional neural network responsible for the feature extraction is called the *backbone*. It is composed of multiple layers described in the previous section. Backbones are sometimes pre-trained on large datasets and then fine-tuned for a specific task. This is called *transfer learning*. Some of the most famous backbones are VGG, ResNet, Inception, DenseNet, etc.

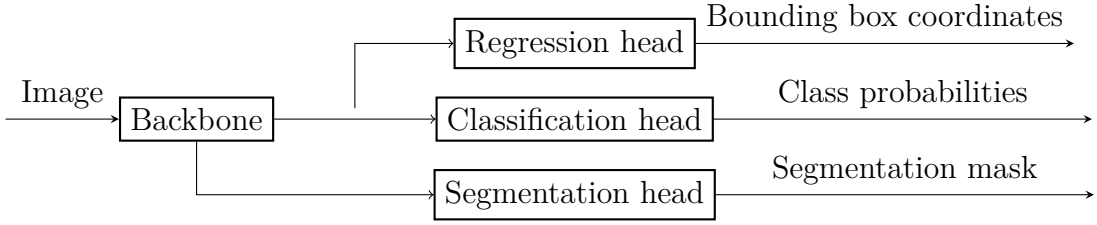


Figure 4.3: Example of a neural network with a backbone and multiple prediction heads. Simplified drawing of Mask R-CNN architecture from He et al. [2017].

Prediction heads

For each computer vision task, the neural network's final output is different. Even though the same backbone can extract the features maps, the requested output might be a single class, a bounding box, a mask, etc. The layer of the network responsible for producing the output is called the *prediction head*.

For tasks such as image classification, where the output is a vector of probabilities, the prediction head consists of multiple fully connected layers which take the flattened output of the backbone as an input.

Flattening would result in a loss of spatial information for the segmentation tasks, where the output is a mask of the same size as the input image. The network would not be able to reconstruct this mask to the image's original resolution (Long et al. [2015]).

The common approach to solve this problem is to use fully convolutional networks (FCN) proposed in Long et al. [2015]. The result of the backbone is passed to several layers of feature processing 1x1 convolutions (bottleneck) and then increased in size to the image's original resolution. This process is called upsampling and is usually done by an operation called transposed or upscaling convolution, which is considered to be the opposite of pooling.

4.1.2 Advantages of convolutional networks

The key idea behind the use of CNNs is that prior knowledge about the data (grid-like topology, invariance to translation) is used to constrain the network architecture to reduce the size of the search space of all possible networks without reducing its capability to represent the desired function. This makes the training faster and less prone to overfitting because many networks representing only the training data, not the testing data, are eliminated (LeCun et al. [1989]).

The advantages of this restrictive architecture of CNNs are emphasized in Goodfellow et al. [2016]. Firstly, CNNs have **sparse interactions** between layers because only a small subset (of the size of the kernel) of input features participates in the computation of each output. This reduces the number of parameters and makes the training more efficient and less memory-demanding. What improves these factors further is the **sharing of the parameters** of the kernel across all positions because the network does not need to learn a separate set of parameters for every location. In this way, parameter sharing ensures the **equivariance to translation** because the features learned by the kernel are positionally independent.

4.2 Training of convolutional neural networks

4.2.1 Performance metrics

Performance metrics are used for the evaluation of the performance of a model. They should be determined by the problem and need to be picked very carefully. In this section, we will discuss the common metrics used for binary segmentation tasks. We use the usual terminology, where the number of correctly predicted pixels belonging to the segmented object is the "number of true positives" (TP),

and in a similar fashion we define true negatives (TN), false positives (FP), and false negatives (FN).

Accuracy

Pixel accuracy is a very intuitive metric for binary segmentation tasks. It is computed as the ratio of the number of correctly predicted pixels to the number of all pixels:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy, unfortunately, suffers from a problem of class imbalance. When the segmented object occupies only a small portion of the image, and the background is much larger, predicting many of the pixels as the background will result in high accuracy.

Precision and recall

Precision and recall are two metrics which could be used for any binary classification task, but when used alone, they can be misleading. They are defined as:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

In some cases, precision and recall can be 100% even though the prediction is not very good. For precision, this is the case when a minimal number of pixels is correctly predicted to belong to an object. On the other hand, when the whole image is predicted as belonging to the object, the recall will be 100%, even though the prediction is not very good.

F1 score, Dice coefficient

For the reasons mentioned above, precision and recall should always be used together, to balance each other's weaknesses. Mathematically, this balance is achieved by the F1 score, which is defined as the harmonic mean of precision and recall:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

When used in the context of semantic segmentation, the F1 score is commonly referred to as the Dice coefficient, and it is used in cases of class imbalance. It can be viewed as a measure of the similarity between the predicted mask and the ground truth segmentation.

Jaccard index

A metric similar to the Dice coefficient is the Jaccard index (Intersection-over-Union, IoU). As the name suggests, it is defined as the ratio of the intersection of the predicted and ground truth masks to the union of the two. It is defined as:

$$Jaccard = \frac{TP}{TP + FP + FN}$$

4.2.2 Loss functions

Correct selection of a loss function is critical to train a neural network, and it should be chosen in accordance with the metric used for evaluation. The loss function is used to measure the difference between the label predicted by the network and the ground truth. During training, the network's weights are updated based on the gradient of the loss function in order to minimize it.

While choosing a loss function, the class imbalance should always be taken into account.

Binary cross-entropy loss

Binary cross-entropy is a loss function used for binary classification. It can also be used for binary segmentation because it can be viewed as a binary classification of each pixel.

Binary cross-entropy is justified by the information theory. When training a model, we optimize it by finding the parameters which would fit the training data well. This is called the maximum likelihood estimation of the parameters. Maximizing this likelihood is mathematically equivalent to minimizing the cross entropy:

$$w = argmax_w L(w) = argmin_w H(\hat{p}_{data}(X), p_{model}(X; w))$$

where w are the model parameters, $L(w)$ is the likelihood of weights, $\hat{p}_{data}(X)$ is the empirical distribution defined by the training data and $p_{model}(X; w)$ is the probability distribution defined by the trained model.

In the case of the binary cross-entropy loss function, the probability distribution defined by the model is the Bernoulli distribution. Consequently, the binary cross-entropy loss, which is minimized during training, is defined as:

$$L(t, p) = -\frac{1}{N}\sum_{i=1}^{N} t_i log(p_i) + (1 - t_i) log(1 - p_i)$$

where t_i is the ground truth label for the i-th pixel, and p_i is the predicted probability of the pixel belonging to the segmented object.

Binary cross-entropy, however, struggles significantly with class imbalance. Training models with this loss can be inefficient if there are many easy negatives that do not contribute any useful learning signal. When the segmented object is very small, the model learns to be biased towards the majority background class (Lin et al. [2020]).

Focal loss

Focal loss (Lin et al. [2020]) is an extension of the cross-entropy loss which addresses the class imbalance problem by introducing a focusing parameter γ. It is defined as:

$$FL(t, p) = -\frac{1}{N}\sum_{i=1}^{N} t_i (1 - p_i)^{\gamma} log(p_i) + (1 - t_i)(p_i)^{\gamma} log(1 - p_i)$$

where t_i is the i-th ground truth label, p_i is the i-th predicted probability of belonging to the segmented object, and γ is a tunable focusing parameter.

The focusing parameter down-weights the contribution of easy examples (with high confidence p_i) to the loss function because their scaling factor decays to zero. Hard examples (with low confidence p_i), on the other hand, are scaled up by the focusing parameter. The model does not get overwhelmed by the large number of easy negatives and can focus on the hard positives.

Dice loss

Dice loss is a loss function which optimizes the Dice coefficient introduced in the Section 4.2.1. It is defined as:

$$Dice(t, p) = 1 - \frac{2 \cdot \sum_{i=1}^{N} t_i p_i + \epsilon}{\sum_{i=1}^{N} t_i + \sum_{i=1}^{N} p_i + \epsilon}$$

where t_i is the ground truth label for the i-th pixel, and p_i is the predicted probability of if belonging to the object. ϵ is a small constant which avoids division by zero.

This definition of the Dice loss is smoothed by using the predicted probabilities and not the final binary predictions. If the final predictions were used, the loss would not be differentiable because it would be defined only for integer values, therefore, not continuous (Sudre et al. [2017]).

4.3 Top-down approaches for segmentation

There are two approaches towards image segmentation using convolutional neural networks: top-down and bottom-up. Models implementing the top-down approach first detect the objects and their location in the image, classify them and only then find the segmentation masks on the basis of this detection.

In this section, we will describe how the models using the top-down approach are trained for object detection and show modifications which have to be made to extend them to be capable of object segmentation.

4.3.1 Object detection

The object detection pipeline usually consists of three parts: the generation of ROI (Regions-of-Interest) proposals, classification and localization of an object on these proposals and pruning of the results.

Proposals are smaller regions of the original image in which objects of interest might be present. The idea of generating the proposals stems from an observation that by offering the network only a limited amount of interesting small regions, it becomes computationally less expensive to train it. The detection of the proposals is implemented by a CNN with two prediction heads: one for the classification of the object and one for the prediction the exact location of its bounding box. Sometimes the proposals overlap, and the same object is present in multiple proposals. To avoid the detection of the same object multiple times, the results are pruned using the non-maximum suppression method.

