



Technical University of Munich
Department of Physics
Chair of Cellular Biophysics (E27)

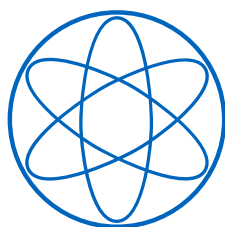
Analysis of 3D Cell Culture Experiments with Machine Learning Methods

Sándor Battaglini-Fischer

Thesis submitted for the degree of
Bachelor of Science (B.Sc.)
Physics with a Focus on Biophysics

Supervisor: Prof. Dr. Andreas Bausch
Advisor: Fabian Englbrecht

December 11, 2022



©2022 – SÁNDOR BATTAGLINI-FISCHER
All rights reserved.

I hereby confirm that this bachelor's thesis is the product of my own work and I have documented all sources and material used. It has not been submitted for any other degree or purpose.

A handwritten signature in black ink. The first part is a large, stylized 'S' that loops around. To the right of the 'S' is the word 'Sándor' in a cursive script. Below 'Sándor' is the name 'Battylini-Fischer' also in a cursive script.

Sándor
Battylini-Fischer

Analysis of 3D Cell Culture Experiments with Machine Learning Methods

ABSTRACT

Three-dimensional cell cultures such as organoids are a useful platform for biomedical research. They can be generated from tumor cells and are embedded in an extracellular matrix. Their differentiation and proliferation patterns are similar to their originating in-vivo cells. This makes them valuable as a test environment for tumor drugs, especially those that affect organoid growth. It is therefore crucial to be able to effectively analyse organoid growth behaviour in three dimensions. This is a tedious process when done manually, because time-series microscopy provides large datasets.

U-net is a convolutional neural network (CNN) that specialises in image segmentation. This thesis demonstrates that U-net effectively aids in the binary separation of the organoid from its background. These masks are then used to create a 3D model, which proves useful for visualisation and analysis purposes.

The branching behaviour, growth patterns and volume and surface area development were subsequently analysed. It was found that organoids show a differentiated growth pattern and that branching events are preceded by bulging at the inception points. Differences were found between longitudinal oriented cell structures and more branched ones, where the latter gain in volume much more rapidly.

These insights pave the way for a more effective analysis of growth patterns which result from drug response.

Contact details:

Sándor Battaglini-Fischer

Enrolment number: 03732406

sandor.battaglini-fischer@tum.de

Contents

1	Introduction	1
2	Background	2
2.1	3D Cell Culture	2
2.1.1	Organoids	2
2.2	Microscopy	5
2.3	Machine learning in cell biology	7
3	Methods	12
3.1	Cell biology	12
3.2	Microscopy of 3D organoid cell cultures	13
3.3	Data processing and analysis	15
3.3.1	Pre-processing	15
3.3.2	Image segmentation in 3D	16
3.3.3	U-net: CNN-aided image segmentation	19
3.3.4	3D representation	23
3.3.5	Hard- and software	24
4	Results	25
4.1	Image segmentation performance in 2D	25
4.2	Visualisation of 3D model	29
4.3	Volumetric analysis of 3D model	33
5	Discussion	37
6	Conclusion and Outlook	39
A	Datasets	40
B	Code	42
	References	48
	List of Figures	52
	List of Tables	55
	Acknowledgements	57

1 Introduction

Cancer is a major cause of death worldwide, accounting for around 10 million deaths per annum[1] and 8 million additional diagnoses [2]. In developed countries, it lies just behind cardiovascular diseases as the leading cause of death [3]. Thus, oncology has been a main focal point in biomedical research in recent years, and great strides have been made to construct an efficient system to prevent, detect and treat all types of cancer.

In order to ensure this process is fast and accurate, it is essential to tailor the therapy to the patient. The translation of scientific models from the lab to the patient's specific situation historically fails in a majority of cases when using traditional models of drug testing, namely 2D cultured human tumour cell lines and rodent xenografts [4].

This is where 3D cell cultures come into play. Organoids are an example of 3D cell cultures. They construct a more natural environment that reacts to external stimuli in a similar manner and construct a stable and more accurate representation of in-vivo conditions. The term "organoid" in this sense was used for the first time in 1946 [5] to refer to a teratoma [6], but only in recent years has it gained the "three-dimensional culture model" meaning it has today[7]. Since the development of the intestinal organoid culture system in 2009 [8], the research field has gained even more momentum.

To be able to evaluate the effects of drug treatments, it is essential to understand the growth patterns of single organoids. The spacial development provides information on growth phases [9], reactions to drugs [10], and can aid in understanding the expansion of tumors in an organism. Thus, being able to quickly represent a given organoid as a 3D model, ideally with automated tools, is crucial. Properties such as its shape, volume, surface area and branching behaviour can be determined from three-dimensional models quickly and with relative ease. A fast and accurate visual representation from all sides can ideally aid patient specific treatment options.

Advancements in recent years have opened the path for high-throughput microscopy, capturing multiple layers of 3-dimensional cell structures sequentially over time and in high resolution. Multi-well plates, populated with a high density of organoids and imaged over an extended time period, can provide large sets of data. This data can be used to feed machine-learning algorithms to accomplish tasks such as image segmentation. This assists the creation of a visual representation and thus the analysis of the organoids.

One of the recent advancements in the field of neural network-based image segmentation is U-net, a fully convolutional network [11] designed for biological image segmentation. It's capable of running on household computers, can quickly be trained with relatively few images and yields a precise segmentation. The objective here is to investigate its capacity in the realm of 3D cell cultures, in particular in conjunction with the 3D visualisation and analysis of growing PDAC (Pancreatic Ductal Adenocarcinoma) organoids.

2 Background

2.1 3D Cell Culture

Until recently, cell cultures have been understood as the two-dimensional growth of single cells or tissue in an artificial environment. It is also possible, however, to construct these research environments in all three dimensions.

2.1.1 Organoids

Organoids are three-dimensional cell structures, cultivated in vitro and embedded in a supporting matrix, such as a collagen-based one. Collagen is an important structural protein in the extracellular matrix in our bodies and supports cell division, differentiation and migration in the body [12]. Used in vitro, it enables the cell structures to propagate in a similar way to in vivo conditions, paving the way for a variety of use cases.

“Here we define an organoid as an in vitro 3D cellular cluster derived exclusively from primary tissue, embryonic stem cells, or induced pluripotent stem cells, capable of self-renewal and self-organization, and exhibiting similar organ functionality as the tissue of origin”

Fatehullah et al. (2016), [8]

Organoids prove especially useful in recreating tumour conditions in-vitro. While drug testing on genetically engineered mouse models, for example, has significantly improved our understanding of cellular mechanisms underlying tumorigenesis and aids the identification of cancer biomarkers, it is increasingly evident that rodents can not replicate the histological complexity and genetic heterogeneity of human tumors. The differences in human and animal biology, as well as the high cost and difficulty of imaging, especially in high-throughput studies, render animal models largely ineffective for targeted therapies [13, 14]. Also, methods such as patient-derived xenograft (PDX) models (see fig. 2.1), which implant human tumor tissue into mice for tumor-treatment testing, have similar deficiencies [15, 16, 17]. The tumor evolution and genetic composition of PDXs can deviate from the parental tumor substantially [18], they are expensive to develop and require rigorous ethics approval.

These shortcomings of animal models can be met by in vitro systems. They do not harm the organism from which they originate and are relatively cheap and reliable. The most prevalent method is to propagate cells from patient material to create cell monolayers in two dimensions. They are kept in a medium and cultivated so that a 2D cell line is created, where every cell is genetically identical. This homogeneity is not an

advantage, however, when it comes to drug research and testing, as tissue physiology research requires a differentiated phenotype profile [16]. Also, due to the lack of cell-to-cell and cell-to-matrix interactions, the culture is not able to mimic the cellular functions, signalling pathways, motility and growth patterns of in vivo tissue [13]. Organoid cell cultures pose a solution for these deficiencies.

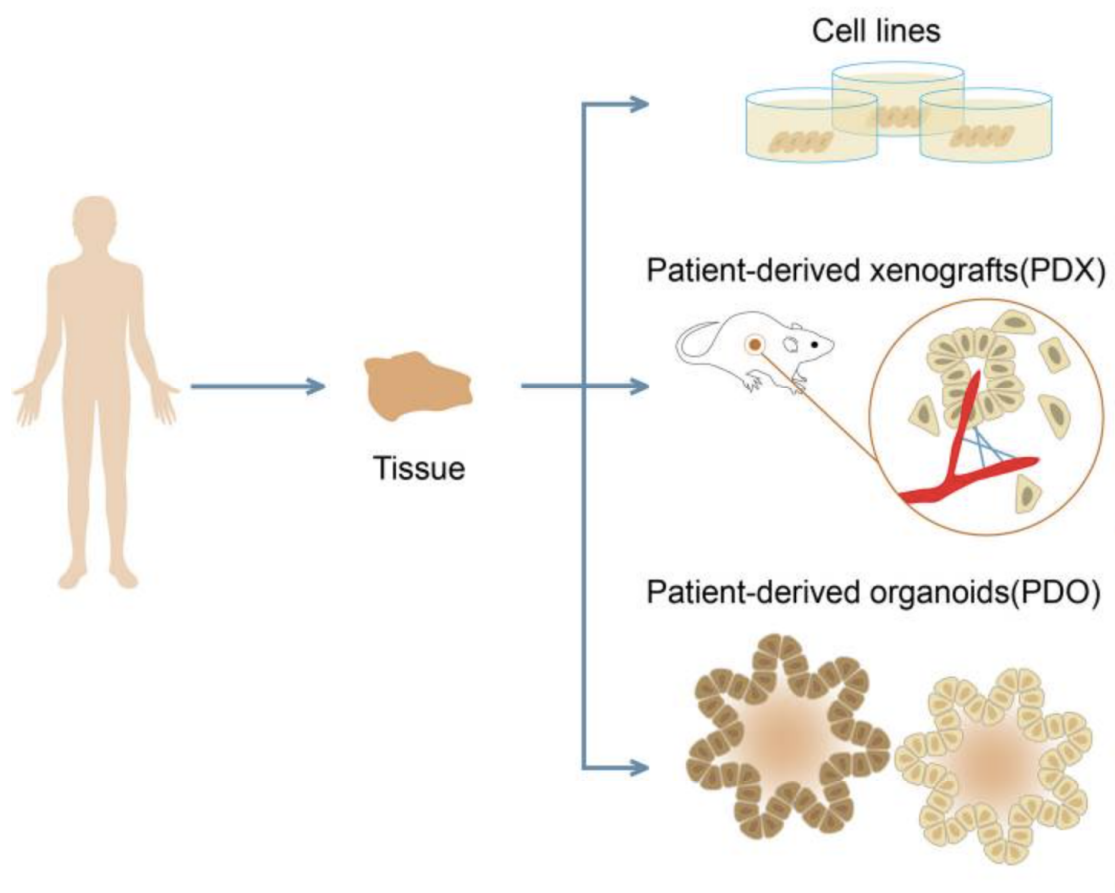


Figure 2.1: Comparison of cancer cell models, from [16]. Human tissue can be used to establish two-dimensional cell lines, patient-derived xenografts (implantation of human tissue in rodents) or organoids.

The three-dimensional structures can be expanded quickly and cryopreserved. They consist of cells with the necessary heterogeneity and genetic adaptability for patient specific drug-testing. They can be used in high-throughput experiments and to create biobanks [16]. In addition, they can be cultivated over a relatively long time period, retaining their patient- and disease-specific characteristics [19].

2 Background

Features	2D cell lines	PDX models	Organoids
Success rate of modeling	High	High	Low
Ease of maintenance	Easy	Medium	Hard
Expansion	Fast	Slow	Fast
Retention of heterogeneity	No	Yes	Yes
Genetic manipulation	Yes	No	Yes
High-throughput drug screens	Yes	No	Yes
Cost	Low	High	Low

Table 2.1: Overview of the advantages of the three preclinical cancer models, adapted from [20]. Especially note the fast expansion, retention of heterogeneity and high-throughput abilities of organoids.

Evidently, an in vitro system that so closely resembles in vivo conditions has a large array of uses beyond the scope of tumour research. Organoids can be used to model neurodevelopment disorders and then test drug properties. They prove useful for research of infectious diseases, like the Zika virus, noroviruses and parasites [21], and, because of their ability to be genetically modified (aided by technologies such as the gene scissors CRISPR/Cas9) and keep their phenotype, they are also being used to model diseases such as cystic fibrosis [21]. They offer an accurate environment to investigate drug toxicity, spreading behavior, and efficacy. This potentially reduces the need for animal testing or improves its results. Ultimately, the organoids can also be used to provide implant solutions to patients as a tissue resource and an assessment environment [22]. A comprehensive biobank can be established in vitro that documents a collection of cancer subtypes in a patient, for example, and provides a continuous supply of samples. Biobanks conserve the genomic landscape of the parent tumor and have proven useful to perform phenotype-genotype correlation analysis, functional tests and to assess drug response [21, 23].

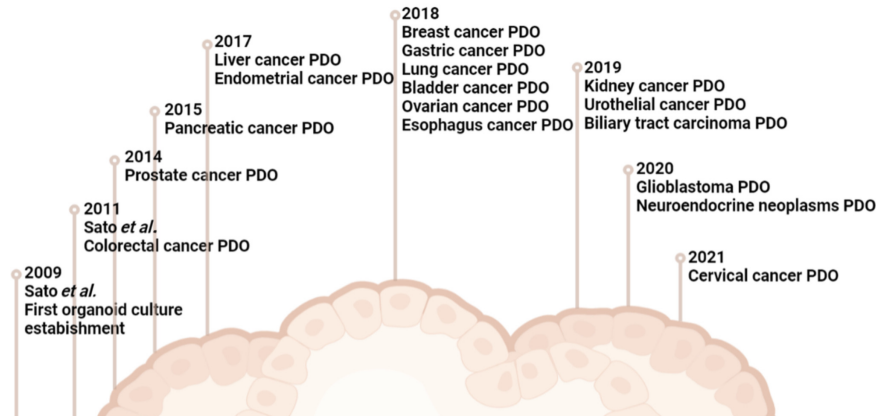


Figure 2.2: Recent developments in the field of patient-derived organoids (PDOs), with the year when the culture was first established, from [20]

The field of 3D cell cultures is a relatively recent one and a lot of progress has been made since the first organoid culture establishment in 2009. Different types of cancer have been successfully modeled according to the cell culture protocol from 2009 (see fig. 2.2). However, what is missing is a set of analysis tools to explore the full potential this research field has to offer.

2.2 Microscopy

The microscope as a method of optical analysis of biological structures is one of the most fundamental research instruments. A common type of microscopy in cell biology is widefield microscopy. This is an approach where the whole object is illuminated from the opposite side of the objective to create contrast. Usually, white light (bright-field microscopy) is absorbed by the cell structures and thus makes them visible in the viewfinder. In cell imaging, widefield microscopy stands in contrast to confocal microscopes, which are often favoured for three-dimensional material. They are well suited for this thicker material, because instead of a diffused bright light they use a focused laser beam. The laser beam scans over the specimen and targets a defined point in three-dimensional space with a high intensity beam. This approach offers a higher resolution, too. However, for high-throughput live-cell imaging, it is advantageous to resort to the wide-field variant because it damages the organism far less. In microscopy, there is always a trade-off between speed of imaging, resolution that can be attained, and sensitivity, where an increase in one leads to a decrease in the other areas. To successfully image 3D organisms in z-stacks, it is beneficial to image at the highest possible speed, and because we are working with live, growing cells, it is important to expose the specimen to as little high-intensity light as possible. Also, for a general morphological analysis, it is less important to be able to capture single details at a high resolution, so for the purpose of imaging live PDAC organoids we will be using wide-field microscopy.

2 Background

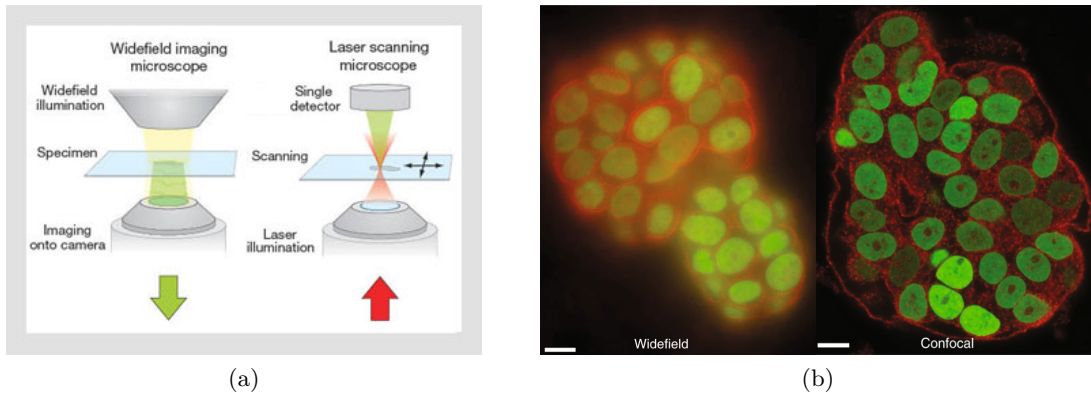


Figure 2.3: Differences in imaging of wide-field and confocal microscopes, from [24, 25]. Note the more defined beam on the confocal microscope which yields very high-resolution images but is disadvantageous when imaging quickly and over a long period of time.

