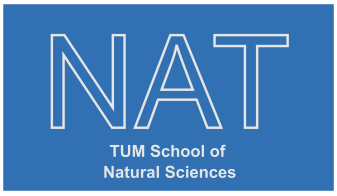

Technical University of Munich
TUM School of Natural Sciences
Center for Functional Protein Assemblies

Optimization of Machine Learning Growth Prediction for Cell Culture Experiments

Matthias Sagerer

This thesis is submitted for the degree of
Bachelor of Science (B.Sc.)

Supervisor: Prof. Dr. Andreas Bausch
Chair of Cellular Biophysics (E27)

Advisor: M.Sc. Fabian Englbrecht

January 23, 2023

Bachelor's Thesis Statement of Originality

I hereby confirm that I have written the accompanying thesis by myself, without contributions from any sources other than those cited in the text and acknowledgments. This applies also to all graphics, figures and images included in the thesis.

Taufkirchen, 23th Jan 2023
Place and Date

M. Sagerer
Signature

Acknowledgments

Supervisor: Prof. Dr. Andreas Bausch
Advisor: Fabian Englbrecht
PhD students: Marion Raich, Sophie Kurzbach
Bachelor Students: Richard Piekara, Sándor Battaglini-Fischer, Daniel Heinrich
Master Student: Sandra Andrusca
The whole Cellular Biophysics Group (E27)
Family and friends

Abstract

In 2020, cancer was responsible for nearly 10 million deaths worldwide, making it one of the leading causes of death. Tumor-derived organoids and 2D cell cultures are used by researchers as model systems to better understand the disease. Due to their ability to mimic *in vivo* characteristics and to reproduce the morphogenesis of complex structures, organoids have the potential to enhance the understanding of tumor structure formation and to enable a more detailed *in vitro* investigation of growth dynamics. Here, growth prediction by machine learning future frame prediction methods can be used as a tool for assistance. By varying different parameters of the dataset used for training and testing of an artificial neural network, the growth prediction for pancreatic ductal adenocarcinoma (PDAC) cells was optimized. The resolution of the training set frames and the time interval between them did not significantly affect the quality of the results when a sufficiently large amount of training data was provided. Most critical for accurate prediction was a large training set and the use of segmented data rather than non-preprocessed brightfield microscopy images. With the exception of the networks recall, data augmentation methods significantly improved the results. An optimized and reliable growth prediction method could be used in many types of cell culture experiments to better understand the development of complex biological systems.

Contents

1 Introduction and Motivation

Although artificial intelligence (AI) had its beginnings already in the 1950s, the field really began to gain traction in the last 25 years [1], [2]. In large part, the increased interest and widespread application of AI is due to the successes of artificial neural networks (ANNs) in a variety of fields, such as autonomous driving [3] or natural language processing (NLP) [4]. They have been made possible by breakthroughs in research, large-scale computing power and an enormous amount of data. It is therefore not surprising that there are also more and more applications of AI in medicine, as large amounts of data are often generated in this science [5].
In medicine, cancer research is an important area of research, as cancer is one of the leading causes of death worldwide. Since there is still no universally applicable cure, cancer was responsible for nearly 10 million deaths in 2020, or almost a sixth of all deaths [6]. There were also just over 19 million new cases in 2020 [7]. Tumor-derived organoids and 2D cell culture experiments serve researchers as model systems. They can be used to increase the knowledge about the disease and to study the effects of treatments [8]. Due to their ability to mimic *in vivo* characteristics and to reproduce the morphogenesis of complex structures, organoids have the potential to enhance the understanding of tumor structure formation and to enable a more detailed *in vitro* investigation of growth dynamics. Here, growth prediction by machine learning future frame prediction methods can be used as a tool for assistance. A reliable growth prediction method for biological cell culture experiments could serve as a digital control group as well. In this way, the development of a treated sample could be compared to the predicted growth of the very same sample without treatment. Thus, the influence of variation between different organoids, which occurs even in a single cell line with the same cell culture protocol, would be eliminated.
For this task, machine learning (ML) methods for future frame prediction can be used. There are already many such methods based on ANNs. With the rise of deep learning (DL) techniques with increased accuracy could be developed [9]. However, the application of them to the problem of autonomous driving, where they are mostly used to predict the behavior of other traffic participants, is most prominent and widespread. In contrast, there is not much research on the use of future frame prediction in biology and specifically for cell culture experiments. For this reason, it is not clear yet how common algorithms can best be adapted for growth prediction in biology. Since data quality and preprocessing strongly influences results in ML, it is necessary for optimization to know which parameters of a given datasets have the greatest impact on prediction accuracy.
In addition to its use in cancer treatment research, a reliable method for growth prediction could be used in multiple types of cell culture experiments to better understand the development of complex biological systems.