This pipeline was first proposed in Girshick et al. [2014] as R-CNN. The proposals were generated by a non-trainable algorithm (the selective search). Since then, the prevailing opinion has been that using handcrafted features or fixed algorithms instead of learning algorithms limits the network's potential to learn complex features that are difficult to detect by humans (Litjens et al. [2017]). Consequently, in Faster R-CNN (Ren et al. [2017]), a separate network is trained to generate the region proposals. Along with the performance improvements, they report a significant speedup of the pipeline.

YOLO, a single stage object detection

Redmon et al. [2016] perform detection by a single YOLO model jointly optimized for proposal generation and classification. Compared to Girshick et al. [2014], Girshick [2015], and Ren et al. [2017], it is much faster in predicting because the image passes through the CNN only once. Because the complete image is seen during training, YOLO is better at reasoning in the global context of objects, and it is not as confused by background noise as region proposal architectures are. While YOLO can be less accurate in the precise localization of smaller objects, it is more accurate in object classification (Redmon et al. [2016]).

4.3.2 Mask R-CNN for instance segmentation

Mask R-CNN proposed in He et al. [2017] is a direct extension of the proposal-based detection methods for the task of image segmentation. It uses Faster R-CNN architecture and extends it by adding a separate head for segmentation implemented by CNN terminated with a small, fully convolutional network (FCN).

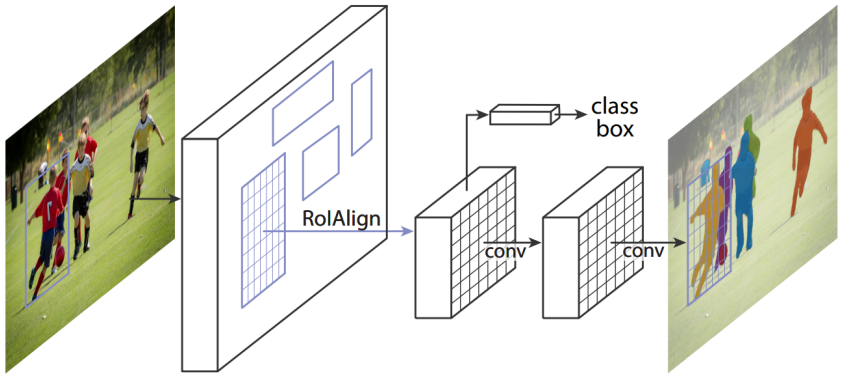


Figure 4.4: Mask R-CNN architecture with one head for object detection and a second head for mask prediction. Source: He et al. [2017]

For several years, Mask R-CNN and its modifications with varying backbones were the state-of-the-art in instance segmentation (MetaAI [2023]).

4.3.3 Usability for bioimaging

Top-down methods usually need to be adjusted for biomedical data because the bounding boxes are often axis aligned. Consequently, the methods work well

with natural images but perform poorly in some cases of biomedical data, which are randomly oriented and very complex (Lalit et al. [2022]). In Schmidt et al. [2018], the authors propose to localize the objects with star-convex polygons which represent the shape of their data better. Overall, however, the top-down methods are more suitable for the task of instance segmentation with multiple objects in the image. This is not the case for our task.

4.4 Bottom-up approaches for segmentation

Bottom-up methods do not rely on image detection; instead, they predict class probabilities for each pixel, and only then the classified pixels are grouped into the final objects. Sometimes the grouping is not even necessary, especially in the case of binary semantic segmentation, where the goal is to segment objects in the foreground from the background.

In the seminal work Ronneberger et al. [2015], the U-Net architecture has been proposed, which follows the bottom-up approach. Thanks to its performance in semantic segmentation, it has soon become widely adopted in bioimaging and related fields. It will be described in this section.

4.4.1 Fully convolutional networks for segmentation

Fully convolutional networks, as described earlier in this chapter, are at the core of all models for segmentation. They can also be used for the bottom-up image segmentation. The network architecture then usually consists of three parts: the encoder, the decoder, and skip connections between them.

The responsibility of the encoder is to take an image and extract a high-quality feature map, which is very semantically rich and contains a lot of context information. With each subsequent layer, the encoder obtains a feature map with a higher level of abstraction but lower spatial resolution (because of pooling). The decoder — by the use of transposed convolutions — expands the encoded feature map back to the original image size to produce the final segmentation mask.

Additionally, a convolutional bottleneck connects the encoder and the decoder with a reduced number of parameters and forces the network to learn only the most essential features.

Long et al. [2015] proposed the use of skip connections which combine fine-grained spatial feature maps from the upper layers of the encoder with the semantic, coarse-grained, upsampled feature maps from the lower levels of the decoder to increase the spatial resolution of the final mask and incorporate detailed localization (see Figure 4.5).

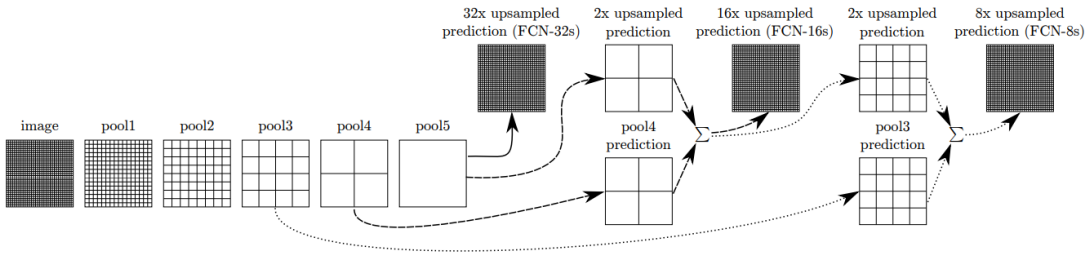


Figure 4.5: An example of FCN combining coarse, high-layer information with fine, low-layer information. The results of the encoder pooling layers are shown on the left with the appropriate resolution, upsampled result feature map is shown on the right with arrows representing the upsampling and skip connections are displayed by the summation signs. The intermediate convolution layers are omitted. Source: Long et al. [2015]

4.4.2 U-Net

In Ronneberger et al. [2015], a novel architecture called U-Net was proposed, with the aim of reducing the need for annotated data and providing accurate localization of the segmented output. Similarly to previous FCNs, U-Net has a classical encoder-decoder architecture. The main difference is that the decoder has a large number of feature channels, which allow the network to propagate context information to higher-resolution layers. As a result, we obtain an approximately symmetric U-shaped architecture (Figure 4.6).

Because of the typical application field in medicine, the desired result is often a binary segmentation mask.

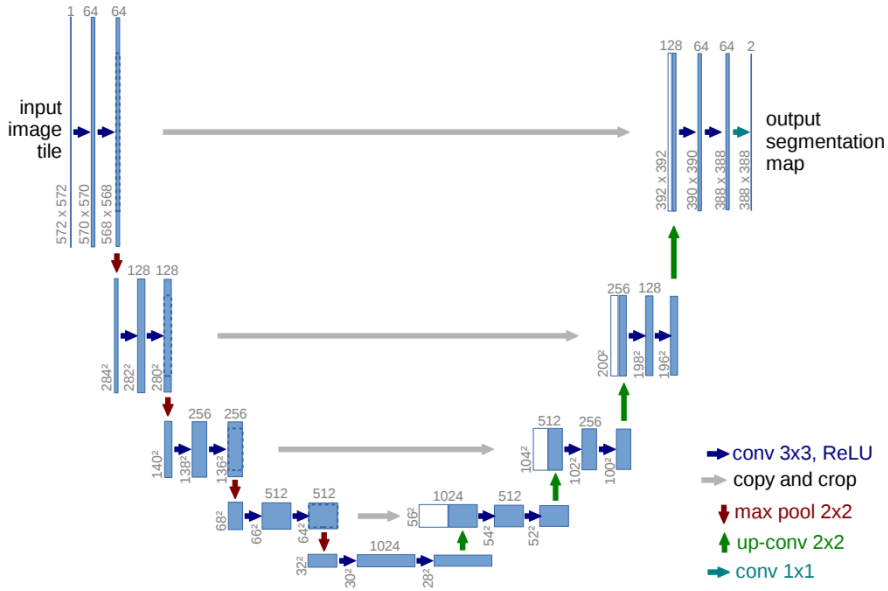


Figure 4.6: U-Net architecture. The contracting path on the left is the encoder, expanding path on the right is the decoder. Skip connections are shown by grey arrows. Source: Ronneberger et al. [2015]

4.4.3 Derivatives of U-Net

There are many derivatives of the U-Net architecture, with the aim to improve its performance by implementing ideas which work in other architectures. Some of them will be described in this section. A common theme shared by these examples is the adjustment of skip connections. They are key to the U-Net architecture but can be improved upon by adding more semantic meaning to the spatially richer feature maps of the encoder.