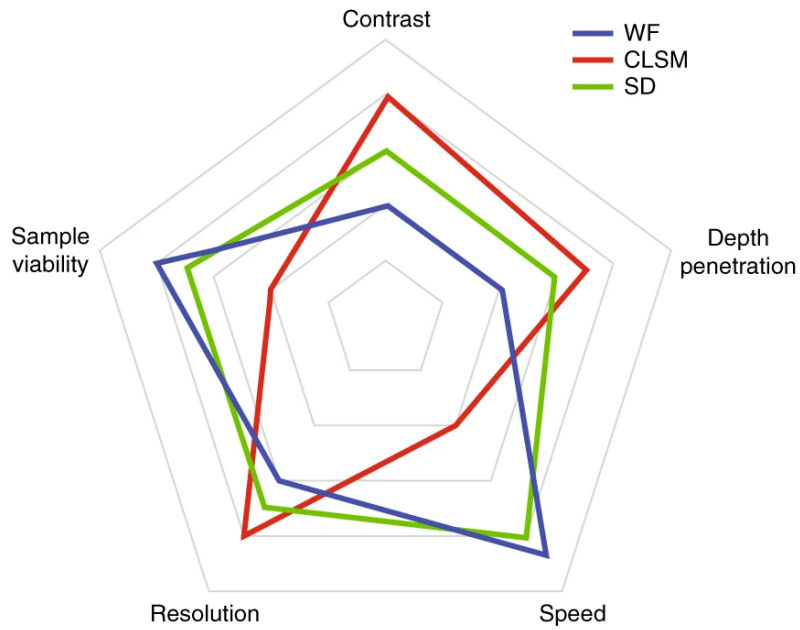


Figure 2.4: Advantages of each microscope type (widefield, confocal laser scanning microscope and spinning disk), from [25]. The strong points of widefield microscopy lie in its speed and sample viability (preserving the samples properties).

2.3 Machine learning in cell biology

In recent years, with the additional processing power and increasing affordability of powerful Graphics Processing Units (GPUs) and access to cloud-based computation, machine-learning techniques have become widespread in nearly every field of science. In particular, for complex tasks such as pattern recognition and various predictive purposes, they are used in a multitude of areas across the physical sciences. The field of machine learning is a branch of artificial intelligence (AI) and broadly describes the process of fitting predictive models to data through a learning process akin to the way humans recognize patterns [26].

A machine-learning based approach to image analysis has multiple advantages. Not only is there considerably more data for 3D cell structures than for 2D, but the need for speed in live cell cultures and thus high-throughput microscopy provides data sets of a much larger dimension. Also, as the temporal development of organoids is essential to understand the effects of drug treatment, for example, and the ability of time-series microscopy opens up a new world of analysis possibilities, there is also a time component in addition to the 3D data. Manual image analysis techniques are no longer sufficient to deal with these large and unhandy data sets, hence the convenience of machine-learning aided techniques in biology [27].

An algorithm can learn in different ways. While traditional machine learning techniques (regression, principal component analysis, gradient boosting) and most neural networks rely on already labeled data, there are algorithms which can also learn from unlabeled data sets. They can input unstructured data, such as text or image material, and automatically determine which set of features distinguish one class from another. However, in cases where the class is already known, for example in image segmentation, it is advantageous to work with labeled data, as it is faster and more accurate.

Neural networks are a subset of deep learning techniques that are modeled after neuron functionality in the brain. Nodes (artificial neurons) are connected in a similar way to synapses in biological neural networks, and structured into layers. Usually, there is an input and output layer, with multiple hidden layers in between that perform a transformation, as depicted in figure 3.5.

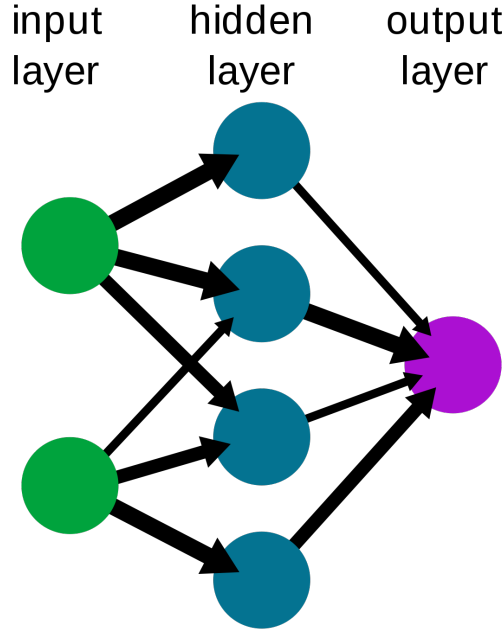


Figure 2.5: Layout of artificial neural networks, showing connected nodes in layers. The connections are depicted as weighted arrows. From [28].

Each node consists of input data, weights, a bias and an output, which leads to the input of the next node. The network can then be described as a linear combination (Z , see eqn. 2.1) of inputs and weights that computes predicted output values. Weights can be thought of as the "strength" of the connection, a measure of how much effect a certain input will have on an output. There might also be a threshold on a neuron, letting the signal through only if it passes a certain value.

$$Z = \sum_{i=1}^m w_i x_i + \text{bias} \quad (2.1)$$

The w_i describe the weights and x_i the inputs. Note the bias, which shifts the activation function by a constant to adjust it to real-life circumstances. The output values are then compared to the actual values (ground truth) during the training process and the loss is taken track of in the so-called error function. The goal is to minimize this loss, so the output of the neural network converges to the optimal solution. The network is then trained, and the attained values can be applied to a test data set [29].

Convolutional neural networks are one step forward from these basic (feedforward) neural networks, and are especially adept at computer vision tasks such as object recognition and image classification, albeit at a higher computational cost. A typical example is to distinguish cats and dogs. The special component of CNNs are convolutional layers, which consist of an input, a filter (kernel), and a feature map. The input data is a 3-dimensional data set with a determined height, width and RGB value, for example.

The filter acts as a feature detector and sweeps as a matrix (usually 3x3) over the image, combining the input pixels with the filter in a dot product and outputting the values in an array. The feature map is the final array once the filter has swept over the whole image. The whole process is called convolution [30].

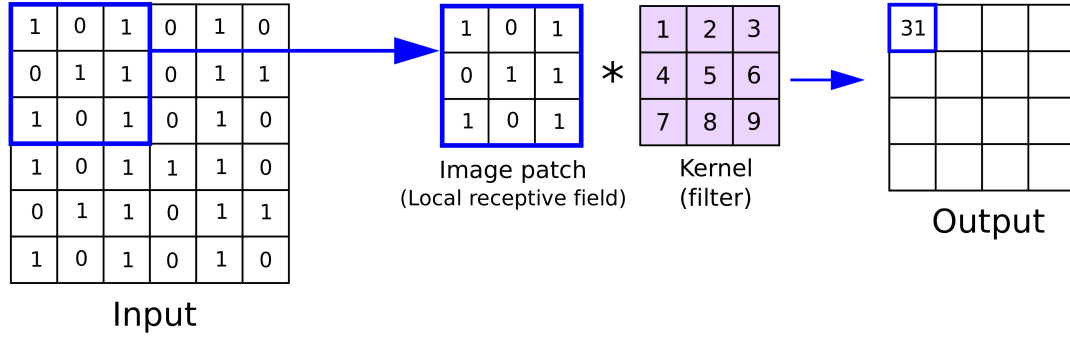


Figure 2.6: The principle of convolution, from [31]

Some important concepts for neural networks are:

- **Supervised and unsupervised learning:** The difference in learning lies in the labeling of the data. The training data in supervised learning is labeled by human hand, so that the algorithm iteratively learns and adjusts its output based on the ground truth. Unsupervised models discover structure in the data by themselves, and only need human intervention to validate the results. It is computationally costlier, and often inaccurate, but less time-consuming [32].
- **Overfitting and underfitting:** These are two common causes of model inaccuracy. While underfit models fail to recognise patterns in the data and show that the used model is too simple, overfit ones follow the data too closely. They both result in the model failing to predict outcomes accurately.

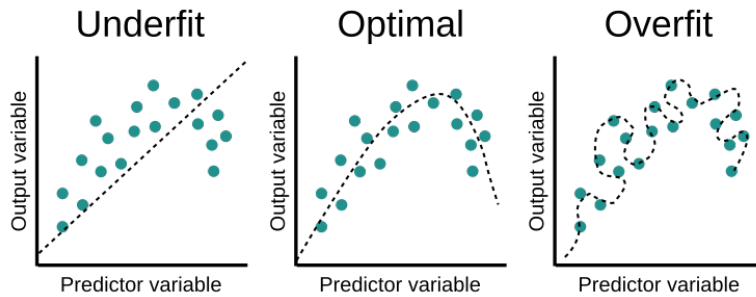
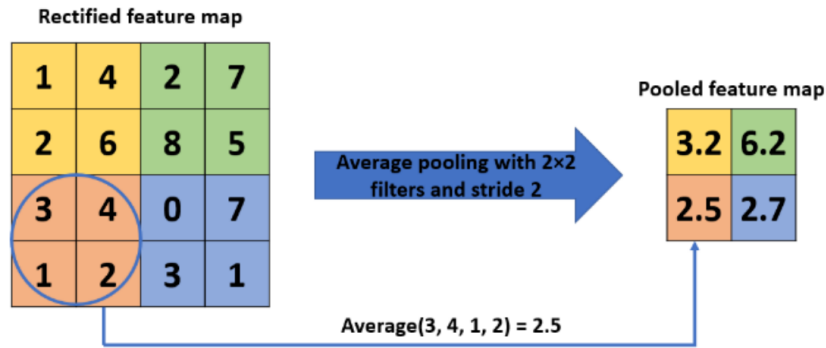


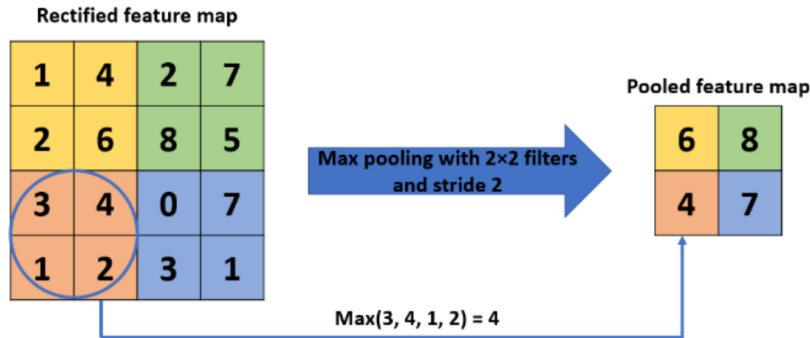
Figure 2.7: An example of underfitting and overfitting of a model, from [33]

2 Background

- **Activation function:** An activation function defines the output of a node and maps it to a value, for example between -1 and 1. Often it is in form of a logarithm or a binary step. [34]
- **Forward and backward propagation:** Forward propagation means moving from input to output through the network with assigned weights. Backward propagation is the process of running back through the neural network during the training process, and while doing this, computing the gradient of the loss function with respect to the weights of the network [35].
- **Pooling:** Pooling describes a downsampling method, that creates a new feature map in a condensed resolution, reducing the spacial dimension. On one hand, it diminishes the number of weights and also the computational cost. On the other hand it's a useful tool to control overfitting [36].



(a) Average pooling



(b) Maximum pooling

Figure 2.8: Example of pooling algorithms, from [36]

- **Dense layer:** Layers in a neural network are called dense when each node is connected to every node in the previous layer.

2 Background

- **Padding** allows the filter to sweep the edges of the input layer by adding extra pixels around the image [37].
- **Stride** describes the amount of movement of filter over the input layer, a stride of 1 would mean it does one more pixel at a time [37].

Typically, convolution networks are used for image classification, assigning a class label to the image. In biological image processing, it is often more useful to include localisation, where the class label is assigned to each pixel individually. To this end, we can introduce the so called "fully convolutional networks" (FCNs), often used for semantic segmentation. Predictions are made on a pixel-by-pixel basis and with supervised pre-training. They avoid using dense layers and only use locally connected layers (using solely convolution, pooling and upsampling), and thus fewer less parameters and can be used with variable image sizes. [38]

The following section outlines the specific microscopy and data analysis methods used for 3D cell visualisation purposes.

3 Methods

3.1 Cell biology

The cells for growing the organoids used in the experiment were from the 9591 PDAC cell line, by courtesy of Prof. Dr. med. Dieter Saur from the "Rechts der Isar" hospital in Munich. Pancreatic Ductal Adenocarcinoma (PDAC) is one of the most malignant forms of cancer. The 5-year survival rate is around 5%, it is of aggressive nature, hard to detect and very resistant to radio- and chemotherapy [39]. Our understanding of the disease progression and reaction to drug treatment is limited, making it an valuable playing field for organoid research [40].

The tissue was cultivated in Dulbecco's Modified Eagle's Medium/Nutrient Mixture F-12 Ham (DMEM) with supplements 10% FBS (Fetal Bovine Serum) and 1% P/S (Penicillin-Streptomycin). The medium was changed every 2-3 days when they were under 70 % to 80 % confluency. The number of cells per well was $2 \cdot 10^5$ at the start of the imaging process.

Growth phases

Current knowledge of organoid biology suggests that the growth of PDAC organoids can be divided into four distinct development phases, characterised by distinctive patterns of deformation, growth rate and cell migration (see fig. 3.1, from [9]).

First, in the "onset phase", the cells structure themselves along a main axis of elongation, proliferating at an exponential rate, distributing themselves homogeneously and reaching about 500 μm in five days. Overall, this stage can last until about day seven. Next, the "extension phase" is where growth along the branches happens (around day seven to nine), from the core to the tips, at a rate of about 195 $\mu\text{m}/\text{day}$. The invasion of the matrix occurs at a similar speed on all branches, and a branching event occurs on average once every two days. The third stage, the "thickening phase", happens after day nine. Here the branches stop elongating and start widening. The tips get blunter and thicken. Towards day 11-12, microlumens form through cell apoptosis and combine to create a single seamless lumen. We can be called the "lumen formation phase".

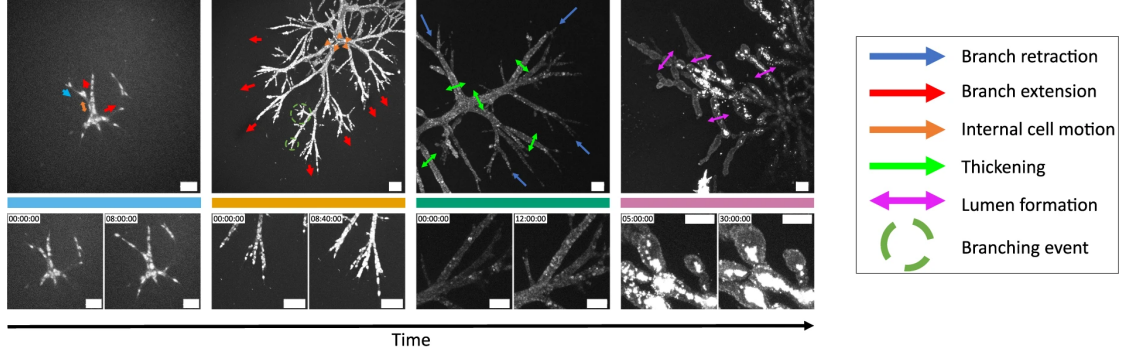


Figure 3.1: Development phases of PDAC organoids stained with SiRDNA, from [9]

For analysis purposes and to demonstrate the functionality of machine-learning aided segmentation during phases of high growth, the organoids chosen were in the growth stage from day five to eight, on the cusp of the extension phase.

3.2 Microscopy of 3D organoid cell cultures

In order to properly analyze the organoids, it is fundamental to firstly grasp the underlying principles of 3-D time-series microscopy. The biological structures being imaged are in the order of up to millimeter, so they lie well within the resolving range of light microscopes.

The one that was used to image the data sets is the [LEICA Thunder Imager Live Cell & 3D Assay](#) inverted bright-field microscope on a DMi8 stand and a motorised stage. The objective was the N Plan 5x/0.12 PHO, with a numerical aperture of 0.12. The cells and the medium (see section 3.1) were kept in the two-well coverslip [Ibidi \$\mu\$ -Slide 2 Well Ph2](#) in the dimension 21.6 x 23.8 x 9.3 mm³. This was connected to an incubator, the [Tokai Hit STX Stage Top Incubator](#), which kept the cell culture at a constant $T = 37.0^{\circ}\text{C}$, 5%CO₂ and 100% humidity.

3D cell structures can be especially tedious to image, especially when the temporal development is also important (4D microscopy). The camera images by scanning over the two wells, taking 2048x2048px images of selected squares. It scans over all positions once per hour, taking 101 single two-dimensional images in the z-plane per position, each 0.5 μm apart. These layers of 2D images, which when combined show the entire three-dimensional plane of a certain position, are called z-stacks.