2 Theory

2.1 Cell Biology

Microscopy images of cells from a pancreatic ductal adenocarcinoma (PDAC), a type of pancreatic cancer, were used for the growth prediction in this thesis. Pancreatic cancer is the third leading cause of cancer-related deaths in the USA and is expected to be the second leading cause by 2030 due to the rapidly aging society. In part because PDAC specifically is insensitive to many chemotherapeutic drugs, it is among the most lethal human cancers [10]. This is one of the reasons why PDAC is responsible for more than 90% of all pancreatic malignancies [11]. Due to an insidious and nonspecific clinical presentation, more than 50% of patients present at an advanced state with metastatic disease. Only a minority of patients are suitable for potential curative resection. Due to remote metastasis, survival rates for PDAC remain low even after surgical treatment. Up to 80% of patients who undergo surgical resection recur within 2 years and as many as 90% are expected to recur within 5 years [12], [13].

2D cell culture: By using 2D cell culture methods, the characteristics and behavior of various cancer cells can be studied in a controlled laboratory setting [12]. This helps researchers understand the mechanisms underlying PDAC. Various techniques, such as immunofluorescence staining, can be used to investigate the cells in greater detail. It is also common to treat samples with different drugs or agents in order to study their effect on them. In the experiments for this thesis, the cells were placed in a coated coverslip with two wells under adapted medium supplementation so that they can be better microscoped later. They are then placed in an incubator or microscope compatible incubator chamber for ideal growth conditions. The growth of the cells in a monolayer finally results in the 2D cell culture. The experimental procedure used for this thesis is explained in more detail in section 3.1.1. The cells grow so fast that growth can be seen after only a few hours, which makes them well suited for growth prediction over different time periods. It is much more complicated to work with image data of 3D organoids, because they are either videos with three spatial dimensions or the different microscopic layers have to be projected onto a two-dimensional surface in a useful way. In the latter case, difficulties arise in the automated segmentation of the data due to blurring and shadows. Among other reasons because of this, data from a 2D cell culture was used for the growth prediction in this thesis. Next, 3D PDAC organoids are briefly described, as they are relevant for a possible extension of the prediction to the three-dimensional case, which will be a part of the discussion in section 5.

3D organoids: Three-dimensional cell culture methods are considered more representative of the *in vivo* environment [12]. 3D PDAC organoids develop through self-organization highly branched structures displaying a seamless lumen connecting terminal end buds. They replicate the in vivo PDAC architecture. Previously cooled liquid collagen is used in the experimental procedure and polymerizes in the incubator. This embeds the cells in a collagen matrix. Based on organoids, the identification of distinct morphogenesis with unique patterns of cell invasion, matrix deformation, protein expression and respective molecular dependencies is possible *in vitro* [14].

2.2 Machine Learning

When was the first time humans thought about whether machines could think? This a question without a clear answer. A well recognized point in time, when this questions was picked up was in Alan M. Turing's paper *"Computing Machinery and Intelligence"* [1]. It marked the beginning of a period of initial efforts to build machines that could make intelligent decisions. Frank Rosenblatt built the first artificial neuron, also called perceptron, in 1958 [2]. This is still an important building block of artificial neural networks (ANNs), but in digital form and not in analog form as it was at that time. The branch of computer science that deals with the demonstration of intelligence by machines is called artificial intelligence (AI). A subfield of AI that has gained importance in the last two decades and has delivered increasingly better results is called machine learning (ML). In the following, the most relevant ML backgrounds for this thesis are briefly explained.

2.2.1 Machine Learning Fundamentals

Higher order categorization: In the field of ML, the different algorithms fall into one of the following four categories: reinforcement learning (RL), supervised learning, unsupervised learning or a mixture of supervised and unsupervised learning, such as semi-supervised learning [15]. There are also algorithms that cannot be unambiguously assigned to one category because they have features from several categories. However, all of them follow the same principle. A system learns a certain behavior from a training dataset or experience, which can be obtained in reality or in a simulation. Subsequently, it is able to generate an output for a given input, which is similar to the training data or experience.