U-Net++, U-Net 3+

Zhou et al. [2018] argue that the skip connections in classical U-Net are not used to their full potential, because they combine feature maps from encoder and decoder, which have a different semantic meaning (feature maps from the encoder are less processed). The proposed architecture, **U-Net++**, uses a series of nested, dense skip pathways to enrich the feature maps of the encoder and unite their semantic meaning with the maps of the decoder. The model also uses deep supervision (supervision on hidden layers) by averaging the results of all skip pathways and applying the loss function to this average. Deep supervision is used to improve the training process but also enables the model to be pruned during inference because the more shallow pathways are more potent.

The resulting network can capture fine details more effectively and surpasses the original U-Net in performance in many medical image segmentation tasks.

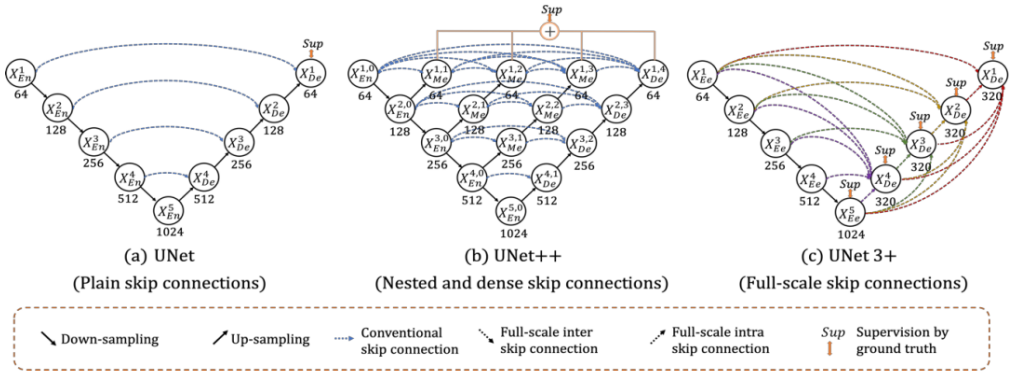


Figure 4.7: Comparison of U-Net (a), UNet++(b) and UNet 3+ (c). Source: Huang et al. [2020]

Huang et al. [2020] builds on top of Unet++ and proposes a new architecture, **Unet 3+**. Instead of dense skip connections, which combined information from multiple layers of the encoder, U-Net 3+ uses full-scale skip connections. The new skip connections connect all encoder layers to all decoder layers, which propagates a lot of spatial information to semantically rich layers of the decoder. There are also skip connections between decoder layers to keep the semantic information strong. Each decoder layer is supervised by a loss function, and the network is trained in a deep supervision manner.

Attention Unet

When the object of interest is small and highly variable in shape and size or when the image contains multiple objects, it is difficult for the vanilla U-Net to segment it correctly.

In some cases, this is solved by using a two-stage approach, by stacking two or more U-Net networks in a cascade manner. The first network is trained for coarse localization and segmentation in the sense of region proposals, while the second network is fed only with the proposals and performs fine segmentation. So the task has been decomposed into separated localization and segmentation steps.

This is, however, computationally expensive because segmentation is done on the same image multiple times (Oktay et al. [2018]). Attention gating was proposed in Oktay et al. [2018] to deal with this issue for a task of pancreas segmentation on a CT scan crowded with other organs. The attention gates are modules which are inserted into the skip connections of the U-Net architecture (which is more spatial) to boost their semantic information by combining them with more semantically rich feature maps.

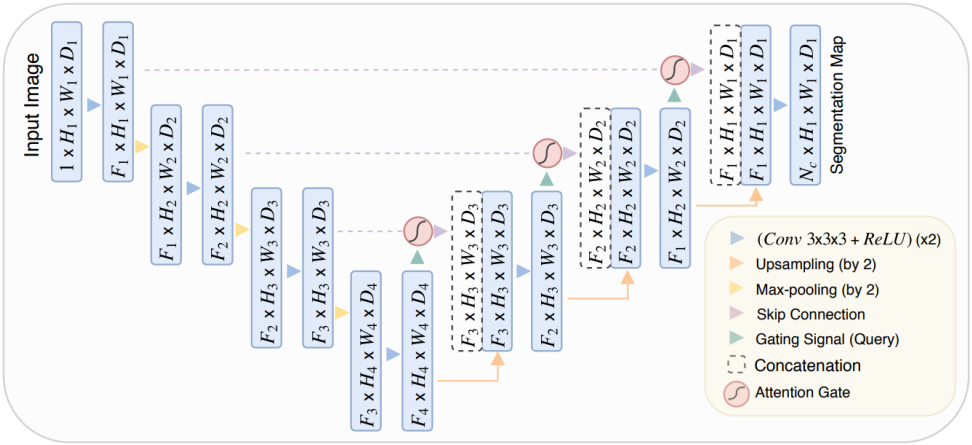


Figure 4.8: Attention U-Net architecture. Source: Oktay et al. [2018]

In practice, attention gates learn to assign weights to the skip connection feature maps to highlight their most important pixels. These weights can be interpreted as soft region proposals for the layer of a decoder to which they are connected. They learn to focus on the object of interest and don't need any additional supervision Oktay et al. [2018].

4.5 Architectures for 3D segmentation

Volumetric data is very common in biomedical imaging, and there is a high demand for models capable of processing it. Even though some recently proposed methods do contain pipelines for both 2D and 3D data, models for 3D data are still much less common (Lalit et al. [2022]).

Even though 2D models can be used for 3D data by processing each slice separately and combining the results, this approach is not optimal because it

ignores the spatial information between the slices, which could be used to improve the results.

Models designed for 2D data can be extended to process volumetric data directly by replacing the 2D operations with the corresponding 3D operations. Unfortunately, models operating on 3D data tend to be very computationally expensive and require a lot of memory.

3D convolutions

It is important to distinguish between 2D data with multiple channels (e.g. RGB images or fluorescence images with multiple dyes applied), and actual 3D data, which might have only one channel. Both can be represented by a 3D tensor, but the meaning of the third dimension is different, which should be reflected in their processing.

For multi-channel 2D data, the convolutional kernel is a 3D tensor of weights with the third dimension (the number of channels) of the same size as the input. Each value of the output feature map is then a weighted combination of values in all channels.

$$S(i,j) = \sum_m \sum_n \sum_c^3 I(i+m, j+n, c) K(m,n,c)$$

where m, n are indices of the kernel's two dimensions and c is the channel index, usually of size 3.

For one-channel 3D data, the third dimension of the kernel is semantically equivalent to the other two dimensions and is smaller than the corresponding input dimension. The kernel is applied to the input image in all positions in all three dimensions, which results in a 3D output feature map.

$$S(i,j,k) = \sum_m \sum_n \sum_o I(i+m, j+n, k+o) K(m,n,o)$$

where m, n, o are indices of the kernel's three dimensions.

3D U-Net

The direct extension of the 2D U-Net architecture for 3D data is the 3D U-Net Çiçek et al. [2016]. The architecture is very similar to the 2D version, the only major difference being the use of 3D convolutions, 3D max pooling and 3D up-convolutions in place of their 2D counterparts.

Interestingly, the authors implement this 3D U-Net in a way that does not use full annotations of the 3D volume. Instead, only several 2D slices are annotated along each axis of the volume, and an indicator is used to apply the loss function only in the annotated positions. This reflects the excessive cost of full annotation of the 3D volumes.

Semi-supervised methods, the DenoiSeg algorithm

Buchholz et al. [2020] work with an assumption that while annotated data is hard to get, unlabeled data is often available in large quantities. Their DenoiSeg is a U-Net shaped architecture trained for the task of segmentation and simultaneous

denoising to leverage the unlabeled data for improvement of the segmentation performance. A custom loss function is introduced, which is a combination of segmentation loss and denoising loss and is optimized jointly. It can be trained with only a few annotated ground truth data.

For denoising loss, authors take inspiration from Noise2Void, a self-supervised denoising model, which is able to remove noise from images without ever seeing a clean image introduced in Krull et al. [2019]. It uses patches of images and, by MSE loss function, is optimized to predict the central pixel of a patch from the surrounding pixels.

They offer two versions of their DenoiSeg model, one of which uses 3D operations and is suitable for work with volumetric data.

4.6 Transformer based models

In current research studies, vision transformers are being proposed as a solution to image analysis problems as an alternative to CNNs. On the COCO dataset competition for image instance segmentation, the current second place is taken by EVA (Fang et al. [2022]), a model employing the transformer-based architecture. Other computer vision tasks are also dominated by transformer-based models (MetaAI [2023])

5. Implementation

In this chapter, we introduce our implementation for the task of semantic segmentation of spindles in volumetric images. We first describe our data processing pipeline, which we used to prepare the data for training and prediction. Then we introduce the architecture of our *Baseline model* and the training process. Finally, we propose methods for improving the performance of the *Baseline model* and for improving its generalization capabilities.

5.1 Data

5.1.1 Train and test sets

We trained our model on the dataset described in section 2.2, which we split into training, validation, and testing sets. To obtain them, we split the original dataset two times.