Due to the growth of the organoids and also the viscosity of the medium, the z-axis alignment of a point can change over time. It is vital that the microscopy system is not submitted to external mechanical influences either. The cells were imaged for 8 days, during which the organoids drifted independently in the xy-plane, but not more than around 40 μm a day. A drift in z-direction could not be determined. The drift can be resolved in the pre-processing process at a later stage.

3 Methods

Because the lighting conditions across a whole position aren't homogeneous, the gradient is sometimes quite noticeable. Although the 3D cell culture images don't exhibit the same discontinuities as stitched 2D images, organoids that are on the overlap region can exhibit shading inconsistencies which complicates processing. It is therefore paramount to leave as much space around the organoid as possible while taking its growth into account.

The main data set used for further analysis was imaged on the July 11, 2022 and shows the development of the PDAC organoids from day five to eight. It can be found in appendix A. The organoids are named according to grid and imaging position on the plates. The ones that were used in the analysis are given in fig. 3.2.

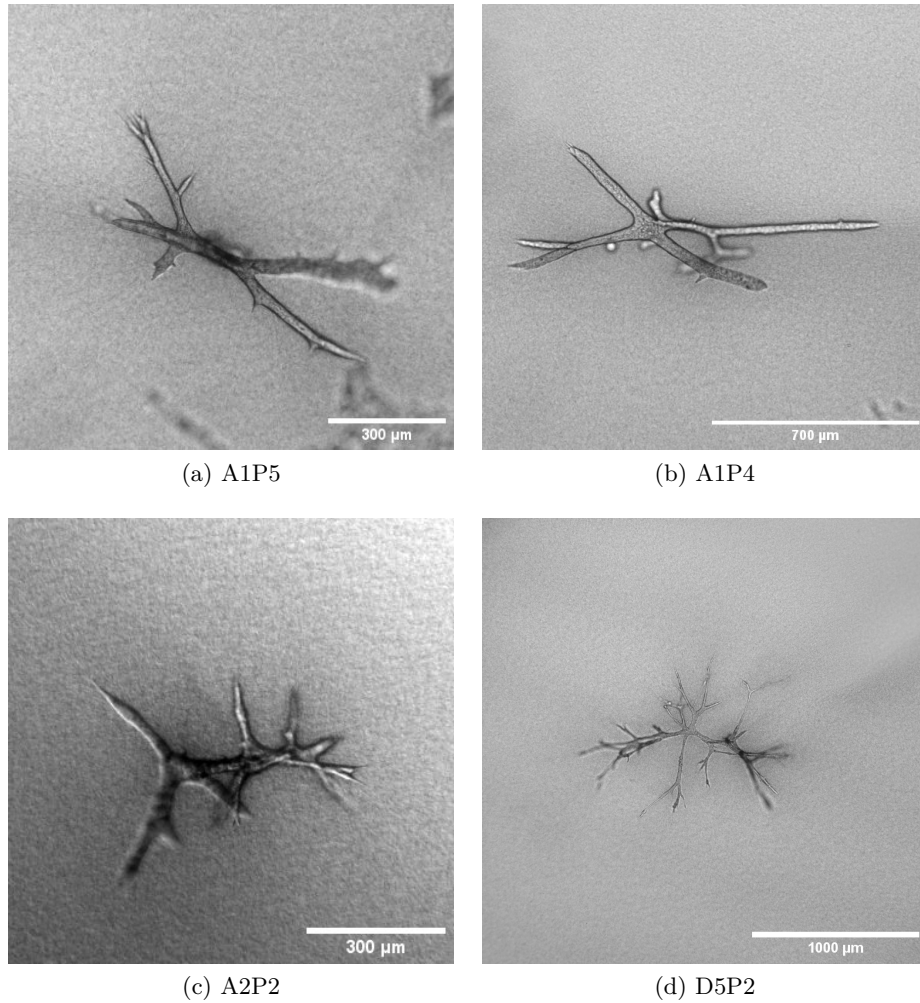


Figure 3.2: The four organoids that were used for further analysis.

3.3 Data processing and analysis

In order to obtain a 3D model, the data set has to go through a number of steps (fig. 3.3). First, the imaged 2D stack in all its time points goes through a procedure of pre-processing, where it is prepared for the segmentation, either manually or automatically with U-net. The segmented images can then be used to create a 3D model which is the foundation of further analysis, or can be used to visualise the organoids as a 3D print. Through voxelization the model can be used to augment the data for further training processes.

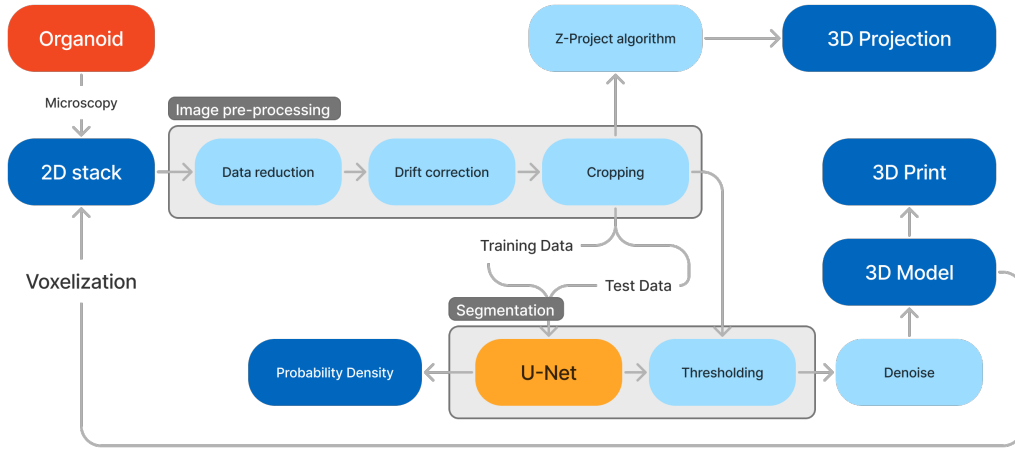


Figure 3.3: Image processing, from the stack of 2D images, over the pre-processing and segmentation to the three-dimensional model.

3.3.1 Pre-processing

To prepare the data for any further processing and analysis, the data set has to be firstly reduced and organised. The raw LIF files from the microscope of the entire well are > 100 GB and thus not useful for processing. Once the single organoids are chosen, cropped, and converted to the TIF format, they can be opened in a scientific image processing program such as ImageJ.

To correct the alignment of the images stack, the plugin Stackreg [41] was used. In most cases, the organoid is only affected by translation and a coordinate transformation of $\vec{x}_1 = \vec{x}_0 + \Delta\vec{x}$. The Stackreg "Translation" algorithm is sufficient in these cases. If a rotation $\vec{x}_1 = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \cdot \vec{x}_0 + \Delta\vec{x}$ is involved, the "Rigid Body" algorithm does the job. The other algorithms add more degrees of freedom and normalise the size, but in this case the aim is to keep the original size for later analysis. For further processing, it is necessary that the images are then cropped and the black bars on the edges removed.

Z-projection

The three-dimensionality can be analysed to a certain degree directly from these pre-processed stacks if they are flattened as a projection. Every pixel on a point in the x-y-plane is combined with all others along its z-axis to create a two-dimensional image. This can be done with a variety of methods, each projecting the pixels differently [42]. While this has limited functionality for the analysis of 3D cell cultures and doesn't offer significantly more information than single 2D images (see fig. 3.4), these methods are being used in biomedical imaging (MRI scans, single cell organelle analysis) extensively.

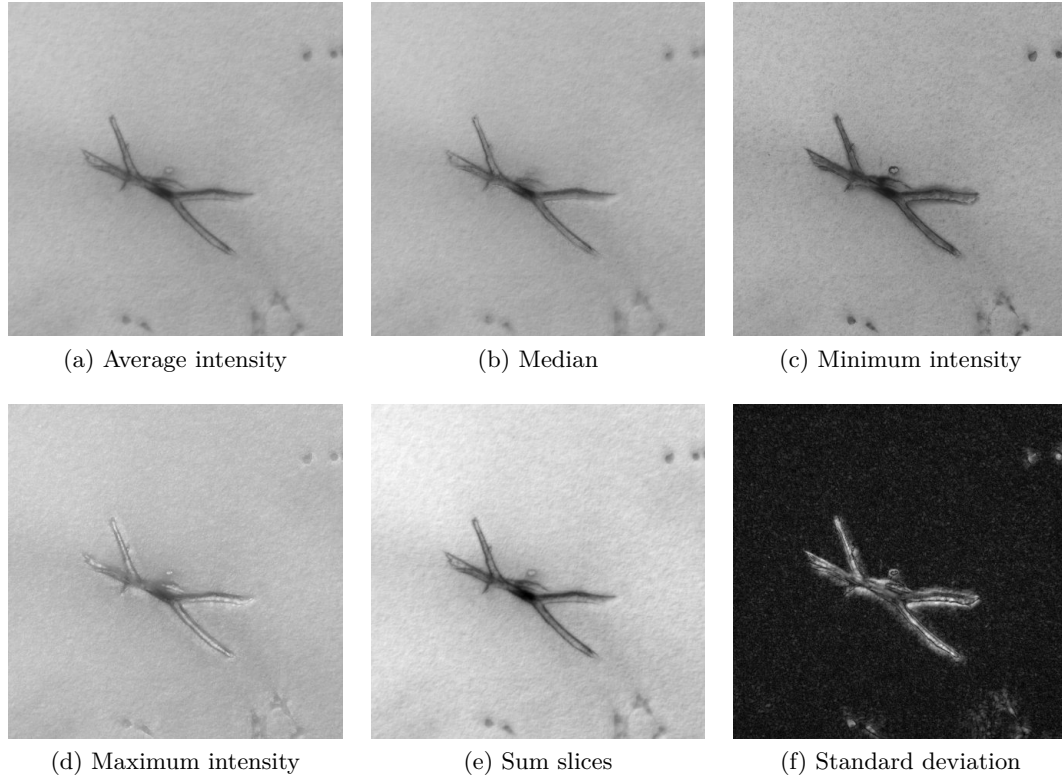


Figure 3.4: The ImageJ Z-Project function applied to organoid A1P5. Each algorithm shows slightly different details, but don't offer a major advantage over the basic 2D image.

3.3.2 Image segmentation in 3D

Image segmentation describes the process in which an image is divided into semantic sub-classes and regions of interest. In self-driving cars, this can occur through categorising the camera output into classes such as the road, street signs, pedestrians, other cars and so on. In biology, it's often important to distinguish cell components, for example. In our case, it is about removing 3-dimensional cell structures from the background.

3 Methods

This is essential if we would like to represent the organoid as a 3D model for further analysis. We do this, by taking the single 2D images, segmenting them individually, and then stacking them back together. This is a substantial amount of images, for one time point up to 100 images, and if we are looking at growth patterns over 30h, the amount is increased to 3000 images. This is too much data to process manually which is why automated techniques come in handy.

In order to automate this process with machine-learning methods, we must understand the classic method first. It is usually done through some type of thresholding. For 2D cell cultures, Edge Detection algorithms help too, but for Z-stacks the edge is less defined and this method is prone to error. ImageJ does segmentation quite effectively already through its auto-thresholding algorithms, and the performance is enhanced if the input image has a high contrast. If the original contrast isn't sufficient, it's possible to manually increase the contrast directly in ImageJ.

Figure 3.6 shows the auto-thresholding methods in ImageJ. A lot of algorithms are very similar, but the best ones for further processing are those with the least noise and the clearest defined lines. The Yen algorithm [43] (bottom picture), Shanbhag [44] and minimum algorithms [45] (on the right side) are especially useful for organoid segmentation.

A disadvantage of this approach is that ImageJ loads all the thresholded images into the RAM, and as a single image can be up to 20 MB, personal computers can run out of space very quickly for large data sets. Also, some additional processing is needed to ensure that only the organoid itself is captured and no artifacts. Through ImageJ's Remove Outliers algorithm, it is possible to specify the size of artifacts not adjacent to the organoid and remove them. This can be done to the whole stack, but is prone to error and will almost always destroy some part of the organoid outline in the process.

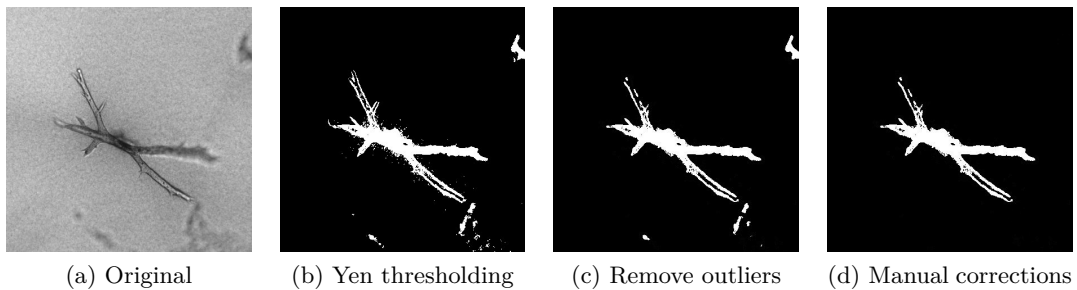


Figure 3.5: Semi-automatic segmentation of a frame of A1P5 using ImageJ

3 Methods

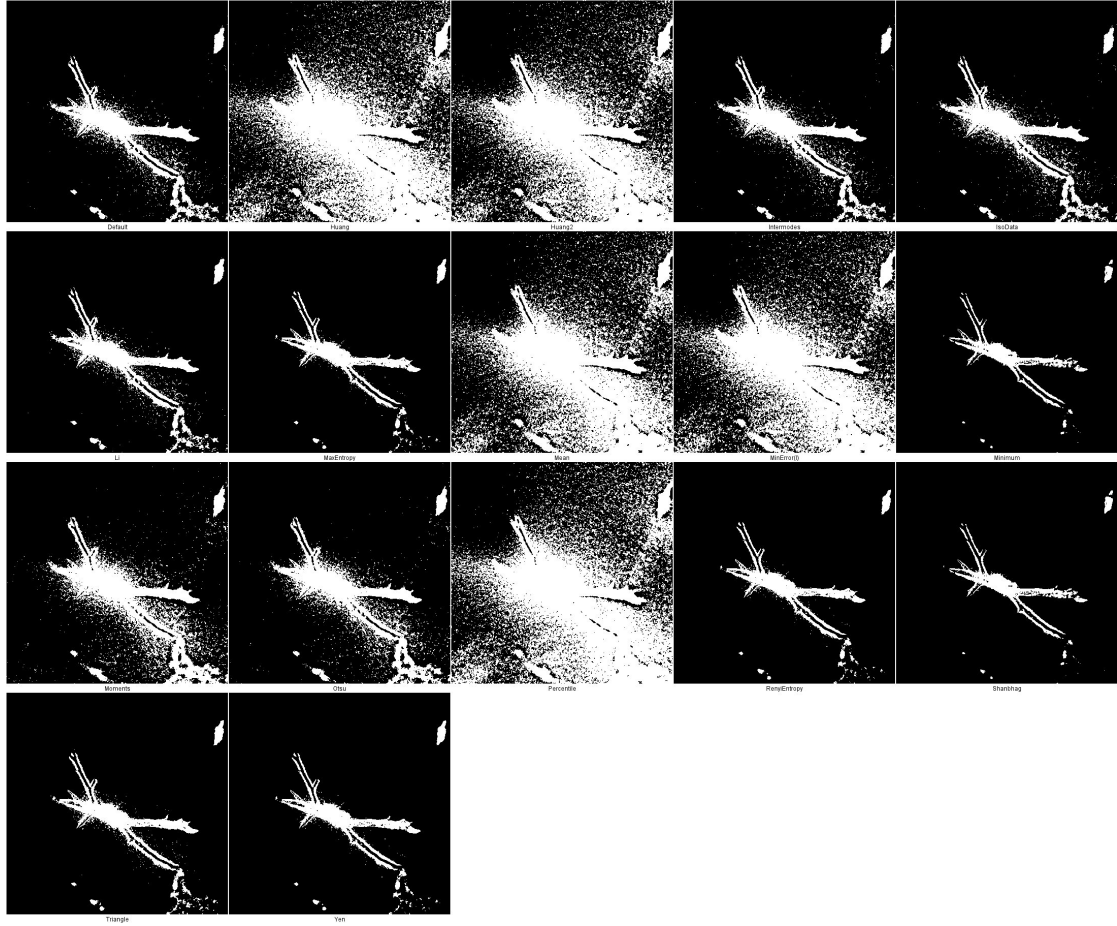


Figure 3.6: Comparison of the auto-thresholding methods applied to A1P5

Manual corrections can be done with the paint brush to remove obvious outliers. However, the process is time consuming and counterproductive to the high-throughput principle. This is where the automated approach using machine-learning comes in.