Reinforcement learning: RL is based around an agent in a specific environment. A prominent example of RL as its application on 49 Atari games such as Pong, Video Pinball or Ms. Pac-Man [16]. Let's take as an example the video game Pong. The goal is to deflect a ball with a paddle and get it behind your opponent's paddle by moving yours up and down. The game roughly simulates table tennis. A screenshot of the came is depicted in figure 2.1. The agent is the paddle controlled by the ML system in the environment of the game screen. It can sense certain things from the playing field,

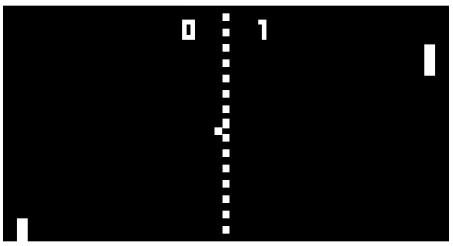

Figure 2.1: The video game pong can be well played by machine learning powered systems. The game roughly simulates table tennis with the goal of getting the ball to the end of the opponent's court to score a point. Both players have a paddle, which can be moved up and down to deflect the ball. An agent of a system with artificial intelligence can learn by reinforcement learning, a form of machine learning, to play this video game. It is trained to achieve the highest possible score. [17]

including the distance of the paddle to the top edge and the x and y distance of the ball. The agent starts with an exploring phase, where it displays random behavior. Given a specific goal, such as scoring a point or avoiding a point for the opponent, the agent learns which behaviors are desirable given a certain state of the environment. Thus, the agent gains experience, through which he can make better decisions in the future. These experiences can take place in "reality" by having the ML system compete against a human player. However, it is also possible for the ML system to learn the game in a simulation, for instance through playing against a pre-programmed player. RL has been implemented in different ways over the years, often in combination with other ML techniques such as deep learning (DL).

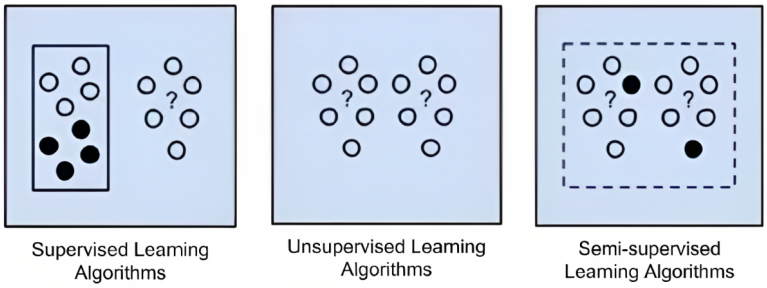

Figure 2.2: The majority of machine learning methods can be classified as supervised, unsupervised or semi-supervised learning. The groups differ in the availability of labels associated with the data points in the dataset used. In the case of un-/supervised learning, there is a training set (left in the images) that is not/completely labeled. In both cases, trained models should correctly assign labels to the test set (right). In semi-supervised learning only a part of the dataset is labeled. [18]

Supervised learning: To use supervised learning, you need a fully labeled dataset. A classic example is a dataset consisting of images of cats and dogs. Every image has a label associated to it, which tells whether the picture shows a cat or a dog. The dataset is split into a training set and a test set. A common split would be 80% of the total dataset for the training set and the remaining 20% for the test set. Now, a ML model learns to differentiate between a picture of a dog and a picture of a cat using the training set. Afterwards members of the test set are presented to the model without their labels. It classifies these images according to the classes "cat" and "dog". Since the correct labels for the test set, which are also called ground truth (GT), are known, they can be used to determine the accuracy of the classification. The described properties of a dataset for supervised learning as well as those of a dataset for unsupervised and semi-supervised learning are depicted in figure 2.2. Growth prediction as presented in this thesis is an example of supervised learning.