First, we separated the testing data from the rest of the dataset. Our goal was to have a good testing set which would be independent of the training set and which would represent well the distribution of real data. Let us remember that the data, which is in the form of time-lapse records of volumetric images was acquired by observing the development of individual oocytes over time. In other words, each record contains images covering the evolution of a single oocyte over the time points (see 2.2). It would not be correct to split the data into the training and testing set randomly at the level of individual images because the images of the same cell can be expected to be not independent and the model would get some information about the testing data during training. Therefore, we decided to first split the data on the level of oocyte observations (also called "positions").

It was also important to preserve the distribution of the data in both sets. Cell observations so far come from 6 different experiments, each containing between 6 and 20 oocyte observations. To have all experiments represented in the testing set, we used stratified sampling with the experiments as strata and cell observation as a sample. Consequently, our test set contains Six cell observations, one is randomly chosen from each experiment, each of which contains 11 to 12 volumetric images.

The rest of the data was used for training and validation. Their split was done randomly on the level of individual images with a ratio of 80:20.

5.1.2 Data preprocessing

The typical preprocessing steps for image data are resizing, rescaling and cropping. Layers for these operations are available in both the TensorFlow and the PyTorch libraries and should be applied as needed to all images during both the training and prediction phases. To preserve their original resolution and all spindle details, we did not use image resizing. To lower the computation costs, we crop the volumes and retain the centre, which contains the spindle image. This was possible because the microscope was set to track the spindle structure during

imaging. Rescaling was not necessary because the images were already in the range [0, 1], which is ideal for neural networks.

Cropping and padding

The size of each image within our training data was 750×750×31 in x×y×z dimensions respectively. This was relatively large in comparison to the size of the spindle itself, and a large portion of the image was just an empty space. Cropping the images to the region of 352×352×31 voxels in the centre of the image was possible without losing any information. We can safely assume this will hold for the testing data and any future real data because the spindle is always in the centre of the image thanks to the tracking mechanism of the microscope.

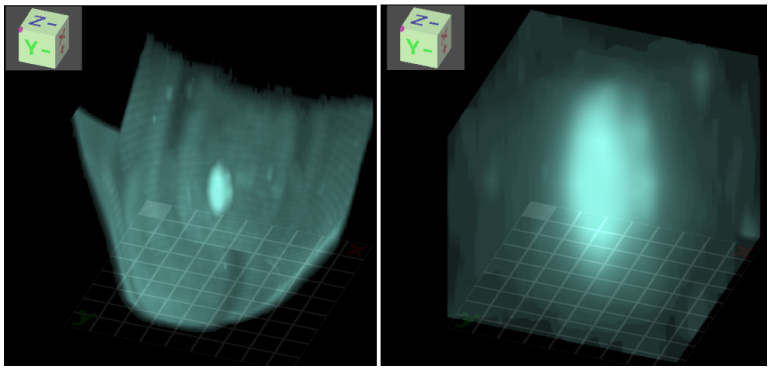


Figure 5.1: Image before and after centre cropping

We also had to pad the images by one layer to 352×352×32, because each dimension of the image should be divisible by 2^n for n pooling layers.

Contrast normalization

The global contrast of an image is computed as the standard deviation of all of the pixels in the image (Goodfellow et al. [2016]). In fluorescence microscopy, it is common for the images to have varying amounts of contrast based on the amount of fluorescent dye in the sample. For contrast normalization, we used the function `per_image_standardization()` included in the `tensorflow.image` package in the TensorFlow library which linearly scales the image to have a mean of 0 and variance of 1. Examples of images before and after contrast normalization are shown in fig. 5.2 on the following page.

5.2 Architecture

For the task of spindle segmentation, we decided to use the 3D Unet architecture described in Section 4.5 with several modifications. Our network was five layers deep. In the encoder, each layer consisted of two convolutional blocks with $(3, 3, 3)$ kernels, ReLu activation and max pooling. The number of convolutional filters was 8, 16, 32, 64, and 128, respectively.

During pooling, the size of an image is reduced by the factors defined by the pooling kernel, which normally stays the same for all dimensions in all layers. Let

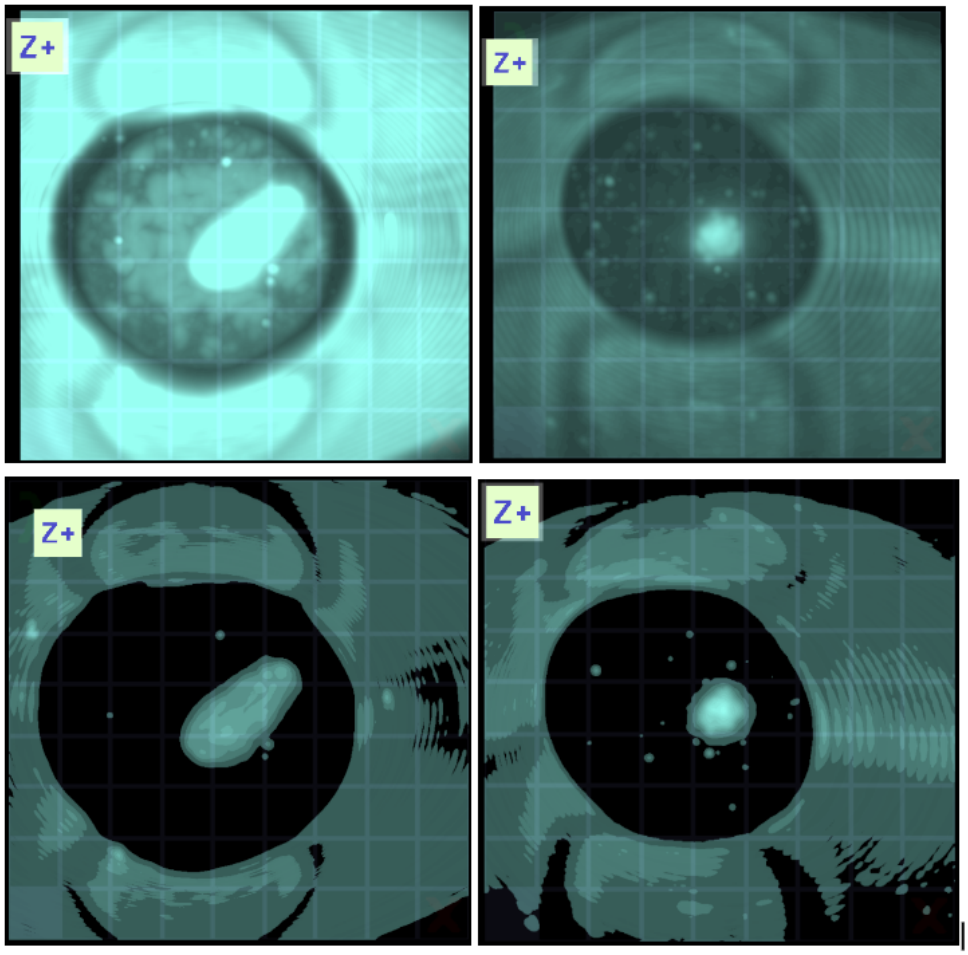


Figure 5.2: Images from two different experiments before(top) and after(bottom) contrast normalization. On the bottom images, only the values above 0 are shown due to the limitations of the displaying software. Posterization is caused by the displaying software, which is not able to display floating point values.

us recall that our image size is $352 \times 352 \times 32$, and the z dimension has $10\times$ lower resolution than the other two dimensions, which is described in Section 2.3. For this reason, we used a variable pooling kernel with dimensions $(2, 2, 1)$ for the first three layers and $2, 2, 2$ for the remaining two layers.

The decoder was symmetric to the encoder, where pooling was replaced by appropriate transposed convolution layers, and the number of filters was halved in each layer.

In classical U-Net, `valid` convolutions are performed. We chose to use `same` convolutions instead because we wanted to preserve the size of the image. The resulting network had 1.4M parameters.

5.3 Performance metrics

To measure the performance of our model, we use the Dice coefficient metric, which is described in Section 4.2.1 in more detail. For implementation purposes, we adjusted the formula by adding a small constant to the nominator and denominator to avoid division by zero.

$$Dice = \frac{2TP + \epsilon}{2TP + FP + FN + \epsilon}$$

We chose this metric because it is suitable for the task of binary segmentation and handles class imbalance well. IoU would also be a suitable metric, but it is positively correlated with the Dice coefficient, and therefore we decided to use only the Dice coefficient.

5.4 Loss functions

While it is possible to use Binary Cross Entropy as a loss function for binary segmentation, it is not suitable for tasks with high class imbalance. For this reason, we decided to use the Focal Loss and the Dice Loss described in Section 4.2.2.

In order to combine the advantages of the Focal Loss and the Dice loss, a combined loss function was used and computed as:

$$L(t, p) = \alpha FL(t, p) + (1 - \alpha)Dice(t, p)$$

where t is the ground truth segmentation, p is the predicted probability map of the pixels belonging to the segmented object and α is a tunable parameter used to balance the two loss functions. We used $\alpha = 0.5$. The Focal Loss was computed with coefficient $\gamma = 2$.