3.3.3 U-net: CNN-aided image segmentation

U-net is a convolutional neural network proposed and developed by Olaf Ronneberger, Philipp Fischer and Thomas Brox at the Computer Science Department at the University of Freiburg, Germany [11]. Its purpose is to offer a segmentation algorithm for biological material that is precise, relatively lightweight to run and utilises less training data than usual.

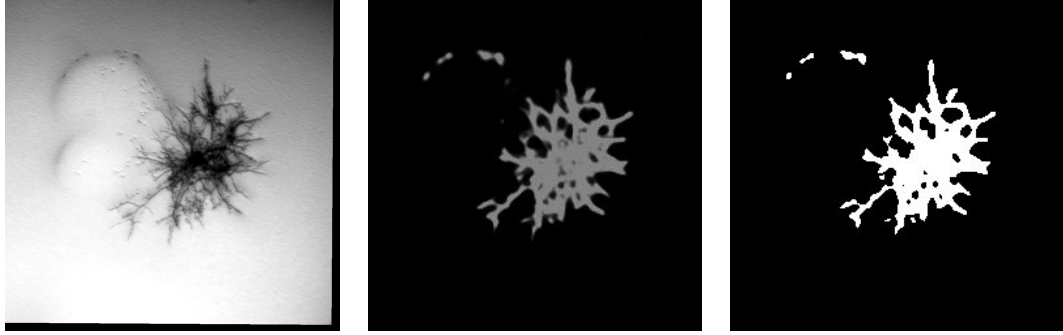


Figure 3.7: An example of a 2D image segmentation using U-net on a 2D image. The input image (left) is segmented by the CNN (middle) and then a mask can easily be created. The organoid is successfully binarily distinguished from its background.

Functionally it is based on the model of the "fully convolutional network". It can separate touching objects of the same class (traditionally a tricky issue in neural networks) by utilising a weighted loss function where the background labels between touching cells are worth a greater loss. To mitigate the common issue of limited data availability in biology, U-net applies elastic deformations to the training data. This is an effective method of data augmentation and makes the network invariant to such deformations, which is vital in biomedical imaging. Even slight changes in tissue deformation can potentially have entirely different causes and effects.



Figure 3.8: To utilise U-net to create a 3D model, it is necessary to split each organoid into single 2D files, crop and ideally drift-correct. These single slices can then be fed to the python script (see the entire code in detail in the appendix B).

3 Methods

The network is made up of a contracting path (left side of the U; see 3.9) and an expansive path (right side). The contracting path applies two 3x3 convolutions (un-padded), each with a ReLU (see fig. 3.10) activation function, to the input image. It is then down-sampled by a 2x2 max pooling operation with stride 2, and these steps are repeated, doubling the number of feature channels at each step.

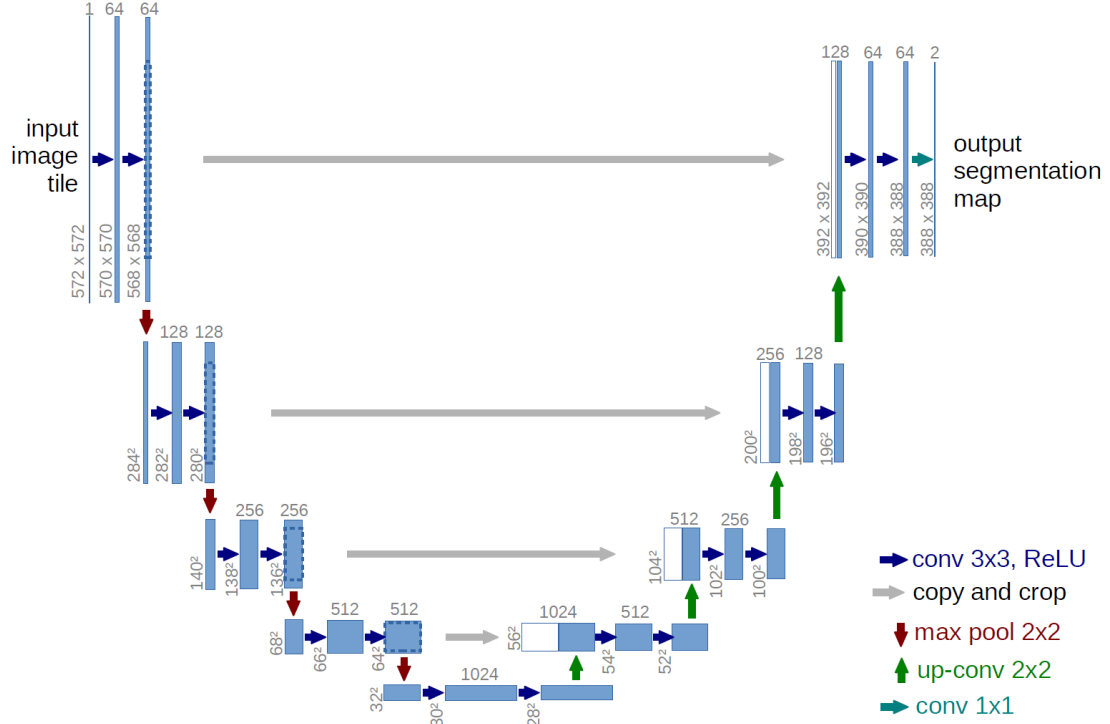


Figure 3.9: The architecture of U-net, from [11]. The left side shows the contracting side, a series of downwards convolutions and max pooling. The rightside essentially retraces the steps through upwards convolution and creates a segmentation map as the output.

On the expansive side, the network takes these same steps again, but in the opposite direction, as shown in fig. 3.9. The upwards convolution halves the number of feature channels and is concatenated with the cropped map from the contracting path (grey arrow). The cropping is essential because there is no padding used, so border pixels get lost with every convolution (note the dimensions written vertically in grey). The last layer has a filter of size 1x1 to map each 64-component feature vector to the wanted number of classes.

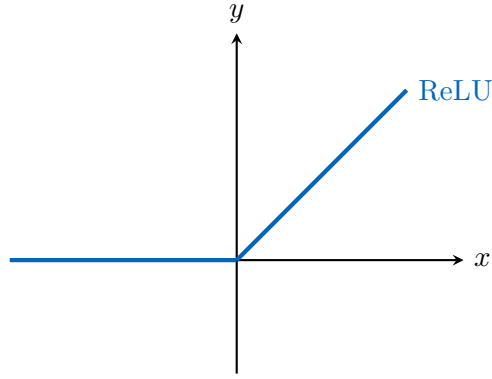


Figure 3.10: The ReLU activation function, also known as the rectifier, is defined as the positive part of its argument: $f(x) = \max(0, x)$. It is a computationally simple function to handle, is scale invariant and offers a better gradient propagation than other functions [46, 47].

The training process is done using ideally large images to make do for the unpadded convolutions. The output image will be of reduced dimensions and gets heavily scaled down from original 2048x2048 images on the input to 256x256 on the output. The dimension of the training images is even more important than the batch size. For further processing, the size of the output image isn't an issue because the outline of the organoid is already clearly defined. A high stochastic gradient descent (momentum of 0.99) is used so that the update in the current optimization step is determined by a large number of previously viewed training samples.

For the purpose of this thesis, a data set with 116 two-dimensional images was used. Through data augmentation in pre-training they were rotated clockwise, anticlockwise and flipped horizontally and vertically to provide a data set of 580 images. Every image was segmented manually to provide an accurate ground truth label. Together with 300 labeled images from a previous training run, the data set consisted of 880 individual images.

The U-net code (see B) was adopted from [48] and was executed in Python. The script utilises the Keras API on top of the Tensorflow 2 library which simplifies processes such as pooling, activation functions and layers.

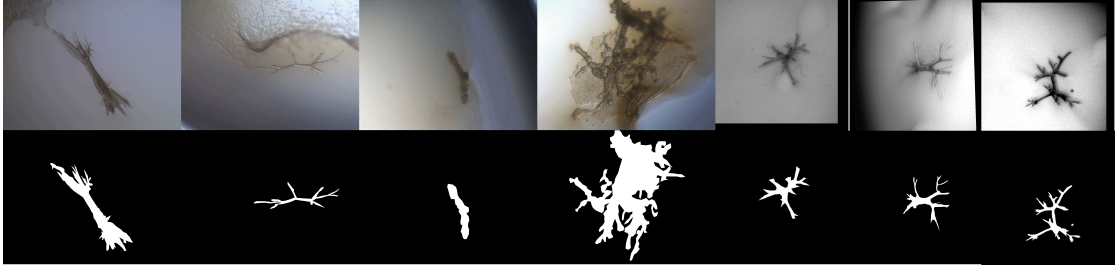


Figure 3.11: Examples of types of organoid structures used for training with their respective manual segmentation below

To calculate the energy function needed to incrementally minimize the error, U-net first uses a soft-max function, eqn. 3.1, pixel-by-pixel. It approximately outputs $p_k(\mathbf{x}) \approx 1$ when the feature channel k has the maximum activation $a_k(\mathbf{x})$ and $p_k(\mathbf{x}) \approx 0$ for all other k . The variable \mathbf{x} denotes the pixel position on $\Omega \subset \mathbb{Z}^2$.

$$p_k(\mathbf{x}) = \frac{\exp(a_k(\mathbf{x}))}{\sum_{k'=1}^K \exp(a_{k'}(\mathbf{x}))} \quad (3.1)$$

Then, at each position \mathbf{x} , the cross energy loss function (eqn. 3.2) penalizes the deviation of $p_{\ell(\mathbf{x})}$ from 1.

$$E = \sum_{\mathbf{x} \in \Omega} w(\mathbf{x}) \log(p_{\ell(\mathbf{x})}(\mathbf{x})) \quad (3.2)$$

where $\ell : \Omega \rightarrow \{1, \dots, K\}$ is the true label of each pixel and $w : \Omega \rightarrow \mathbb{R}$ is the weight map (eqn. 3.3). d_1 and d_2 denote the distance to the nearest and second nearest boundary pixels, w_c is the weight map to balance class frequencies and w_0 and σ are constants.

$$w(\mathbf{x}) = w_c(\mathbf{x}) + w_0 \cdot \exp\left(-\frac{(d_1(\mathbf{x}) + d_2(\mathbf{x}))^2}{2\sigma^2}\right) \quad (3.3)$$

There is a higher weight at the border of the segmented objects, so these have a greater effect on the loss. This leads to a discontinuous, binary segmentation of objects, making U-net particularly suited to biomedical image segmentation. Every pixel is placed in a category which turns a classical segmentation issue into a multi-class classification one. This is a lot more accurate [11]. There is a version of U-net that works directly in 3D [49], but it is tailored toward single-cell structures.

To evaluate the performance, the training process was run twice on different settings. On high settings (training T2), the U-net training process ran with a batch size (hyper-parameter that determines the number of used images per training iteration) of 8, 1000 epochs (number of times the network went through the entire training data set) and 125 steps per epoch. Training T1 was done on much lower settings, with a batch size of 4 and 8 steps per epoch.

Segmentation evaluation metrics

To analyse the accuracy of the resulting masks, we can use a variety of statistical measures. The Jaccard index (also known as "Intersection over Union") is given by

$$J(A, B) = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \frac{|A \cap B|}{|A \cup B|} \quad (3.4)$$

where A and B denote the masks of the manual and automatic segmentation. The precision, recall and f-score are statistical binary classification methods and are given by following equations. They were calculated by a point map on the segmentation masks.

$$r = \frac{\text{true positive}}{\text{true positive} + \text{false negative}} \quad (3.5)$$

$$p = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \quad (3.6)$$

$$f_B = (1 + B^2) \cdot \frac{p \cdot r}{p + r} \quad (3.7)$$

The f-score (Dice coefficient) is a the harmonic mean between precision and recall, and the B gives the balance. The recall is B times as important as the precision in a f_B -score. In table 4.1 they are both given the same weight.

In contrast to the other metrics, the structural similarity index (SSIM) is not pixel-based. It works by evaluating the luminence, contrast and structure on both images and comparing those. This is a somewhat more substantial measure for intricate branch structures [50].

3.3.4 3D representation

Once we have the segmented images from U-net, we can stack them back all together using the "Images to Stack" utility in ImageJ. By setting the voxel depth in to $0.5 \mu\text{m}$ as given by the microscope reading and opening the segmented stack in the 3D viewer, we can already visualize what the cell structure looks like. However, the manipulation and analysis options are very rudimentary, and so to perform further analysis we will export the 3D object as an STL file and import it into Blender.

The goal is to be able to visualize the organoid from all sides, ideally with very few steps to ensure a quick analysis that keeps up with the high-throughput imaging and retains specificity. We can import the STL for each timeframe and assign them to their respective timeline position by going into the object properties and under "visibility" adding a driver to the viewport with the expression `frame != float(self.name)`. When applied to all objects, every model is assigned to the corresponding frame. This enables a visualisation of growth in the timeline.

If the models look rough, they can be partially smoothed by enabling the "Auto Smooth" option on the object. For rough obvious artifacts, the smoothing brush can be

used to manually edit them out. This is non-invasive and doesn't change parameters such as the surface area or volume. If the segmentation is good enough though, there shouldn't be the need for a lot of smoothing. Holes, if any, can be filled by a simple interpolation of mesh points.

The scaling in blender is by default by a factor of 10^3 , so this was taken into account when determining metrics such as the surface area and volume of the object. They can be read from the 3D print menu.

3.3.5 Hard- and software

Below is a list of the software used, including the version.

- [Leica Microscope Software Platform LAS X Life Science 3.7.6](#)
- [ImageJ 1.53t](#)
- [Blender 3.3.1.0](#), the free and open source 3D creation suite
- Python 3.9.0, running in Visual Studio Code 1.73.1 on Ubuntu 20.04.5 LTS with an Intel Core™ i9-10900X CPU @ 3.70GHz x20, 64GB DDR4 RAM and two NVIDIA GeForce RTX 2080Ti RevA graphics cards
- U-net, adapted from [\[48\]](#), utilising Tensorflow and Keras libraries

4 Results

4.1 Image segmentation performance in 2D

Training T2 (higher settings: batch size of 8, 125 steps per epoch) ran for roughly 120 h. Each epoch took on average 7.2 min with each improving the loss to a final 0.02247. In comparison, when running the training process T1 with a batch size of 4 and 8 steps per epoch (which took 26 min on the same system), the loss was 0.06971. Applying the trained network to 20 images took only roughly 20 s, a second per image. Due to hardware limitations, it wasn't possible to process the whole z-stack at once, so this was done in batches of 20.

Figure 4.1 shows the output for a part of organoid A1P5. Because of the large data set, training T1 was used for this round. Even though parts of the organoid are out of focus and partially covered by artifacts in the medium, the neural network managed to segment successfully, also getting details right. It shows that even with low training settings, a sufficient segmentation can be achieved. With a simple batch thresholding algorithm applied, the images were then used directly to create a 3D model (see fig. 4.3).

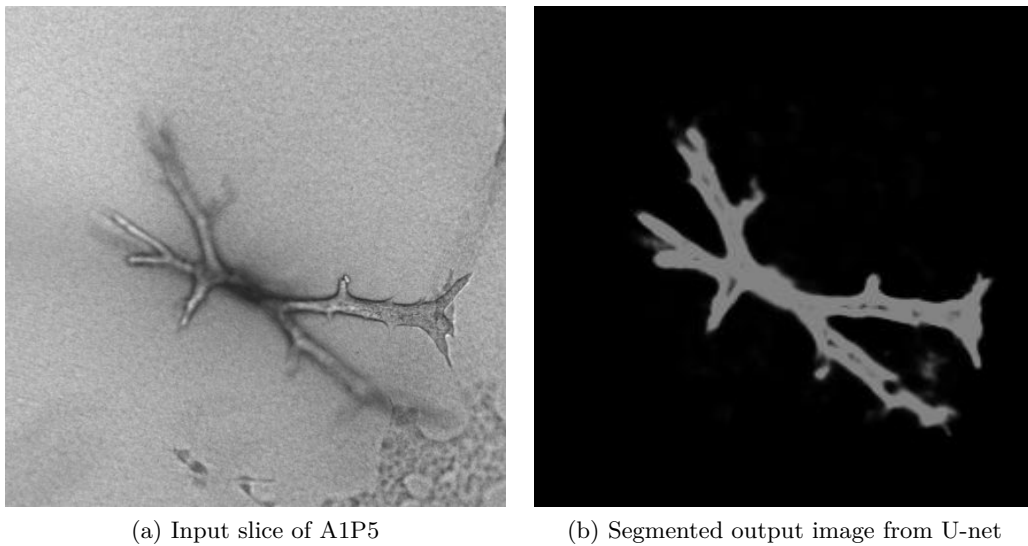


Figure 4.1: Organoid A1P5, segmented with training run T1 (lower settings). The outlines are well defined but the contrast is not as pronounced as with T2 (see below).

4 Results

To gain a better overview over the achievable accuracy with the U-net network, different morphologies, detailing and imaging conditions were tested and compared to manual segmentation on a variety of metrics, this time run on the higher settings, T2. Some examples are depicted in fig. 4.2 and the results are given in table 4.1. On these images we can clearly see a shading gradient in the output which can be understood as probability density. It gives the relative likelihood of a part of the organoid being at a certain position. Less probable features (for example due to artifacts or lighting) are shaded in a darker tone. There is also a surprising amount of detail on overlapping branches. The three-dimensionality can be seen. This is especially clear on the T2 images which show that for direct analysis of two-dimensional images a more thorough training process is beneficial.