Unsupervised learning: For supervised learning there is no need for labels. In the example above the pictures of cats and dogs would be the same, but wouldn't have the label attached to them. A ML model now has the task of extracting useful information from the dataset. It might be able to successfully split the dataset into one cluster containing only images of cats and another one containing only images of dogs, based on the different appearances of both animals. However, it will not be able to assign a label to each of the two clusters. The model "recognizes", so to speak, that the dataset contains images of two different entities, but does not "know" which entities they are. A more practical sample application for unsupervised learning would be to find out the consumption behavior of certain customer groups from a supermarket's sales data.

Semi-supervised learning: An example for a form between supervised and unsupervised learning is semi-supervised learning. It can be used for a dataset which is partly, but not entirely labeled. Usually the number of labeled data points is much smaller than the one of unlabeled ones. Just like with unsupervised learning, an ML model can split the dataset into two cluster. The difference here is that it is also able to guess what the labels of all the unlabeled images in the clusters are based on the labeled images. Since it is often difficult to get a large amount of labeled data, semi-supervised learning is a good alternative to supervised learning. There are also other mixed forms, such as self-supervised learning [18].

An artificial neural network is a ML method that has become very popular in recent years due to the fact that this type of model has achieved very good results. Besides this method, there are many others, for instance regression algorithms, decision trees or dimensonality reduction. The focus of this thesis will be on ANNs, since this type of model was used for the growth prediction. In the literature on this topic, it is often referred to neural networks and neurons without calling them artificial. Therefore, this convention will also be used in this thesis. ANNs and artificial neurons are nevertheless to be distinguished from biological neural networks consisting of biological neurons.

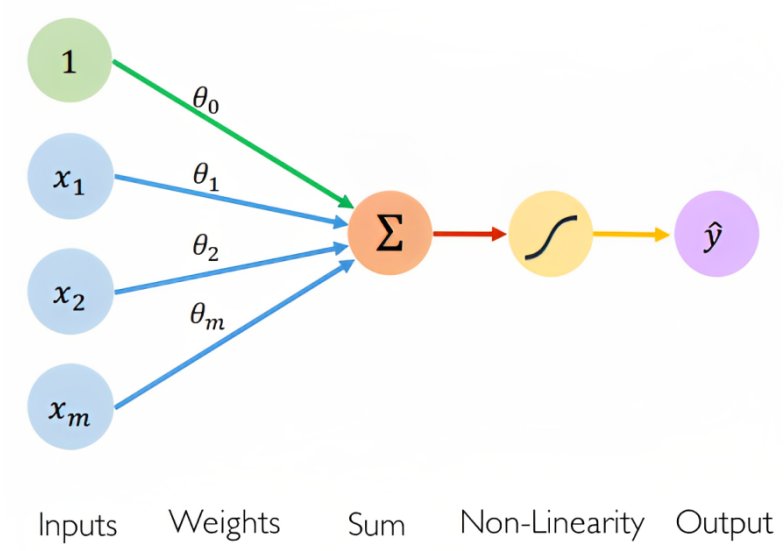

Figure 2.3: An artificial neuron, also called a perceptron, performs a mathematical operation. The inputs x_i weighted by their weights θ_i and the bias θ_0 are summed. The output of the perceptron $\hat{y}$ is obtained by applying a nonlinear function to the sum. [19]

Perceptron: Just as a biological neuron is the basic building block of a biological neural network, a perceptron is the basic building block of an ANN. How the perceptron works is inspired by a biological neuron. Nevertheless, there are further, more complex behaviors in the biological case, which are not reproduced in the computer. Therefore, an artificial neuron should be understood purely as a mathematical function and not as a model of a biological neuron. Its functionality is shown graphically in figure 2.3 and the mathematicaly function it performs in equation 2.1.

$$\hat{y} = g\left(\theta_0 + \sum_{i=1}^{m} x_i\,\theta_i\right) \tag{2.1}$$

First, the inputs x_i, which have their origin either directly in the input of the network or in the outputs of other neurons, enter the perceptron in the form of numbers. Each of them is multiplied by an associated weight θ_i that is determined or learned during the training of a network. Then a perceptron specific bias θ_0 is added to the product. On the resulting sum a so-called activation function $g(x)$ is applied, which is typically nonlinear, to produce the final output $\hat{y}$ of the artificial neuron [20]. Examples for an activation function are the heaviside step function and sigmoid function. Nowadays, various forms of rectified linear units (ReLUs) are very widespread. In addition to the classic ReLU, there is also the leaky ReLU (LReLU), for example [21].