5.5 Training parameters

Optimizers and learning rate

For optimizing the objective function (minimizing the loss function), we used the Adam optimizer algorithm implemented in the `tensorflow.keras.optimizers` package. We have set the starting value of the learning rate to 5×10^{-5}. Initially, we tried higher starting values, but the training process was rather unstable.

The learning rate decay was implemented using the `ReduceLROnPlateau` callback from the `tensorflow.keras.callbacks` package. The decaying factor was set to 0.5 and patience to 15, so the learning rate was halved every 15 epochs, during which the validation loss did not improve. The minimal learning rate was set to 10^{-6}, as the model was not able to learn with lower values.

The Adam optimizer was chosen as it is often recommended as the default optimizer and it is reported to give good results and be rather fast in comparison with other optimizers.

5.6 Regularization techniques

During training, model's weights are adjusted to minimize the loss function for the training data. In cases when the amount of data is too small, the training is long, or the model is too complex, the training process may *overfit* the model, which will perform poorly on the test data.

To prevent overfitting and improve model's performance on unseen data, regularization techniques are used.

The regularization techniques aim to address the causes of overfitting. We have used data augmentation to increase the amount of training data and dropout to prevent the model from memorizing the training data (Goodfellow et al. [2016]). Also, the use of convolutional neural networks can be considered a regularization technique: as we discuss in Section 4.1.2, thanks to sparser connections, the CNN architecture is less prone to overfitting.

5.6.1 Data augmentation

We perform data augmentation by random rotations and flipping of the image slices. Each image from the training set had a 50% probability of being flipped in the x and y dimensions and then an equal 25% probability of being rotated by 0, 90, 180 or 270 degrees around the z-axis. By this data augmentation, we increased the amount of data by a factor of 8.

We do not flip the images along the z-axis or rotate them in the x and y dimensions, because the test pit, which holds the oocyte, may be visible in the image and it has an approximately conical shape along the z-axis.

We decided not to use any other data augmentation techniques, such as shifting, because the spindle is always in the centre of the image.

5.6.2 Dropout

Dropout is a regularization technique which randomly drops some fractions of the network weights. In a dropout layer, each weight may be dropped out independently with a probability of dropout rate p. In that case, it is not used during the forward pass through the network in a given epoch. (Srivastava et al. [2014]).

Dropout is commonly used in fully connected layers, however, Karshiev et al. [2020] describes its application in convolutional layers.

We used dropout layers for training in one of our models with a dropout rate of 0.15. When the dropout layers were used, they were applied before each max pooling in the encoder and before each transposed convolution in the decoder.

5.7 Attention gating

We have used attention gating described in Section 4.4.3 to improve the performance of our Baseline model because the spindle is variable in size and shape. When attention gates were used, they were applied only to the skip connections in the first layer of the decoder.

6. Experiments and Results

Our goal was to train a model which would be able to predict the volume of the spindle apparatus in a 3D image of a cell. For this task, we chose to train a model for semantic segmentation, which creates a segmentation mask of the spindle. Based on this mask, the spindle volume is computed.

Based on our research, we have used a 3D U-Net architecture adjusted for the needs of our data. We experimented with various hyperparameters to tune the performance of our model to achieve the best results.

In this chapter, we describe our experiments with *baseline model* and compare its performance with the performance of two other variants: model with dropouts and model with attention-gating. We trained them until convergence and evaluated the performance using the standard metrics for semantic segmentation and volume prediction. We evaluate performance on individual images, but also in the context of the time series of oocyte evolution and in the context of individual experiments, which will be more accessible for the cooperating biologists.

6.1 Graph sources

Several of the graphs of model training performance presented in this chapter were created using the TensorBoard visualization toolkit. They show the data for both the training and validation datasets, with the data points computed for each epoch. All of these graphs have a detailed description in the caption.

The remaining graphs were created using the matplotlib library.

6.2 Model training and performance

We have trained the models with various hyperparameters. In this chapter, we compare the performance of four particular models to demonstrate the effects of attention gating (A) and dropout (D). We refer to these models as *baseline model* (architecture described in Section 5.2), *model A* and *model D*, respectively, and, finally, *model AD* which combines both dropouts in the learning phase and attention gates.

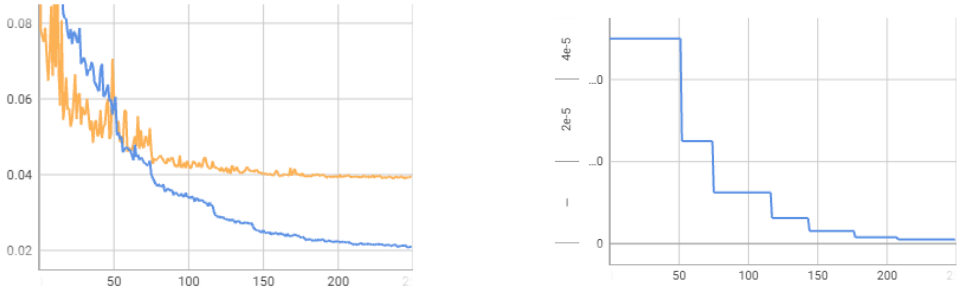
Models were trained with the same hyperparameters, all for a relatively short time, with low patience in lowering the learning rate.

Model Name	Attention gating	Dropout
baseline model	No	No
model A	Yes	No
model D	No	Yes
model AD	Yes	Yes

Figure 6.1: Models used for comparison of effects of the attention gating and dropout.

Training

Figure 6.2a shows the training and validation loss for our baseline model, computed by the combined loss function (see Section 5.4). We can see that both losses are decreasing during the training, which means the model is learning to predict correct values for the segmentation mask. After the 75th epoch, the rate of decrease of the validation loss starts to slow down; however, the training loss continues to decrease. This is a sign that we should use some regularisation methods to prevent overfitting.



(a) (b)

Figure 6.2: (a) Training (blue) and validation (orange) loss for the baseline model as a function of the number of epochs. (b) Learning rate development for the baseline model. Generated by TensorBoard.

We reduce the learning rate by a factor of 2 whenever the loss stagnates for over 15 epochs. This can be seen in the graph as a sudden decrease in the loss value. The learning rate decreased six times throughout the 250 epochs. When the learning rate is as low as 10^{-6}, the model cannot learn any more, and the loss stagnates.

Metrics

In Figure 6.3, we display the performance of our models for the segmentation task measured by the Dice coefficient metric over the course of training. During the training of the baseline model, the validation Dice coefficient stops improving approximately after 75 epochs. We have considered this a sign of the baseline model overfitting, and we decided to employ the regularization techniques.

The *model A* with attention gating achieves better results for both training and validation datasets than the baseline model and even narrow training and validation gap.

Results of the *model D* and *model AD* models demonstrate the effects of the dropout as a regularization method. While model A improved the gap between the training and validation Dice coefficients, *model D* and *model AD* were able to achieve even better results - their validation Dice coefficient reaches comparable or even higher values than the training Dice coefficient of the *baseline* model and *model A*.

We should note that the training Dice coefficients of *model D* and *model AD* models are *lower* than their respective validation Dice coefficients. This is considered normal behaviour, as dropout is applied during the training of these

models, so they do not use all information available during training. Dropout is not applied during validation; the models use all available information in this phase and achieve better results.