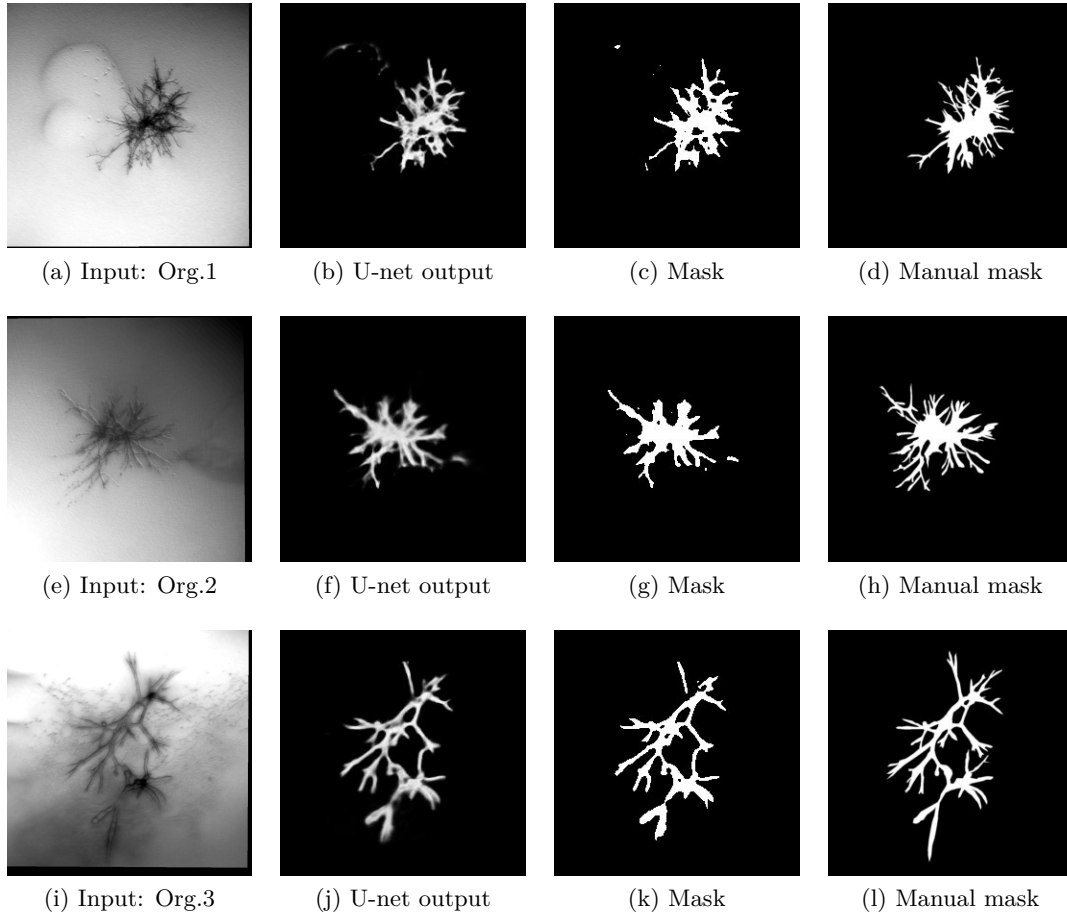


Figure 4.2: Some more complex organoids were segmented with U-net to test its capabilities, with the high settings of training T2. The second column shows the output from U-net which is then masked (third column). For comparison, a manual masking was done, as seen on the right. Running U-net takes about 1 s per image, while the manual masking at this level of detail took around 12 min per image in ImageJ. The statistical comparison can be found in table 4.1.

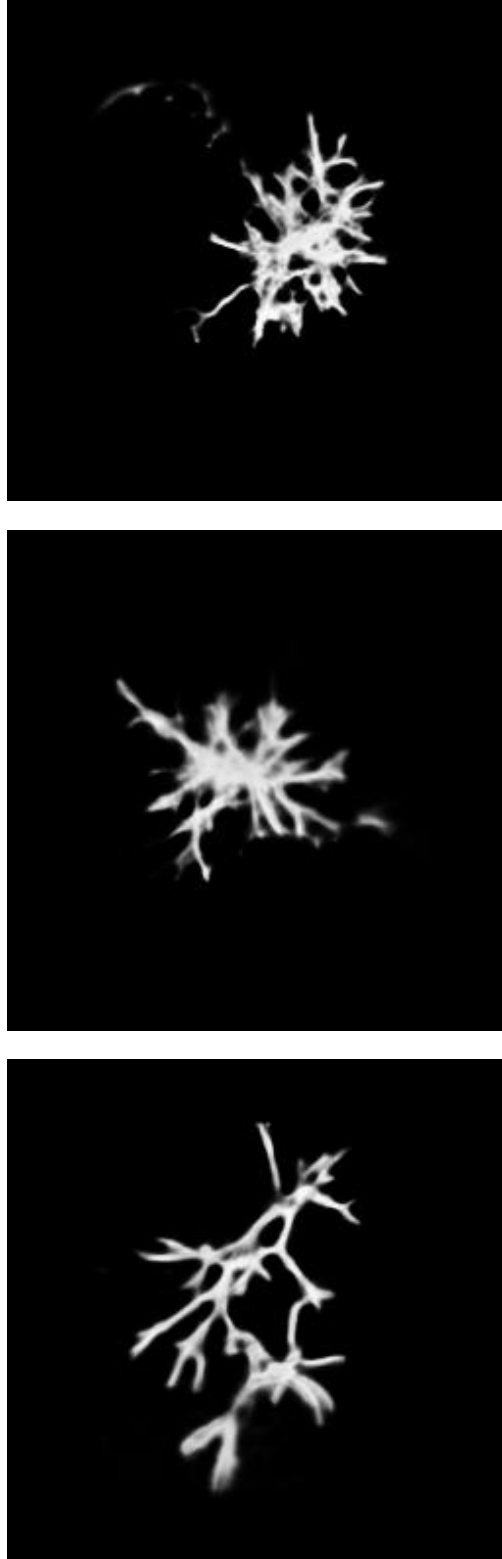


Figure 4.3: Figure 4.2 (b, f, j) enlarged to show the gradient across the branches which indicates the probability density.

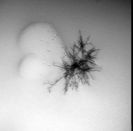
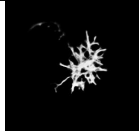


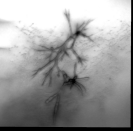

Organoid	Image	U-net	J T	J T+RO	p	r	f_1	SSIM
1			0.579	0.762	0.923	0.814	0.865	0.914
2			0.596	0.81	0.960	0.832	0.892	0.903
3			0.626	0.817	0.912	0.887	0.899	0.899
Avg.			0.572	0.779	0.942	0.820	0.875	0.905

Table 4.1: U-Net quality metrics. J is the Jaccard index (Intersection over Union, IoU), with T being the comparison with the thresholded U-net segmentation and RO with the additional Remove Outliers algorithm. p , r , and f_1 give the precision, recall and f_1 score respectively. Metrics from [51]

As is evident from the side-by-side comparison in fig. 4.2, fine details, especially the tips, aren't captured as accurately as in the manual segmentation. However, the high average values of $J = 0.78$, $p = 0.94$, $r = 0.82$ and $f_1 = 0.88$ demonstrate a significant overlap and general pixel-wise matching of features, considering the complexity of the input images. The SSIM-score of 0.91 shows a high match of luminance, contrast and structure. Also, the fact that the manual segmentation at this level of detail takes 12 min longer demonstrates the practicality and versatility of U-net. With a higher resolution, a greater batch size and more steps per epoch in training, an even higher match can be achieved.

4.2 Visualisation of 3D model

The three-dimensional model on the other hand allows for a comprehensive analysis. The model in fig. 4.4 gives a picture of the dimensions, positioning and directional growth of the branches of organoid A1P5. The organoid is roughly symmetrical, with two main branches on either side along the main axis, each with a similar diameter ($10\text{ }\mu\text{m}$), and a similar angle between them (51° and 56°). It grows mainly in one plane, but the front branch spouted from the lower half of the structure, and also the two branches at the far end don't emerge from the planar equator of the organoid. The branches are overall cylindrical with a diameter of around $50\text{ }\mu\text{m}$ to $60\text{ }\mu\text{m}$, but also relatively pointy at the tips. This indicates the points where the branches are elongated during the extension phase.

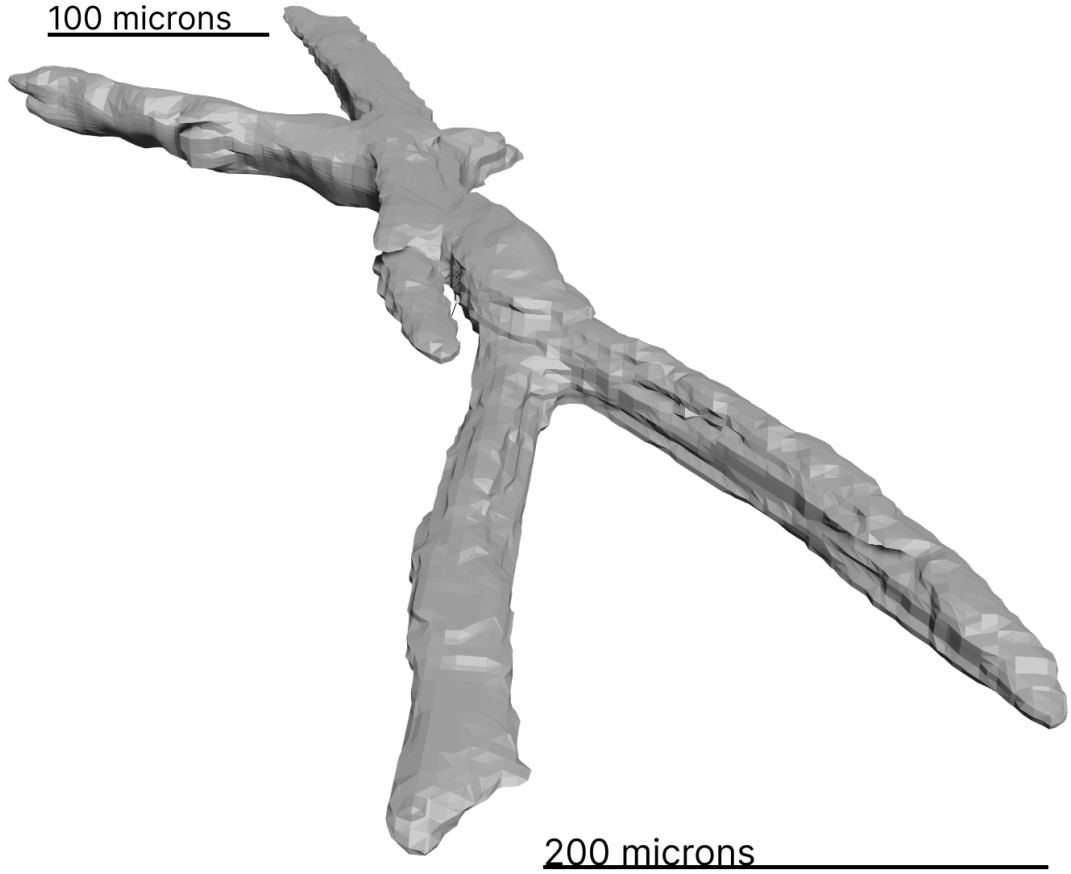


Figure 4.4: 3D representation of organoid A1P5 at the $t = 0$. Note the branching along the main axis of extension.

While A1P5 is a specimen that grows primarily along its longitudinal axis and shows very few branching events, D5P2 (fig. 4.7) is a lot more finely divided. The branches are thinner (about $30\mu\text{m}$ across on average), and instead of growing longitudinally, it focuses on sprouting in several directions, preferring a higher number of branches over thickness. While a main axis of elongation can be recognised, it isn't as pronounced as with A1P5.

An interesting aspect are the bulges at the inception points of the branches in D5P2 (fig. 4.7). They appear even before the branches start emerging, indicating a build up of cells to prepare for the elongation (points c , d and e). At the end of the main axis, the point a is a center point for branching in seven directions, while point b does similarly on the other side of the main axis of elongation (blue line).

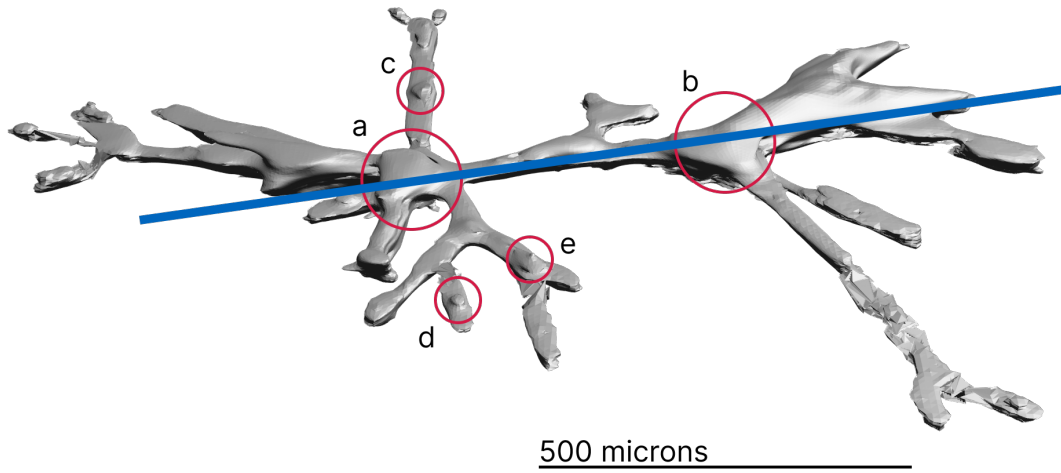


Figure 4.5: 3D representation of organoid D5P2 at $t = 0$, note how it is morphologically different to A1P5. It shows much more branching and doesn't only grow longitudinally but aims to cover a large area. The blue line shows the main axis and the red circles show bulges that form at division points. They indicate a build-up of cells in the area, preparing for protrusions in the spot. Points a and b are the main sprouting points at both ends of the main axis, while c , d and e are bulges that have formed before the beginning of branch growth.

4 Results

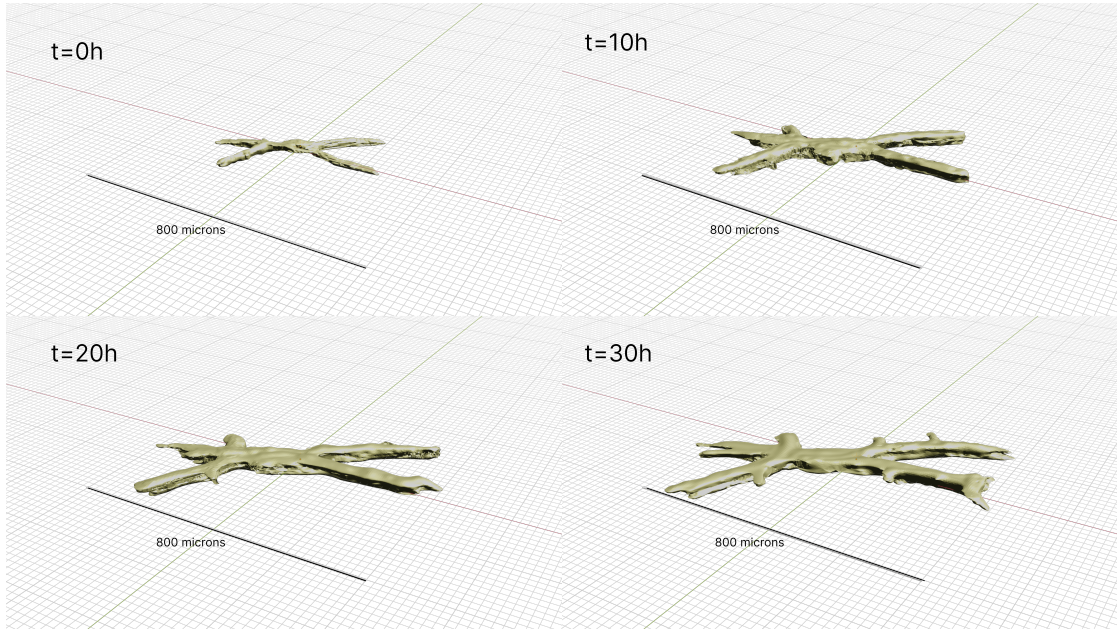


Figure 4.6: 3D models of A1P5 every 10h. $t=0h$ is normalised to the beginning of day 5. It shows the growth over a period of 30 h. The branches gain in girth early on in development, and don't change much later on. New branching only occurs heavily around the 20 h mark from various points along the branches.