Dense layer: Let's go back to the example of determining whether a picture shows a dog or a cat. For a computer, a digital color image consists of many numbers. It is

divided into many pixels according to its resolution and for each of them three numerical values are stored, which are assigned to the colors red, green and blue (RGB). These values indicate the intensity of the respective color to create the required color of the respective pixel by color mixing. For the classification of an image with an ANN, exactly these numbers are the inputs for the network. In it, the neurons are typically arranged in layers. The first layer, which directly receives the image, is called input layer. The last layer, which finally outputs the result of the classification, is called output layer. All layers in between are called hidden layers because they are not directly visible when a network receives data and returns a result.

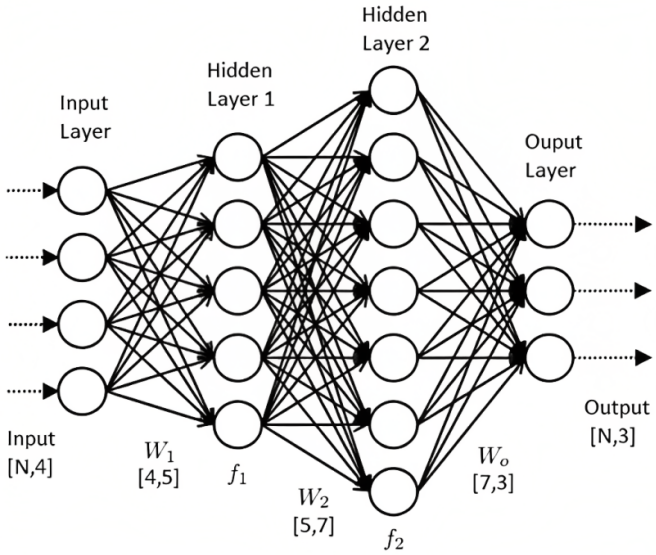

Figure 2.4: An artificial neural network consists of many artificial neurons, so-called perceptrons, arranged in layers. In the input layer, this network receives a four-dimensional input vector, computes mainly in the hidden layer, and the output layer finally produces the three-dimensional output vector with the weights matrix W_0. The intermediate representation vectors f_1 and f_2 are calculated with the weight matrices W_1 and W_2. [18]

In a dense layer, each neuron receives the output of all neurons of the preceding layer as input and its output is passed on to all neurons in the next layer. Thus, each neuron in this type of layer is connected to each neuron in the layer before and after it, but neurons within a layer share no connections. A graphical representation of a network with two dense layers is shown in figure 2.4.

$$\vec{y} = g(M^{\text{number of inputs x number of neurons}} \cdot \vec{x} + \vec{b}) \tag{2.2}$$

The mathematical function of a single neuron is now extended to a whole dense layer (eq. 2.2). As input, the layer now receives a vector $\vec{x}$, which contains all outputs of the precious layer. The weights of the individual connections of a layer are now arranged

in a matrix M, where a row corresponds to a neuron and a column to an input. This is reflected in figure 2.4 and equation 2.2 in the dimensions of the respective matrix. Following a matrix vector multiplication of $\vec{x}$ and M, another vector $\vec{b}$, which contains the biases, is added to the result. After applying the activation function $g(\vec{x})$ element by element to the resulted vector, the output vector $\vec{y}$ is obtained, which serves as the input vector for the next layer. This function is repeated with the associated weight matrices and bias vectors for all layers, until one finally arrives at the output layer and with this the output for the network is determined [22].

Convolutional neural network (CNN): When a color image, which has a three-dimensional data structure, is taken in the form of an input vector from a dense layer, it is reduced to a one-dimensional data structure. Therefore, the information about which three numerical values belong to one pixel and which pixels are next to each other in the image is lost. This does not happen with a convolutional layer. An ANN that uses one or more convolutional layers in its architecture is called a convolutional neural network (CNN) or ConvNet. Furthermore, for a network with exclusively dense layers, the number of weights, also called parameters, to be learned increases with the size of the images which is accompanied by the need for a lot of computing power. A convolutional layer only contains one neuron whose parameters are arranged in a three-dimensional volume, also called filter or kernel. One with a kernel with a height, width and depth of three is depicted in figure 2.5. The kernel is assigned to the pixels in a corner and then moves row by row over the entire image. At each position, a numerical value of the inputs is assigned to each parameter and multiplied by it. The products are then summed, an activation function is applied to the sum and the result becomes part of the layer's output. Thus, a three-dimensional input resulted in a two-dimensional output. However, often a layer contains several filters. Outputs of the respective filters are then layered again to a three-dimensional output. The depth of these outputs no longer represents the color channels red, green and blue, but the different filters, while the width and height still roughly reflect the width and height of the original image.