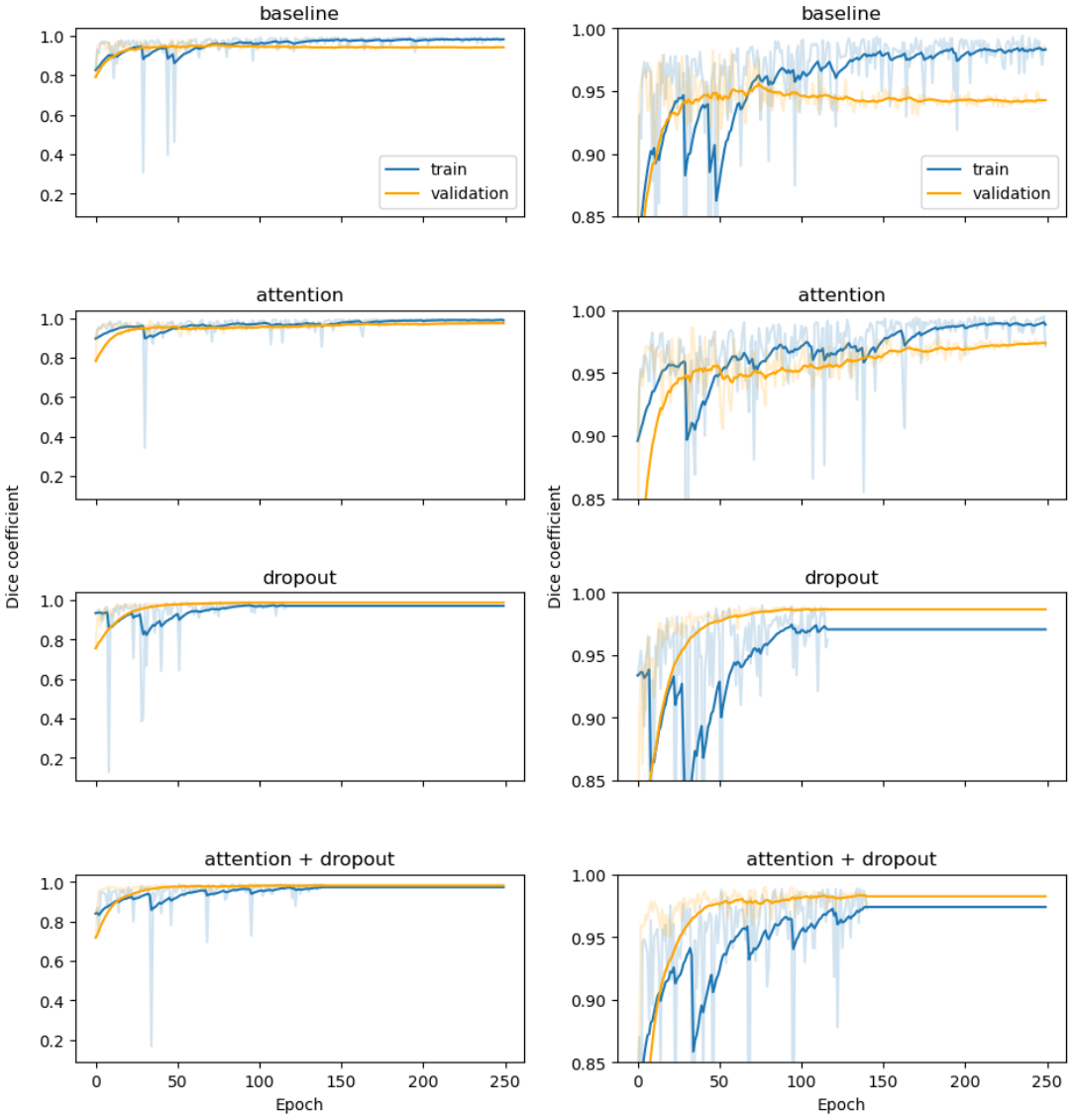


Figure 6.3: Training and validation performance measured by the Dice coefficient. The right column is a zoomed-in version of the left column. Values are smoothed by an exponential weighted average with a factor of 0.9.

For illustration, we also provide a graph of the accuracy metric in Figure 6.4. Considering the high imbalance of our data, accuracy is not a good metric. It increases to 99.8% in the first epoch, and for the rest of the training, it only changes in the fourth decimal place.

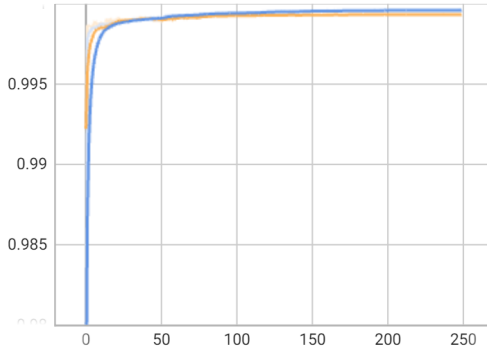


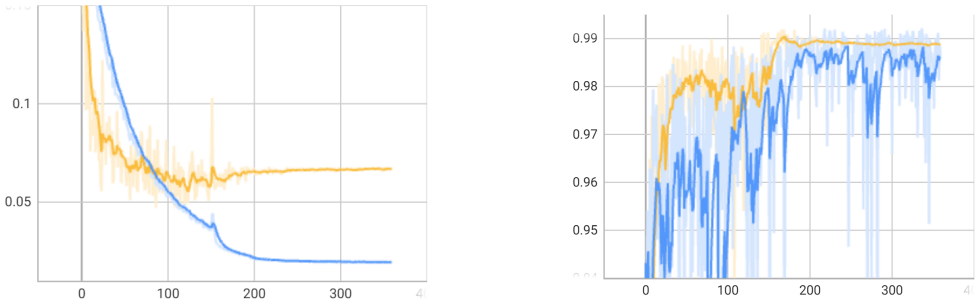
Figure 6.4: Accuracy for the baseline model as a function of the number of epochs. Generated by TensorBoard.

6.3 Extended training

To explore the effects of attention gating and dropout even further, we trained *model A* and *model AD* for longer time, resulting in models *A-long* and *AD-long*. In order to prevent the learning rate from dropping too quickly to a point where effective learning ceases, we initiated the process with a higher learning rate of 10^{-4} and increased the patience of learning rate decay. We have also increased the maximal number of epochs to 400, but the training has effectively stopped early for both models, with the validation loss stagnating for the last 150 epochs.

For the *AD-long* model, we added more dropout layers to the bottleneck of the U-Net architecture.

For the *A-long* model, the results did not provide any significant improvement over the short training. However, by longer training and fine-tuning the hyper-parameters, we were able to achieve a validation Dice coefficient of 0.989 for the *AD-long* model. We give the training and validation loss and Dice coefficient in the Figure 6.5.



(a) (b)

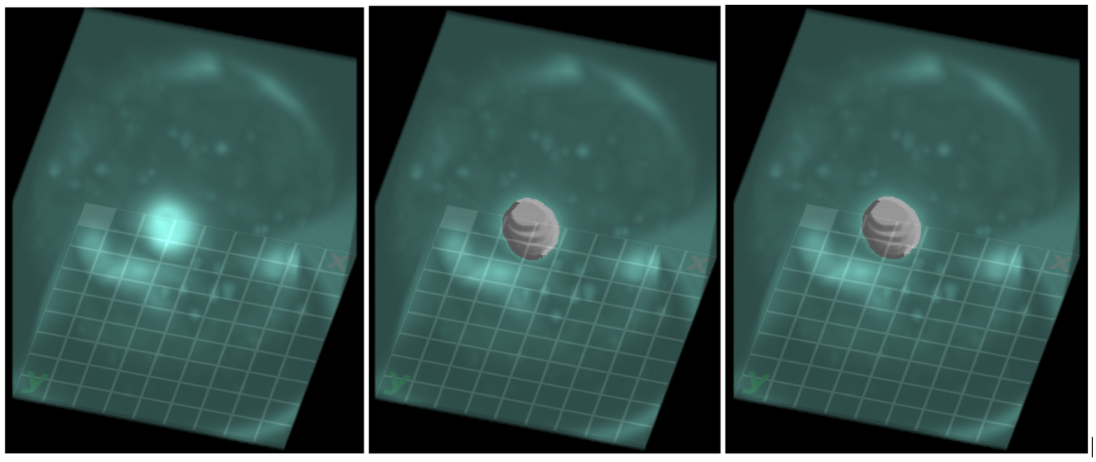
Figure 6.5: (a) Training loss (blue) and validation loss (orange) for the AD-long model. Loss (vertical axis) is plotted against the number of epochs (horizontal axis). (b) Dice coefficient plotted against the number of epochs. Generated by TensorBoard.

6.4 Results

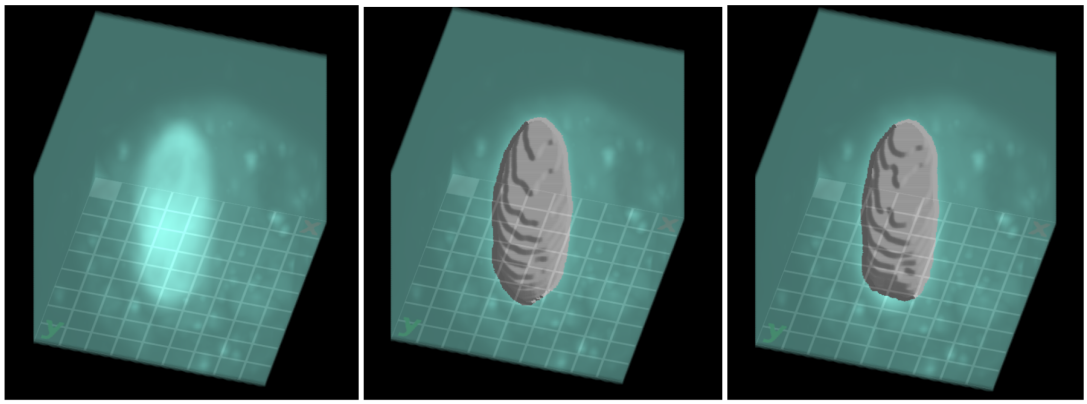
The models with longer training (*A-long* and *AD-long*) were evaluated on the test set consisting of seven time series (individual oocyte positions), altogether accounting for 55 images.

6.4.1 Segmentation performance

In Figure 6.6, we display the comparison between the segmentation mask predicted by our model *AD-long* and the ground truth segmentation mask for a single image from the test set. The model was able to predict the mask accurately without any artefacts. This is not a cherry-picked example but a typical result for our model. We present the results of the *AD-long* because it achieved the best results on the testing data with the Dice coefficient of 0.947 in comparison to 0.940 for the *A-long* model.



(a)



(b)

Figure 6.6: Examples of segmentation masks on the test set. Image without a label(left), image with the ground truth segmentation mask(middle), image with the predicted segmentation mask(right) for (a) image with background noise and (b) image of a spindle with an elongated shape.