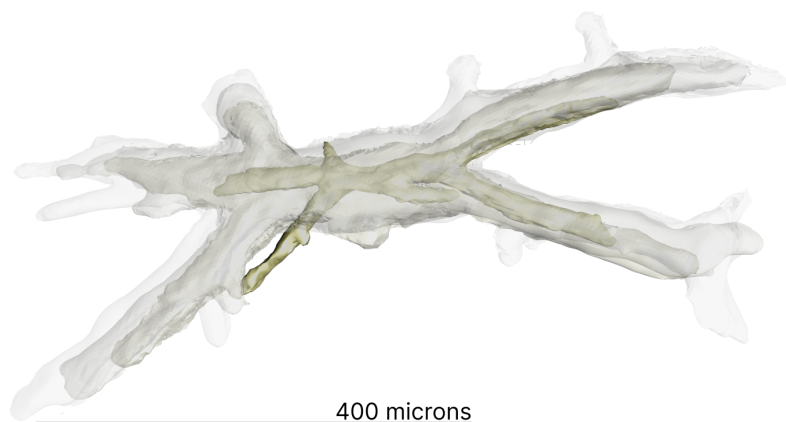
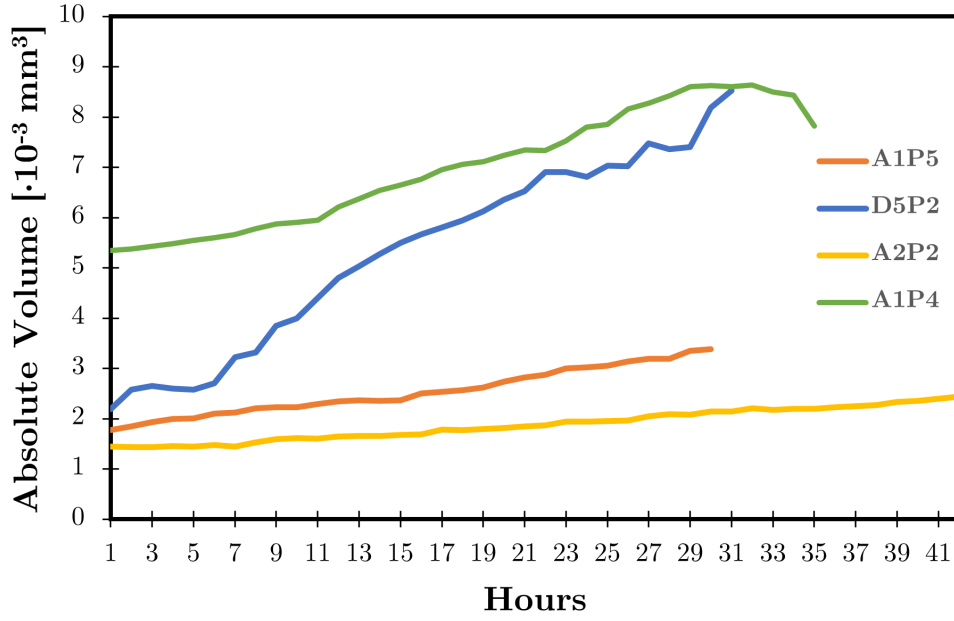


Figure 4.7: The growth of A1P5 every 10h. Note the longitudinal growth on the main axis at the beginning, and then focusing more on extending the branches

4.3 Volumetric analysis of 3D model

In combination with two more organoid models shown in fig. 3.2, the volume can be plotted over time (fig. 4.8). The growth is normalised to the organoid's size at hour 30 from day five.

The branched organoid (D5P2) displays a completely different growth pattern to the others which all grow longitudinally. It starts off at a much lower relative size (26.8%) than the other three (60.4% on average). It grows exceptionally fast, at a rate of $2.62\% \text{ h}^{-1}$ (vs. $1.28\% \text{ h}^{-1}$) until hour 23, from where on it exhibits a similar behaviour to the longitudinal growing ones.



(a) Volume of the organoids over a time period of > 30 h.

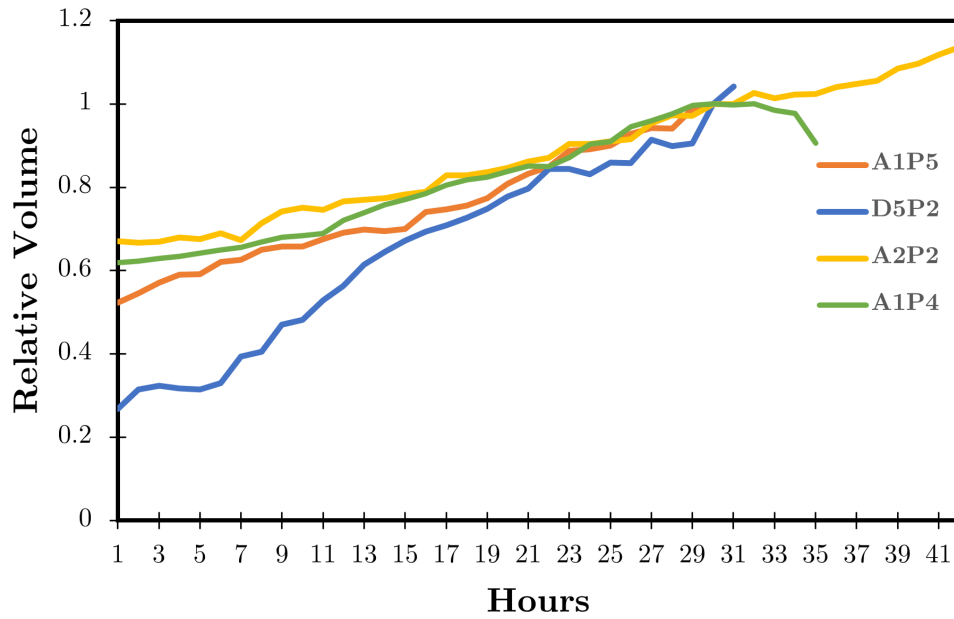
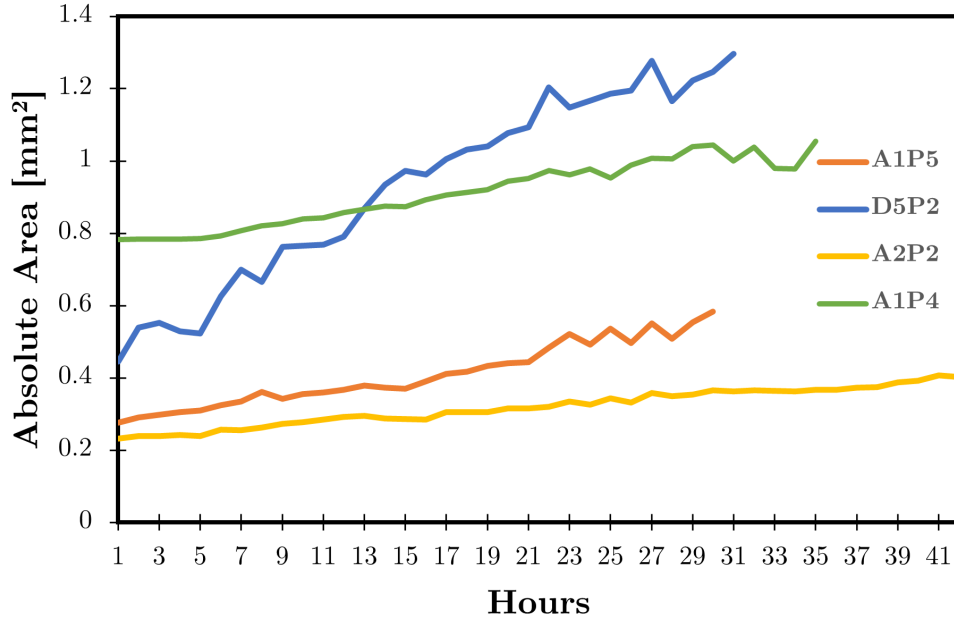
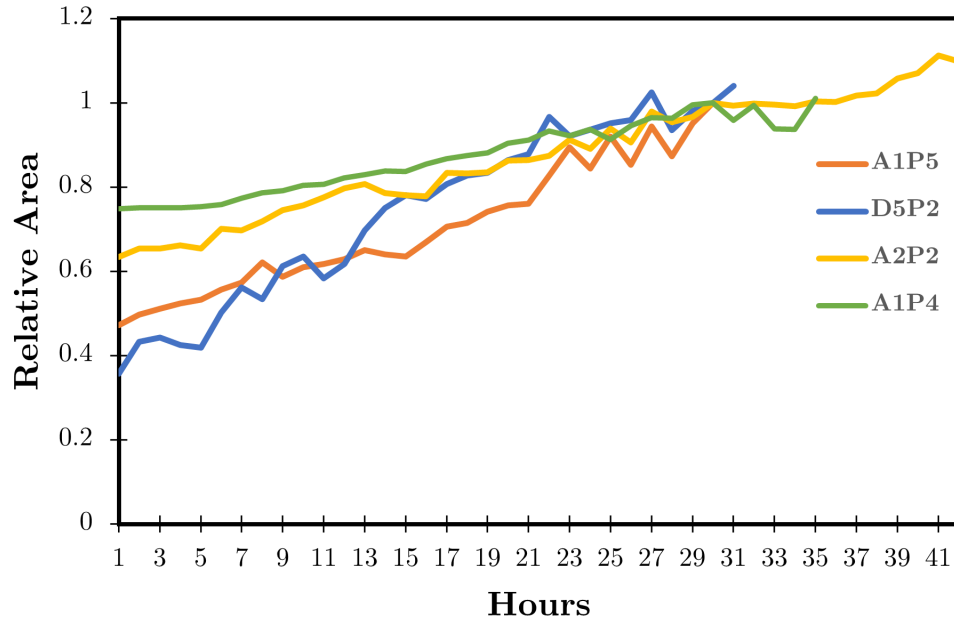
(b) Volume normalised to the organoids size at $t = 30$ h.

Figure 4.8: Volumetric growth of the inspected organoids from fig. 3.2. The extremely branched D5P2 has the fastest growth, which is evident especially in (b), with a growth of $2.62\% \text{ h}^{-1}$. The more linearly structured organoids exhibit a slower and more uniform growth (A1P5, A2P2, A1P4).



(a) Absolute growth of the surface area over time. Organoid D5P2 has an extremely fast growth, whereas A2P2 grows linearly and relatively slowly.



(b) Relative growth of the organoid surface area.

Figure 4.9: Surface area development over a period of > 30 h. The differences in growth speed are less evident but still noticeable. There are more fluctuations in the data due to inconsistencies in the surface mesh of the models.

4 Results

The pattern for the surface area is less evident, but still noticeable. Due to the inconsistencies in the mesh creation, there are fluctuations in the surface area, especially around hours 23 to 30, but the overall tendencies are similar to the volumetric growth.

5 Discussion

The approach of using the machine-learning model U-net to segment organoids proved successful. Large data sets were effectively masked and were highly similar to their corresponding manual segmentation. This took 12 minutes per frame instead of the one second for U-net. The processing time can even be sped up with improved hardware. Additional local processing power is expensive, so cloud-based solutions could aid in increasing the speed and accuracy.

The difficulty in the segmentation with U-net was having a large, pre-processed data set. The training images had to be labeled (manually segmented), which was a lengthy process, but fortunately only has to be done once. The data augmentation through rotation and mirroring helped in increasing the data set size, other image transformations could have been used as well.

A larger data set with more differentiated organoid morphology would have made the network more robust and the results more substantial. A method of creating a larger data set without changing the microscopy process is through voxelization. The 3D models could be broken back down into a theoretically infinite number of 2D images and used for data augmentation of the training set. This would provide more exact results on the following training runs. The code for voxelization is easy to run and is given here:

```
1 pip install stl-to-voxel
2 stltovoxel 3D.stl 2D_images.png --voxel-size 1 --resolution 256
```

Program 1: Simple voxelization utility in python using the "stl-to-voxel" package from [52]. It converts an STL-file to a specified number of individual, masked PNGs. These can be used for data augmentation of U-net, for example.

To increase the variety of organoid shapes would require imaging more cell cultures. This is not straightforward to do, as it requires more primary tissue from clinical interventions. The cell cultures also go through a lengthy process of cultivation and medium changes to create the right conditions, so an immediate increase in available organoids to image is not always possible.

In addition to three-dimensional data sets, images of similar cell structures in 2D cultures could also make the network more robust and exact in its segmentation. Adding more "difficult" images, either with artifacts, contrast issues or overlapping structures would also bolster U-net for suboptimal segmentation conditions. This would also make the method more applicable to more conventional setups and different use cases, too. For the segmentation in this thesis, the z-stacks had to be carefully prepared and selected first.

5 Discussion

The creation of the 3D model required an accurate segmentation. Any artifacts result in an uneven and incomplete mesh, which falsifies volumetric and surface area readings. Smoothing algorithms make the model more presentable without changing the data, but they can only be applied to an already complete mesh.

With regard to the growth patterns of organoids, it was found that the branched organoid D5P2 grew significantly faster than its less branched counterparts. It would have been beneficial to compare more morphologically similar organoids. This would have required a larger data set. For the analysis in this thesis, four individual, three-dimensional organoids were used, three of which had a similar morphology and growth behaviour. While this confirmed the consistency in growth of longitudinal growing organoids, a more varied data set would have provided more insight into the differences of growth patterns between morphologies. More complex structures are more difficult to segment, however, and would have required a better trained network to capture all branching details.

The imaging time of the data set was also limited to under three days, preventing a comprehensive analysis of all growth phases. The difference in growth speed of the branched organoid could thus also be attributed to a different onset of the extension phase in each typology of organoid. It would be interesting to do a similar analysis, but from initial growth to the lumen formation phase, on multiple, morphologically diverse organoids. This would enable an exhaustive investigation of the actual differences between organoid types.

The bulges that form at branching points are particularly interesting. On a cellular level, they could provide more insights into the reason organoids branch out when they do, what physical and genetic mechanisms underly the branching process and how it could be controlled.

6 Conclusion and Outlook

This thesis has demonstrated the practicality of the CNN-based U-net for segmentation of three-dimensional cell cultures. Large data sets were processed quickly and accurately, from the initial z-stacks to accurately masked images. The 3D models created from the segmentation proved essential in the analysis of volumetric data and branching behaviour. It was shown that different organoid morphologies exhibit different growth profiles, and branching events are preceded by a build up of cells. Further quantitative analyses are possible from the models.

As a consequence, the effect of drug treatments on growth behaviour could be effectively analysed with this method. Applying drugs to the organoids, imaging the growth before and after and using these methods to create a 3D model would provide researchers with information on the effect of the drugs on organoid growth behaviour. The knowledge gained could contribute to the understanding of pathologies and their treatment.

Additionally, the models can be used for 3D printing which would be useful for demonstration purposes and even doctor-to-patient interactions. Furthermore, in an increasingly digitalized world, it paves the path for augmented reality or virtual reality models which can be stored on a cloud and exchanged between members of a team around the world in near real-time.

It is to be expected that these methods would also work for similar objects and can be used for parametrization in fundamental research beyond the field of biology.

Appendix A

Datasets

220711_9591 Day 5-8_Calyculin_Plate.2

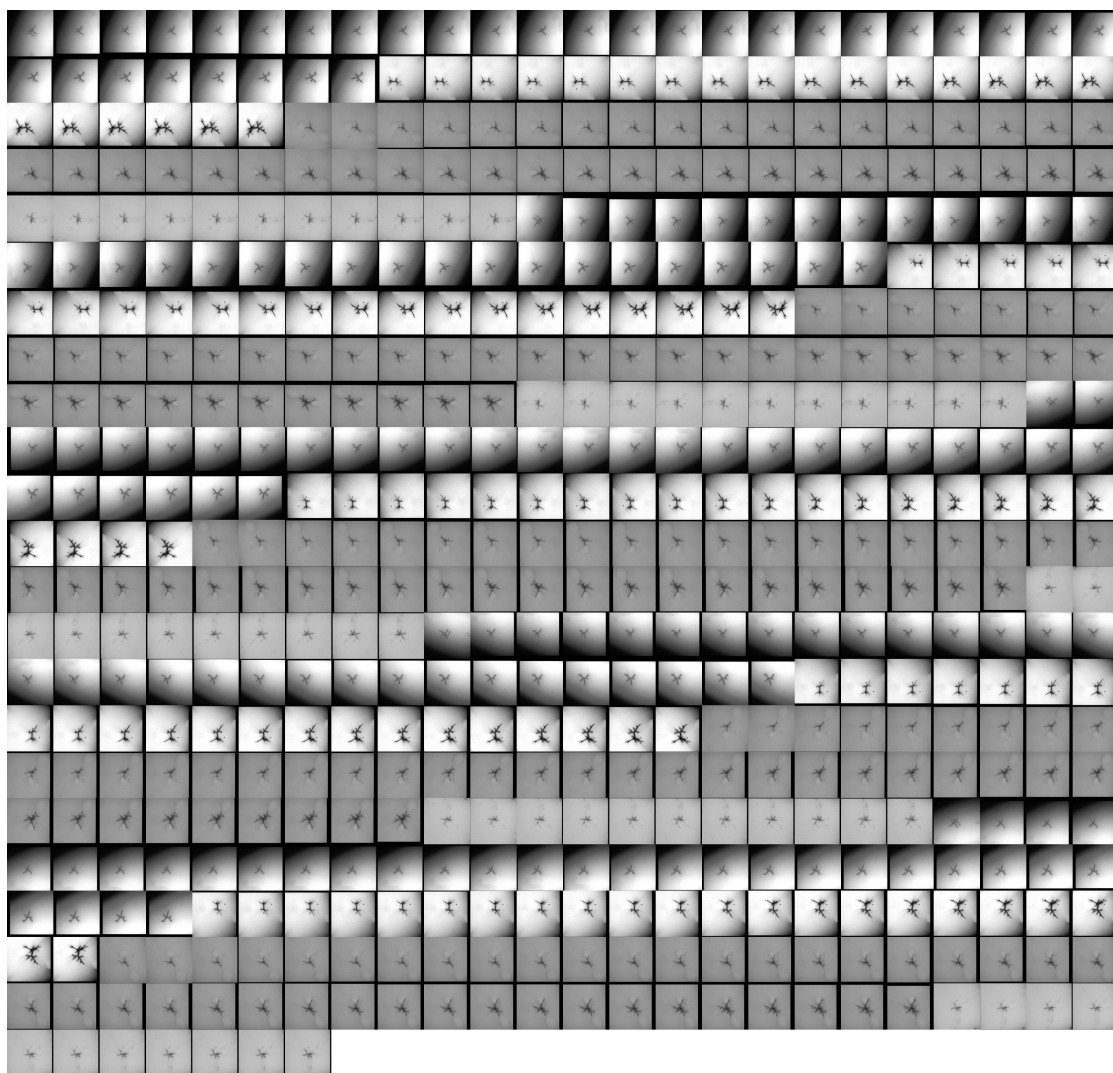


Figure A.1: Training dataset montage 1

Appendix A Datasets

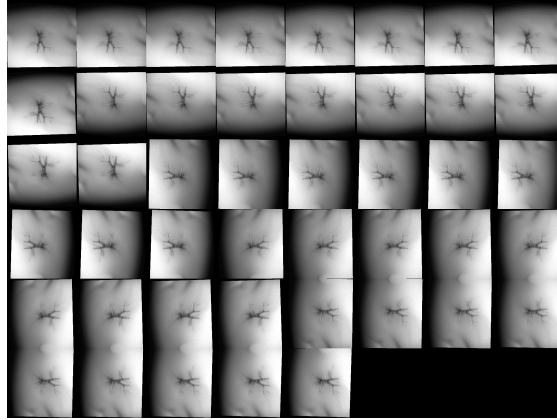


Figure A.2: Training dataset montage 2

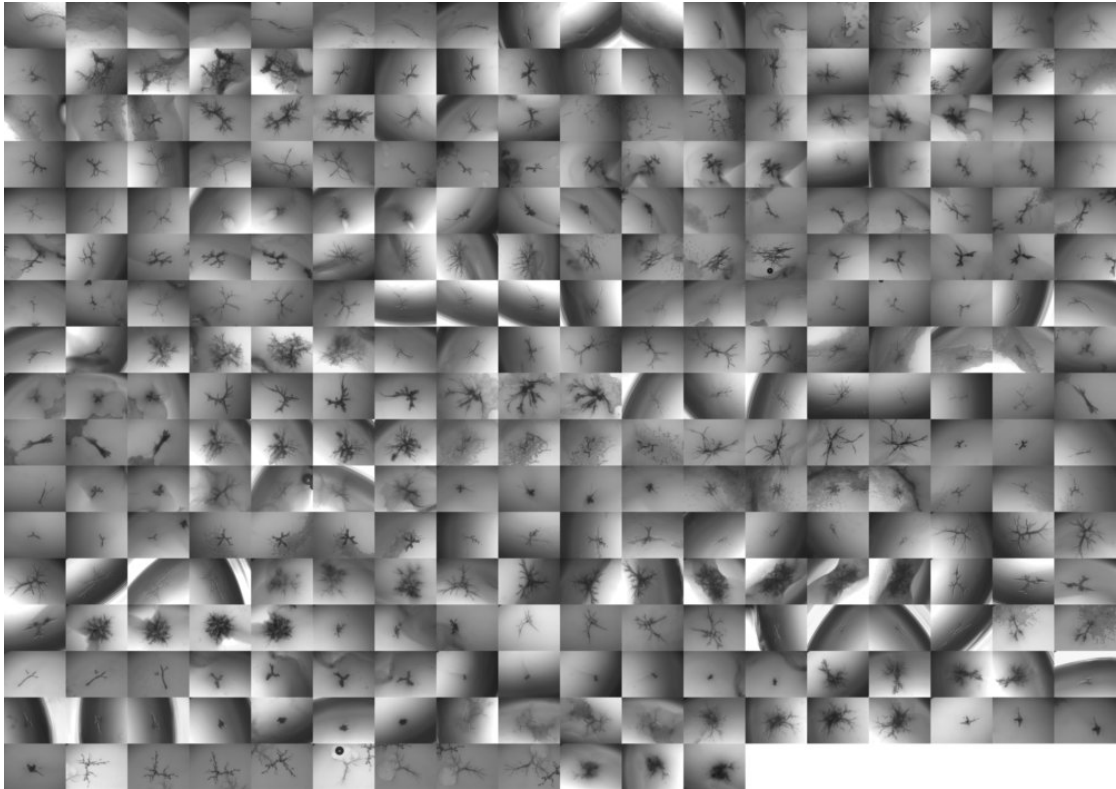


Figure A.3: Training dataset montage 3

Appendix B

Code

Model:

```
1 import numpy as np
2 import os
3 import skimage.io as io
4 import skimage.transform as trans
5 import numpy as np
6 from keras.models import *
7 from keras.layers import *
8 from keras.optimizers import *
9 from keras.callbacks import ModelCheckpoint, LearningRateScheduler
10 from keras import backend as keras
11
12
13 def unet(pretrained_weights = None, input_size = (256,256,1)):
14     inputs = Input(input_size)
15     conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same',
16 kernel_initializer = 'he_normal')(inputs)
17     conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same',
18 kernel_initializer = 'he_normal')(conv1)
19     pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
20     conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same',
21 kernel_initializer = 'he_normal')(pool1)
22     conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same',
23 kernel_initializer = 'he_normal')(conv2)
24     pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
25     conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same',
26 kernel_initializer = 'he_normal')(pool2)
27     conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same',
28 kernel_initializer = 'he_normal')(conv3)
29     pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
30     conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same',
31 kernel_initializer = 'he_normal')(pool3)
32     conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same',
33 kernel_initializer = 'he_normal')(conv4)
34     drop4 = Dropout(0.5)(conv4)
35     pool4 = MaxPooling2D(pool_size=(2, 2))(drop4)
36
37     conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same',
38 kernel_initializer = 'he_normal')(pool4)
39     conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same',
40 kernel_initializer = 'he_normal')(conv5)
41     drop5 = Dropout(0.5)(conv5)
```

Appendix B Code

```
32
33     up6 = Conv2D(512, 2, activation = 'relu', padding = 'same',
34     kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(drop5))
35     merge6 = concatenate([drop4,up6], axis = 3)
36     conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same',
37     kernel_initializer = 'he_normal')(merge6)
38     conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same',
39     kernel_initializer = 'he_normal')(conv6)
40
41     up7 = Conv2D(256, 2, activation = 'relu', padding = 'same',
42     kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv6))
43     merge7 = concatenate([conv3,up7], axis = 3)
44     conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same',
45     kernel_initializer = 'he_normal')(merge7)
46     conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same',
47     kernel_initializer = 'he_normal')(conv7)
48
49     up8 = Conv2D(128, 2, activation = 'relu', padding = 'same',
50     kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv7))
51     merge8 = concatenate([conv2,up8], axis = 3)
52     conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same',
53     kernel_initializer = 'he_normal')(merge8)
54     conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same',
55     kernel_initializer = 'he_normal')(conv8)
56
57     up9 = Conv2D(64, 2, activation = 'relu', padding = 'same',
58     kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv8))
59     merge9 = concatenate([conv1,up9], axis = 3)
60     conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same',
61     kernel_initializer = 'he_normal')(merge9)
62     conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same',
63     kernel_initializer = 'he_normal')(conv9)
64     conv9 = Conv2D(2, 3, activation = 'relu', padding = 'same',
65     kernel_initializer = 'he_normal')(conv9)
66     conv10 = Conv2D(1, 1, activation = 'sigmoid')(conv9)
67
68     model = Model(inputs, conv10)
69
70     model.compile(optimizer = Adam(lr = 1e-4), loss = '
71     binary_crossentropy', metrics = ['accuracy'])
72
73     #model.summary()
74
75     if(pretrained_weights):
76         model.load_weights(pretrained_weights)
77
78     return model
```

Data:

```

1 from __future__ import print_function
2 from keras.preprocessing.image import ImageDataGenerator
3 import numpy as np
4 import os
5 import glob
6 import skimage.io as io
7 import skimage.transform as trans
8 from skimage.util import img_as_ubyte
9
10
11 Sky = [128,128,128]
12 Building = [128,0,0]
13 Pole = [192,192,128]
14 Road = [128,64,128]
15 Pavement = [60,40,222]
16 Tree = [128,128,0]
17 SignSymbol = [192,128,128]
18 Fence = [64,64,128]
19 Car = [64,0,128]
20 Pedestrian = [64,64,0]
21 Bicyclist = [0,128,192]
22 Unlabelled = [0,0,0]
23
24 COLOR_DICT = np.array([Sky, Building, Pole, Road, Pavement,
25                        Tree, SignSymbol, Fence, Car, Pedestrian,
26                        Bicyclist, Unlabelled])
27
28 def adjustData(img,mask,flag_multi_class,num_class):
29     if(flag_multi_class):
30         img = img / 255
31         mask = mask[:, :, :, 0] if (len(mask.shape) == 4) else mask[:, :, 0]
32         new_mask = np.zeros(mask.shape + (num_class,))
33         for i in range(num_class):
34             #for one pixel in the image, find the class in mask and
35             #convert it into one-hot vector
36             index = np.where(mask == i)
37             index_mask = (index[0],index[1],index[2],np.zeros(len(index
38 [0]),dtype = np.int64) + i) if (len(mask.shape) == 4) else (index[0],
39 index[1],np.zeros(len(index[0]),dtype = np.int64) + i)
40             #new_mask[index_mask] = 1
41             new_mask[mask == i,i] = 1
42         new_mask = np.reshape(new_mask,(new_mask.shape[0],new_mask.shape
43 [1]*new_mask.shape[2],new_mask.shape[3])) if flag_multi_class else np.
44 reshape(new_mask,(new_mask.shape[0]*new_mask.shape[1],new_mask.shape
45 [2]))
46     mask = new_mask
47     elif(np.max(img) > 1):
48         img = img / 255
49         mask = mask /255
50         mask[mask > 0.5] = 1
51         mask[mask <= 0.5] = 0

```

Appendix B Code

```
46     return (img,mask)
47
48
49
50 def trainGenerator(batch_size,train_path,image_folder,mask_folder,
51     aug_dict,image_color_mode = "grayscale",
52     mask_color_mode = "grayscale",image_save_prefix = "
53     image",mask_save_prefix = "mask",
54     flag_multi_class = False,num_class = 2,save_to_dir =
55     None,target_size = (256,256),seed = 1):
56     '''
57     can generate image and mask at the same time
58     use the same seed for image_datagen and mask_datagen to ensure the
59     transformation for image and mask is the same
60     if you want to visualize the results of generator, set save_to_dir =
61     "your path"
62     '''
63     image_datagen = ImageDataGenerator(**aug_dict)
64     mask_datagen = ImageDataGenerator(**aug_dict)
65     image_generator = image_datagen.flow_from_directory(
66         train_path,
67         classes = [image_folder],
68         class_mode = None,
69         color_mode = image_color_mode,
70         target_size = target_size,
71         batch_size = batch_size,
72         save_to_dir = save_to_dir,
73         save_prefix = image_save_prefix,
74         seed = seed)
75     mask_generator = mask_datagen.flow_from_directory(
76         train_path,
77         classes = [mask_folder],
78         class_mode = None,
79         color_mode = mask_color_mode,
80         target_size = target_size,
81         batch_size = batch_size,
82         save_to_dir = save_to_dir,
83         save_prefix = mask_save_prefix,
84         seed = seed)
85     train_generator = zip(image_generator, mask_generator)
86     for (img,mask) in train_generator:
87         img,mask = adjustData(img,mask,flag_multi_class,num_class)
88         yield (img,mask)
89
90
91
92 def testGenerator(test_path,num_image = 31,target_size = (256,256),
93     flag_multi_class = False,as_gray = True):
94     for i in range(num_image):
95         img = io.imread(os.path.join(test_path,"%d.jpg"%i),as_gray =
96         as_gray)
97         #img = io.imread(os.path.join(test_path, "%d.png" % i), as_gray=
98         as_gray)
99         img = img / 255
```

Appendix B Code

```
92     img = trans.resize(img,target_size)
93     img = np.reshape(img,img.shape+(1,)) if (not flag_multi_class)
else img
94     img = np.reshape(img,(1,)+img.shape)
95     yield img
96
97
98 def geneTrainNpy(image_path,mask_path,flag_multi_class = False,num_class
= 2,image_prefix = "image",mask_prefix = "mask",image_as_gray = True,
mask_as_gray = True):
99     image_name_arr = glob.glob(os.path.join(image_path,"%s*.jpg"%
image_prefix))
100     #image_name_arr = glob.glob(os.path.join(image_path, "%s*.png" %
image_prefix))
101     image_arr = []
102     mask_arr = []
103     for index,item in enumerate(image_name_arr):
104         img = io.imread(item,as_gray = image_as_gray)
105         img = np.reshape(img,img.shape + (1,)) if image_as_gray else img
106         mask = io.imread(item.replace(image_path,mask_path).replace(
image_prefix,mask_prefix),as_gray = mask_as_gray)
107         mask = np.reshape(mask,mask.shape + (1,)) if mask_as_gray else
mask
108         img,mask = adjustData(img,mask,flag_multi_class,num_class)
109         image_arr.append(img)
110         mask_arr.append(mask)
111     image_arr = np.array(image_arr)
112     mask_arr = np.array(mask_arr)
113     return image_arr,mask_arr
114
115
116 def labelVisualize(num_class,color_dict,img):
117     img = img[:, :,0] if len(img.shape) == 3 else img
118     img_out = np.zeros(img.shape + (3,))
119     for i in range(num_class):
120         img_out[img == i,:] = color_dict[i]
121     return img_out / 255
122
123
124
125 def saveResult(save_path,npyfile,flag_multi_class = False,num_class = 2):
126     for i,item in enumerate(npyfile):
127         img = labelVisualize(num_class,COLOR_DICT,item) if
flag_multi_class else item[:, :,0]
128         #io.imsave(os.path.join(save_path,"%d_predict.png"%i),img)
129         io.imsave(os.path.join(save_path,"%d_predict.jpg"%i),img_as_ubyte
(img))
130 #img_out = img_as_ubyte(img_out)
```

Main:

```

1 from model import *
2 from data import *
3 #from keras.utils import multi_gpu_model
4
5 def importing(x):
6     from keras.utils import multi_gpu_model
7
8 os.environ["CUDA_VISIBLE_DEVICES"] = "0"
9 print(os.environ["CUDA_VISIBLE_DEVICES"])
10
11 data_gen_args = dict(rotation_range=0.2,
12                       width_shift_range=0.05,
13                       height_shift_range=0.05,
14                       shear_range=0.05,
15                       zoom_range=0.05,
16                       horizontal_flip=True,
17                       fill_mode='nearest')
18
19 myGene = trainGenerator(2, 'data/membrane/train', 'image', 'label',
20                          data_gen_args, save_to_dir = None)
21
22 model = unet()
23 model_checkpoint = ModelCheckpoint('unet_membrane.hdf5', monitor='loss',
24                                   verbose=1, save_best_only=True)
25 model.fit_generator(myGene, steps_per_epoch=2, epochs=1000, callbacks=[
26     model_checkpoint], verbose=1)
27
28 testGene = testGenerator("data/membrane/test")
29 results = model.predict_generator(testGene, verbose=1, steps=20)
30 saveResult("data/membrane/test", results)

```

References

- [1] World Health Organization. Cancer. URL <https://www.who.int/news-room/fact-sheets/detail/cancer>. Last visited on 2022-11-24.
- [2] World Cancer Research Fund International. Global cancer data by country. URL <https://www.wcrf.org/cancer-trends/global-cancer-data-by-country/>. Last visited on 2022-11-24.
- [3] Centers for Disease Control and Prevention. Leading causes of death in the u.s. URL <https://www.cdc.gov/nchs/fastats/leading-causes-of-death.htm>. Last visited on 2022-11-24.
- [4] A Kamb. What's wrong with our cancer models? *Nature Reviews Drug Discovery*, 4:161–165, 2005. URL <https://doi.org/10.1038/nrd1635>. Last visited on 2022-11-25.
- [5] Pierre-Olivier Frappart and Thomas G. Hofmann. Pancreatic ductal adenocarcinoma (pdac) organoids: The shining light at the end of the tunnel for drug response prediction and personalized medicine. *Cancers*, 12:10, September 2020. URL <https://doi.org/10.3390/cancers12102750>. Last visited on 2022-11-26.
- [6] E Smith and W.J. Cochrane. Cystic organoid teratoma: (report of a case). *Cancers*, 55,2:151–152, August 1946. URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1582935>. Last visited on 2022-11-26.
- [7] Marina Simian and Mina J Bissell. Organoids: A historical perspective of thinking in three dimensions. *The Journal of Cell Biology*, 216,1:31–40, February 2017. URL <https://doi.org/10.1083/jcb.201610056>. Last visited on 2022-11-26.
- [8] A. Fatehullah, S. Tan, and N. Barker. Organoids as an in vitro model of human development and disease. *Nature Cell Biology*, 18:246–254, February 2016. URL <https://doi.org/10.1038/ncb3312>. Last visited on 2022-11-25.
- [9] S. Randriamanantsoa, A. Papargyriou, and H.C. et al. Maurer. Spatiotemporal dynamics of self-organized branching in pancreas-derived organoids. *Nature*, 13 (5219), September 2022. URL <https://doi.org/10.1038/s41467-022-32806-y>. Last visited on 2022-11-22.
- [10] Koo BK. Knoblich J.A. Kim, J. Human organoids: model systems for human biology and medicine. *Nature Reviews Molecular Cell Biology*, 21:571–584, October 2020. URL <https://doi.org/10.1038/s41580-020-0259-3>.