6.4.2 Volume prediction performance

It is important for our task to predict the spindle volumes accurately and consistently for each time series of oocyte observations. In Figure 6.7 on page 39, we present the graphs of the volumes predicted by our two models for oocyte observations, along with their comparison to the ground truth volumes. The graphs show a comparison of the spindle volumes after GVBD because the volumes before GVBD are not relevant from a biological point of view, and ground truth masks and volumes were not provided.

In figure Figure 6.8 on page 40, we present the relative errors of the predicted volumes for each time series. The most significant relative error is present in the earlier time points, soon after GVBD when the spindle is small, so even a small error in the segmentation mask can lead to a large relative error in the volume. The relative error decreases as the spindle grows.

Interestingly, the dropout model *AD-long* is systematically worse at predicting the volume of the spindle close to GVBD but better at predicting the volume of the spindle later in the time series.

6.4.3 Discussion

The results of our experiments show that the *AD-long* model with attention gating and dropout performs slightly better than the *A-long* model with only attention gating. Both models, however, perform well on the tasks of segmentation *and* volume prediction. The segmentation masks predicted by our models are accurate and without artefacts. The volume predictions are consistent with the ground truth volumes. The curve of the development of the spindle volume over time has a similar shape to the ground truth curve for both models.

Based on these results, it is our opinion that the model can be considered for deployment in the cooperating biology laboratory.

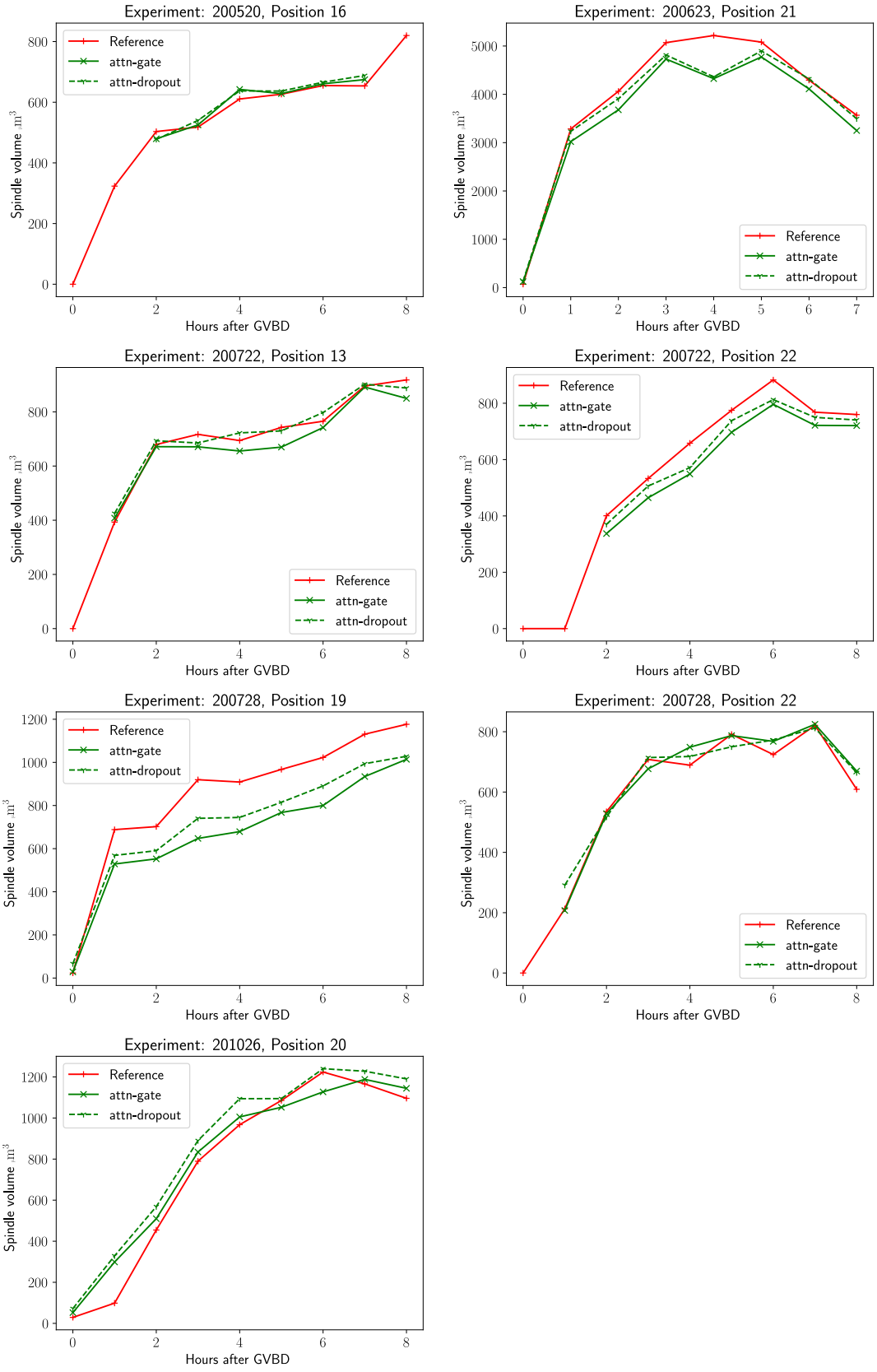


Figure 6.7: Volumes predicted by model A-long as attn-gate and model AD-long as attn-dropout for time series of oocyte observations with a comparison to the reference volumes.

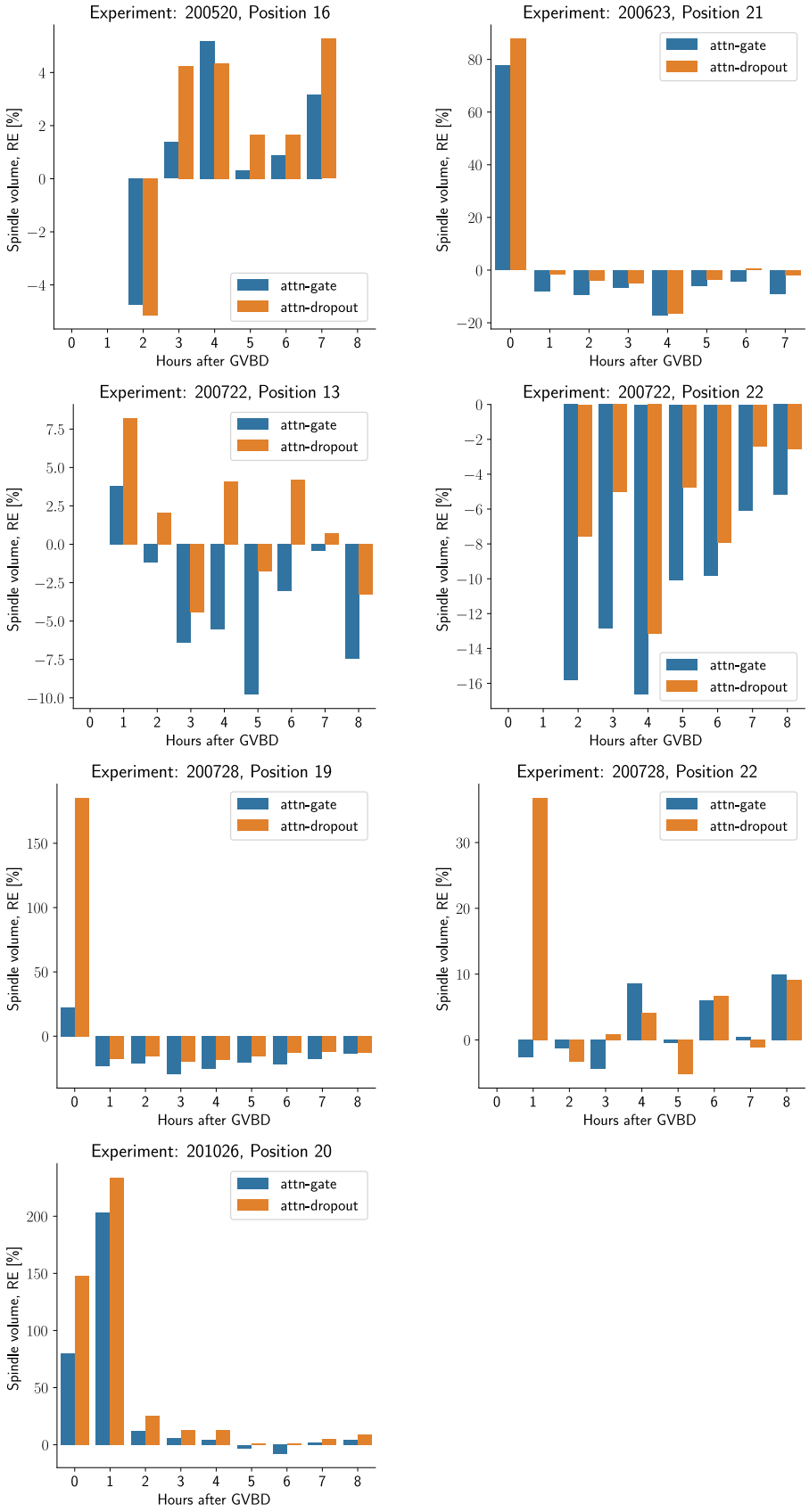


Figure 6.8: Relative errors of the volumes predicted by models A-long and AD-long in relation to reference volumes for time series of oocyte observations.

Conclusion

Our task was to propose a method for the automated processing of microscopic time-lapse images of the meiotic spindle in mice oocytes. The goal was to facilitate the analysis of the spindle evolution. We have decided to focus on the prediction of the spindle volume because it is the most time-consuming step in the manual analysis. In order to respect the pipeline of the current manual process, we approached the problem as a segmentation task and the consequent volume calculation. Additional advantages of this approach are that it is transparent to the biologists and that it allows us to compare our results with the reference ones obtained by manual processing.

We have performed research on the state-of-the-art methods for segmentation and detection with a focus on bioimaging, microscopy imaging and processing of volumetric data. We have reviewed convolutional neural networks as a powerful tool used for many computer vision tasks and justified their use in our case. We described several architectures suitable for semantic segmentation, along with their advantages and disadvantages.

Based on this research, we have chosen a slightly customized 3D U-Net architecture as our segmentation model. We have performed extensive experiments to find the best hyperparameters, and we have also experimented with several architecture updates to the model to improve its performance.

We have evaluated the best-performing models on the test set in terms of segmentation performance and volume prediction and presented the results in order to demonstrate the feasibility of our approach. We believe that our model can be used as a part of an automated pipeline for the analysis of meiotic spindles in mice oocytes.

To improve the volume prediction even further, we propose exploring the possibility of using the DenoiSeg model, mentioned in Section 4.5, on the dataset. This model leverages the unannotated data (which represents 5/6 of the dataset volume) to improve the segmentation performance. Unfortunately, we were not able to include the results of this model in this thesis because the official DenoiSeg implementation did not support data as large as those contained in our dataset, and the model would need to be reimplemented with modifications reflecting their size.

In future work, the pipeline should be extended to cover more of the manual steps, such as the detection of GVBD, detection of the start of chromosome separation (anaphase) and determination of the spindle polarity.

Last but not least, an easy-to-use interface will be necessary for practical deployment of the pipeline to be successful.

Bibliography

Cecilia S. Blengini, Patricia Ibrahimian, Michaela Vaskovicova, David Drutovic, Petr Solc, and Karen Schindler. Aurora kinase a is essential for meiosis in mouse oocytes. *PLoS Genetics*, 17, 4 2021. ISSN 15537404. doi:10.1371/journal.pgen.1009327.

Tim-Oliver Buchholz, Mangal Prakash, Deborah Schmidt, Alexander Krull, and Florian Jug. Denoiseg: Joint denoising and segmentation. In Adrien Bartoli and Andrea Fusiello, editors, *Computer Vision – ECCV 2020 Workshops*, pages 324–337, Cham, 2020. Springer International Publishing. ISBN 978-3-030-66415-2.

Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation. In Sebastien Ourselin, Leo Joskowicz, Mert R. Sabuncu, Gozde Unal, and William Wells, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, pages 424–432, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46723-8.

Yuxin Fang, Wen Wang, Binhui Xie, Quan Sun, Ledell Wu, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva: Exploring the limits of masked visual representation learning at scale, 2022.

Ross Girshick. Fast R-CNN. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. doi:10.1109/ICCV.2015.169.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014. doi:10.1109/CVPR.2014.81.

Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. http://www.deeplearningbook.org.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. doi:10.1109/ICCV.2017.322.

Long Hoang, Suk-Hwan Lee, Oh-Heum Kwon, and Ki-Ryong Kwon. A deep learning method for 3d object classification using the wave kernel signature and a center point of the 3d-triangle mesh. *Electronics*, 8(10), 2019. ISSN 2079-9292. doi:10.3390/electronics8101196. URL https://www.mdpi.com/2079-9292/8/10/1196.

Huimin Huang, Lanfen Lin, Ruofeng Tong, Hongjie Hu, Qiaowei Zhang, Yutaro Iwamoto, Xianhua Han, Yen-Wei Chen, and Jian Wu. Unet 3+: A full-scale connected unet for medical image segmentation. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1055–1059, 2020. doi:10.1109/ICASSP40776.2020.9053405.

Sanjar Karshiev, Bekhzod Olimov, Jaeil Kim, Jaesoo Kim, Anand Paul, and Jeonghong Kim. Improved u-net: Fully convolutional network model for skin-lesion segmentation. *Applied Sciences*, 10:3658, 05 2020. doi:10.3390/app10103658.

Alexander Krull, Tim-Oliver Buchholz, and Florian Jug. Noise2void - learning denoising from single noisy images. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2124–2132, 2019. doi:10.1109/CVPR.2019.00223.

Manan Lalit, Pavel Tomancak, and Florian Jug. Embedseg: Embedding-based instance segmentation for biomedical microscopy data. *Medical Image Analysis*, 81:102523, 2022. ISSN 1361-8415. doi:https://doi.org/10.1016/j.media.2022.102523. URL https://www.sciencedirect.com/science/article/pii/S1361841522001700.

Yann LeCun et al. Generalization and network design strategies. *Connectionism in perspective*, 19(143-155):18, 1989.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, 2020. doi:10.1109/TPAMI.2018.2858826.

Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017. ISSN 1361-8415. doi:https://doi.org/10.1016/j.media.2017.07.005. URL https://www.sciencedirect.com/science/article/pii/S1361841517301135.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015. doi:10.1109/CVPR.2015.7298965.

Florian Luisier, Thierry Blu, and Michael Unser. Image denoising in mixed poisson–gaussian noise. *IEEE Transactions on Image Processing*, 20(3):696–708, 2011. doi:10.1109/TIP.2010.2073477.

MetaAI. Instance segmentation on coco test-dev, 2023. URL https://paperswithcode.com/sota/instance-segmentation-on-coco. Accessed: 2023-05-01, License: CC BY-SA.

So I. Nagaoka, Terry J. Hassold, and Patricia A. Hunt. Human aneuploidy: Mechanisms and new insights into an age-old problem. *Nature Reviews Genetics*, 13:493–504, 7 2012. ISSN 14710056. doi:10.1038/nrg3245.

S. Niyas, S.J. Pawan, M. Anand Kumar, and Jeny Rajan. Medical image segmentation with 3d convolutional neural networks: A survey. *Neurocomputing*, 493:397–413, 2022. ISSN 0925-2312. doi:https://doi.org/10.1016/j.neucom.2022.04.065. URL https://www.sciencedirect.com/science/article/pii/S0925231222004362.

Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. Attention u-net: Learning where to look for the pancreas, 4 2018. URL http://arxiv.org/abs/1804.03999.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. doi:10.1109/CVPR.2016.91.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017. doi:10.1109/TPAMI.2016.2577031.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4.

Uwe Schmidt, Martin Weigert, Coleman Broaddus, and Gene Myers. Cell detection with star-convex polygons. *CoRR*, abs/1806.03535, 2018. URL http://arxiv.org/abs/1806.03535.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M. Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 240–248, Cham, 2017. Springer International Publishing. ISBN 978-3-319-67558-9.

Nima Tajbakhsh, Laura Jeyaseelan, Qian Li, Jeffrey N. Chiang, Zhihao Wu, and Xiaowei Ding. Embracing imperfect datasets: A review of deep learning solutions for medical image segmentation. *Medical Image Analysis*, 63:101693, 2020. ISSN 1361-8415. doi:https://doi.org/10.1016/j.media.2020.101693. URL https://www.sciencedirect.com/science/article/pii/S136184152030058X.

Muhamad Yani, M.T. Budhi Irawan S, Si., and M.T. Casi Setiningsih S.T. Application of transfer learning using convolutional neural network method for early detection of terry's nail. *Journal of Physics: Conference Series*, 1201 (1):012052, may 2019. doi:10.1088/1742-6596/1201/1/012052. URL https://dx.doi.org/10.1088/1742-6596/1201/1/012052.

Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In *Deep Learning in Medical Image Analysis and Multimodal Learning*

for Clinical Decision Support, pages 3–11, Cham, 2018. Springer International Publishing. ISBN 978-3-030-00889-5.

List of Figures

List of Abbreviations

MTOC Microtubule Organizing Center

aMTOC Acentriolar Microtubule Organizing Center

PCM Pericentriolar Material

Aurka Aurora Kinase A

AURK Family of Aurora Kinases

GVBD Germinal Vesicle Breakdown

RGBD Red Green Blue Depth

CNN Convolutional Neural Network

ReLU Rectified Linear Unit

ELU Exponential Linear Unit

PReLU Parametric Rectified Linear Unit

R-CNN Region-based Convolutional Neural Network

FCN Fully Convolutional Neural Network

TP, TN, FP, FN True Positive, True Negative, False Positive, False Negative

IoU Intersection over Union

ROI Region of Interest

YOLO You Only Look Once

CT Computed Tomography

COCO Common Objects in Context