References

- [11] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. URL <http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a>. (available on arXiv:1505.04597 [cs.CV]).
- [12] Development of collagen-based 3d matrix for gastrointestinal tract-derived organoid culture. *Stem Cells Int.*, 13, 2019.
- [13] Millipore Sigma. 3d organoid culture: New in vitro models of development and disease. URL <https://www.sigmaaldrich.com/US/en/technical-documents/technical-article/cell-culture-and-cell-culture-analysis/3d-cell-culture/3d-organoid-culture>. Last visited on 2022-11-22.
- [14] D.-J. Cheon and S. Orsulic. Mouse models of cancer. *Annual Review of Pathology: Mechanisms of Disease*, 6:95–119, February 2011. URL <https://doi.org/10.1146/annurev.pathol.3.121806.154244>. Last visited on 2022-11-25.
- [15] J. Drost and H. Clevers. Organoids in cancer research. *Nature Reviews Cancer*, 18: 407—418, April 2018. URL <https://doi.org/10.1038/s41568-018-0007-6>. Last visited on 2022-11-25.
- [16] Yaqi et al. Li. Organoid based personalized medicine: from bench to bedside. *Cell Regeneration*, 9:21, November 2020. URL <https://doi.org/10.1186/s13619-020-00059-z>. Last visited on 2022-11-25.
- [17] National Cancer Institute. Patient-derived xenograft. URL <https://www.cancer.gov/publications/dictionaries/cancer-terms/def/patient-derived-xenograft>. Last visited on 2022-11-25.
- [18] Ha G. Tseng YY. et al Ben-David, U. Patient-derived xenografts undergo mouse-specific tumor evolution. *Nature Genetics*, 49:1567—1575, 10 2017. URL <https://doi.org/10.1038/ng.3967>.
- [19] R.J. Porter, G.I. Murray, and M.H. McLean. Current concepts in tumour-derived organoids. *British Journal of Cancer*, 123:1209–1218, February 2020. URL <https://doi.org/10.1038/s41416-020-0993-5>. Last visited on 2022-11-26.
- [20] Xiaoxue Ren, Weikang Chen, Qingxia Yang, Xiaoxing Li, and Lixia Xu. Patient-derived cancer organoids for drug screening: Basic technology and clinical application. *Journal of Gastroenterology and Hepatology*, 37(8):1446–1454, 2022. doi: <https://doi.org/10.1111/jgh.15930>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1111/jgh.15930>.
- [21] Benedetta Artigiani and Hans Clevers. Use and application of 3D-organoid technology. *Human Molecular Genetics*, 27(R2):R99–R107, 05 2018. URL <https://doi.org/10.1093/hmg/ddy187>.

References

- [22] Madeline A. Lancaster and Jürgen A. Knoblich. Organogenesis in a dish: Modeling development and disease using organoid technologies. *Science*, 345, 6194, July 2014. URL <https://doi.org/10.1126/science.1247125>. Last visited on 2022-11-27.
- [23] Lumir Kunovsky, Pavla Tesarikova, Zdenek Kala, Radek Kroupa, Petr Kysela, Jiri Dolina, and Jan Trna. Pancreatic cancer organoids recapitulate disease and allow personalized drug screening. *Proceedings of the National Academy of Sciences*, 116,52, December 2019. URL <https://doi.org/10.1073/pnas.191127311>. Last visited on 2022-11-27.
- [24] Martin J. Booth, Delphine Débarre, and Alexander Jesacher. Adaptive optics for biomedical microscopy. *Optics and photonics news*, January 2012. URL https://www.optica-opn.org/home/articles/volume_23/issue_1/features/adaptive_optics_for_biomedical_microscopy/. Last visited on 2022-12-05.
- [25] Brown C.M. Wright G.D. et al Jonkman, J. Tutorial: guidance for quantitative confocal microscopy. *Nature Protocols*, 15:1585—1611, March 2020. URL <https://doi.org/10.1038/s41596-020-0313-9>. Last visited on 2022-12-05.
- [26] Kandathil S.M. Moffat L. et al Greener, J.G. A guide to machine learning for biologists. *Nature Reviews Molecular Cell Biology*, 23:40–55, 2022. URL <https://doi.org/10.1038/s41580-021-00407-0>. Last visited on 2022-12-01.
- [27] E. Moen, D. Bannon, and T. et al. Kudo. Deep learning for cellular image analysis. *Nature Methods*, 16:1233–1246, May 2019. URL <https://doi.org/10.1038/s41592-019-0403-1>. Last visited on 2022-11-27.
- [28] Wikipedia contributors. Neural networks. URL https://en.wikipedia.org/w/index.php?title=Neural_network&oldid=1124274991. Last visited on 2022-12-03.
- [29] IBM Cloud Education. Neural networks, . URL <https://www.ibm.com/cloud/learn/neural-networks>. Last visited on 2022-12-03.
- [30] IBM Cloud Education. Convolutional neural networks, . URL <https://www.ibm.com/cloud/learn/convolutional-neural-networks>. Last visited on 2022-12-03.
- [31] Anh H. Reynolds. Convolutional neural networks (cnns). URL <https://www.educative.io/answers/overfitting-and-underfitting>. Last visited on 2022-12-04.
- [32] IBM Cloud Education. Supervised vs. unsupervised learning: What’s the difference?, . URL <https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning>. Last visited on 2022-12-04.
- [33] Educative Answers Team. Overfitting and underfitting. URL <https://www.educative.io/answers/overfitting-and-underfitting>. Last visited on 2022-12-04.

References

- [34] S. Sharma, S. Sharma, and Athaiya A. Activation functions in neural networks. *International Journal of Engineering Applied Sciences and Technology*, 4, 12:310–316, April 2020. ISSN 2455–2143. URL <https://doi.org/10.48550/arXiv.1411.4038>. Last visited on 2022-12-01.
- [35] Laxman Singh. Forward and backward propagation — understanding it to master the model training process. URL <https://medium.com/geekculture/forward-and-backward-propagation-understanding-it-to-master-the-model-training-process-3819727dc5c1>. Last visited on 2022-12-04.
- [36] Hossein Gholamalinezhad and Hossein Khosravi. Pooling methods in deep neural networks, a review. *CoRR*, abs/2009.07485, 2020. URL <https://arxiv.org/abs/2009.07485>.
- [37] Padding and stride. URL https://d2l.ai/chapter_convolutional-neural-networks/padding-and-strides.html. Last visited on 2022-12-10.
- [38] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014. URL <http://arxiv.org/abs/1411.4038>. Last visited on 2022-12-01.
- [39] Lumir Kunovsky, Pavla Tesarikova, Zdenek Kala, Radek Kroupa, Petr Kysela, Jiri Dolina, and Jan Trna. The use of biomarkers in early diagnostics of pancreatic cancer. *Canadian Journal of Gastroenterology and Hepatology*, August 2018. URL <https://doi.org/10.1155/2018/5389820>. Last visited on 2022-11-26.
- [40] S.R. Nelson, C. Zhang, and S. et al. Roche. Modelling of pancreatic cancer biology: transcriptomic signature for 3d pdx-derived organoids and primary cell line organoid development. *Scientific Reports*, 10,2778, February 2020. URL <https://doi.org/10.1038/s41598-020-59368-7>. Last visited on 2022-11-27.
- [41] Philippe Thévenaz. Stackreg - an imagej plugin for the recursive alignment of a stack of images. URL <http://bigwww.epfl.ch/thevenaz/stackreg/>. Last visited on 2022-12-06.
- [42] ImageJ Documentation. Z-functions. URL <https://imagej.net/imaging/z-functions>. Last visited on 2022-12-07.
- [43] Jui-Cheng Yen, Fu-Juay Chang, and Shyang Chang. A new criterion for automatic multilevel thresholding. *IEEE Transactions on Image Processing*, 4(3):370–378, 1995. doi: 10.1109/83.366472.
- [44] A.G. Shanbhag. Utilization of information measure as a means of image thresholding. *CVGIP: Graphical Models and Image Processing*, 56(5):414–419, 1994. ISSN 1049-9652. doi: <https://doi.org/10.1006/cgip.1994.1037>. URL <https://www.sciencedirect.com/science/article/pii/S1049965284710376>.

References

- [45] Judith M. S. Prewitt and Mortimer L. Mendelsohn. The analysis of cell images*. *Annals of the New York Academy of Sciences*, 128(3):1035–1053, 1966. doi: <https://doi.org/10.1111/j.1749-6632.1965.tb11715.x>. URL <https://nyaspubs.onlinelibrary.wiley.com/doi/abs/10.1111/j.1749-6632.1965.tb11715.x>.
- [46] Danqing Liu. A practical guide to relu. URL <https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7>. Last visited on 2022-12-04.
- [47] Xavier Glorot, Antoine Bordes, and Y. Bengio. Deep sparse rectifier neural networks. volume 15, 01 2010.
- [48] Zhixuhao. U-net. URL <https://github.com/zhixuhao/unet>. Last visited on 2022-11-28.
- [49] Ö. Çiçek, A. Abdulkadir, S.S. Lienkamp, T. Brox, and O. Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation. In M.R. Sabuncu G. Unal S. Ourselin, W.S. Wells and L. Joskowicz, editors, *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9901 of *LNCS*, pages 424–432. Springer, Oct 2016. URL <http://lmb.informatik.uni-freiburg.de/Publications/2016/CABR16>. (available on arXiv:1606.06650 [cs.CV]).
- [50] Z. Wang, E.P. Simoncelli, and A.C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, volume 2, pages 1398–1402 Vol.2, 2003. doi: 10.1109/ACSSC.2003.1292216.
- [51] Alex Herbert. Imagej findfoci plugins. URL <https://www.sussex.ac.uk/gdsc/intranet/pdfs/FindFoci.pdf>. Last visited on 2022-11-22.
- [52] Christian Pederkoff. stl-to-voxel. URL <https://github.com/cpederkoff/stl-to-voxel>. Last visited on 2022-11-22.

List of Figures

2.1	Comparison of cancer cell models, from [16]. Human tissue can be used to establish two-dimensional cell lines, patient-derived xenografts (implantation of human tissue in rodents) or organoids.	3
2.2	Recent developments in the field of patient-derived organoids (PDOs), with the year when the culture was first established, from [20]	4
2.3	Differences in imaging of wide-field and confocal microscopes, from [24, 25]. Note the more defined beam on the confocal microscope which yields very high-resolution images but is disadvantageous when imaging quickly and over a long period of time.	6
2.4	Advantages of each microscope type (widefield, confocal lases scanning microscope and spinning disk), from [25]. The strong points of widefield microscopy lie in its speed and sample viability (preserving the samples properties).	6
2.5	Layout of artificial neural networks, showing connected nodes in layers. The connections are depicted as weighted arrows. From [28].	8
2.6	The principle of convolution, from [31]	9
2.7	An example of underfitting and overfitting of a model, from [33]	9
2.8	Example of pooling algorithms, from [36]	10
3.1	Development phases of PDAC organoids stained with SiRDNA, from [9] .	13
3.2	The four organoids that were used for further analysis.	14
3.3	Image processing, from the stack of 2D images, over the pre-processing and segmentation to the three-dimensional model.	15
3.4	The ImageJ Z-Project function applied to organoid A1P5. Each algorithm shows slightly different details, but don't offer a major advantage over the basic 2D image.	16
3.5	Semi-automatic segmentation of a frame of A1P5 using ImageJ	17
3.6	Comparison of the auto-thresholding methods applied to A1P5	18
3.7	An example of a 2D image segmentation using U-net on a 2D image. The input image (left) is segmented by the CNN (middle) and then a mask can easily be created. The organoid is successfully binarily distinguished from its background.	19
3.8	To utilise U-net to create a 3D model, it is necessary to split each organoid into single 2D files, crop and ideally drift-correct. These single slices can then be fed to the python script (see the entire code in detail in the appendix B).	19

List of Figures

3.9	The architecture of U-net, from [11]. The left side shows the contracting side, a series of downwards convolutions and max pooling. The rightside essentially retraces the steps through upwards convolution and creates a segmentation map as the output.	20
3.10	The ReLU activation function, also known as the rectifier, is defined as the positive part of its argument: $f(x) = \max(0, x)$. It is a computationally simple function to handle, is scale invariant and offers a better gradient propagation than other functions [46, 47].	21
3.11	Examples of types of organoid structures used for training with their respective manual segmentation below	22
4.1	Organoid A1P5, segmented with training run T1 (lower settings). The outlines are well defined but the contrast is not as pronounced as with T2 (see below).	25
4.2	Some more complex organoids were segmented with U-net to test its capabilities, with the high settings of training T2. The second column shows the output from U-net which is then masked (third column). For comparison, a manual masking was done, as seen on the right. Running U-net takes about 1 s per image, while the manual masking at this level of detail took around 12 min per image in ImageJ. The statistical comparison can be found in table 4.1.	27
4.3	Figure 4.2 (b, f, j) enlarged to show the gradient across the branches which indicates the probability density.	28
4.4	3D representation of organoid A1P5 at the $t = 0$. Note the branching along the main axis of extension.	30
4.5	3D representation of organoid D5P2 at $t = 0$, note how it is morphologically different to A1P5. It shows much more branching and doesn't only grow longitudinally but aims to cover a large area. The blue line shows the main axis and the red circles show bulges that form at division points. They indicate a build-up of cells in the area, preparing for protrusions in the spot. Points a and b are the main sprouting points at both ends of the main axis, while c , d and e are bulges that have formed before the beginning of branch growth.	31
4.6	3D models of A1P5 every 10h. $t=0h$ is normalised to the beginning of day 5. It shows the growth over a period of 30 h. The branches gain in girth early on in development, and don't change much later on. New branching only occurs heavily around the 20 h mark from various points along the branches.	32
4.7	The growth of A1P5 every 10 h. Note the longitudinal growth on the main axis at the beginning, and then focusing more on extending the branches .	33
4.8	Volumetric growth of the inspected organoids from fig. 3.2. The extremely branched D5P2 has the fastest growth, which is evident especially in (b), with a growth of $2.62\% h^{-1}$. The more linearly structured organoids exhibit a slower and more uniform growth (A1P5, A2P2, A1P4).	34

List of Figures

4.9	Surface area development over a period of > 30 h. The differences in growth speed are less evident but still noticeable. There are more fluctuations in the data due to inconsistencies in the surface mesh of the models.	35
A.1	Training dataset montage 1	40
A.2	Training dataset montage 2	41
A.3	Training dataset montage 3	41

List of Tables

2.1	Overview of the advantages of the three preclinical cancer models, adapted from [20]. Especially note the fast expansion, retention of heterogeneity and high-throughput abilities of organoids.	4
4.1	U-Net quality metrics. J is the Jaccard index (Intersection over Union, IoU), with T being the comparison with the thresholded U-net segmentation and RO with the additional Remove Outliers algorithm. p , r , and f_1 give the precision, recall and f_1 score respectively. Metrics from [51] . . .	29

Acknowledgements

First and foremost, I would like to thank my supervisor Fabian Englbrecht for the advice and guidance throughout the whole thesis. He helped me massively in gaining insight to the research area and was always available for questions and open to ideas.

A special thank you also to Prof. Dr. Andreas Bausch for the opportunity to do my thesis in his research group and to the whole research group for the insightful group meetings and friendly working atmosphere.

Also, to my fellow bachelor students Richard Piekara and Matthias Sagerer, who were great fun to work with and go to lunch with.

Finally, my family and friends, who supported and motivated me when I most needed it. Thank you!