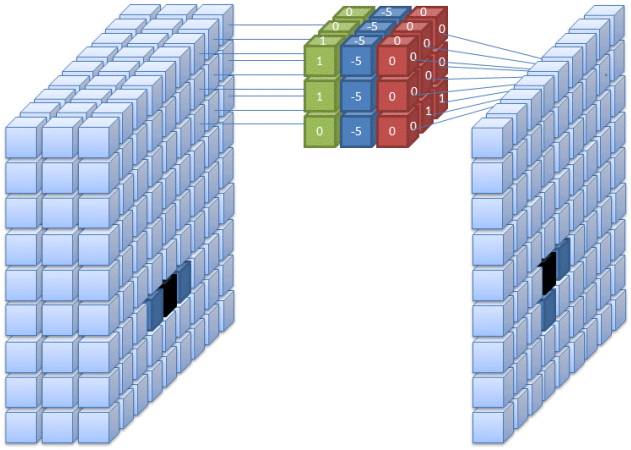

Figure 2.5: A convolutional layer performs a mathematical operation called convolution. A three-dimensional filter, also called kernel, with parameters (middle) runs over the input (left side) to produce the elements for the output (right side). At each position the products of the filters parameters with the respective input value are summed to produce the output value. This type of layer is commonly used in neural network for image and video processing tasks for the extraction of certain features of an image. These features are at different levels of abstraction, from edges, for example, to entire objects, such as a table. [23]

Since in a convolutional layer a number equal to the height times the width of the kernel of input values becomes a single output value, so-called zero-padding is often used. In this case, a frame consisting of zeros is added to the input before processing. Thus the size of the input volume in the dimensions height and width is reduced by the convolutional layer less or not at all. The width of the frame determines how little the size is reduced. The stride value describes how many pixels a filter jumps at a time when sliding over the image. This value and in CNNs often used pooling layers, which will not be further explained here, also influence the resizing in width and height during processing [24] [25]. When a CNN is used to classify images or something similar, after some convolution (and pooling) layers, usually some fully connected layers are used in the architecture, which then output the final output. A well-known example of this is the LeNet-5 from the paper "Gradient-Based Learning Applied to Document Recognition" by LeCun et al. (1998) [26]. Its architecture is depicted in figure 2.6. In the application of ANNs to image and video data, CNNs have now become widely used due to very good results. Since this type of application is intended to teach the computer to develop a kind of sense of sight to understand what is displayed on images and in videos, this subarea of ML is also called computer vision (CV).

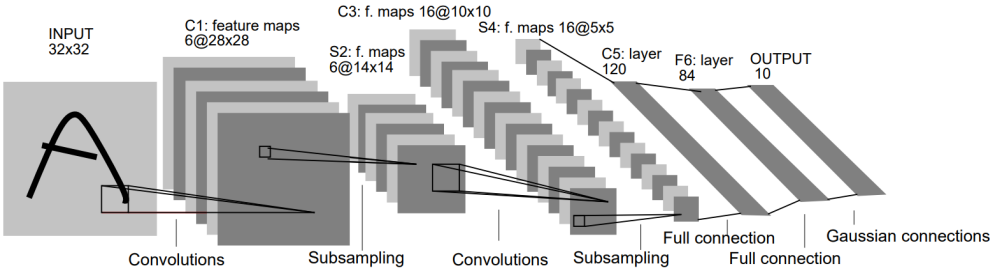

Figure 2.6: The convolutional neural network LeNet-5 recognizes handwritten characters. With a 32 px × 32 px image as input it computes ten probabilities that the image contains a certain character. Its architecture consists of multiple convolutional, subsampling, fully connected and Gaussian connected layers. Each of them detects features of the input at different levels of abstraction. [26]

Gradient-based learning: As can be seen form the title of LeCun's 1998 seminal paper [26], it dealt with gradient-based learning. It was the first application of this learning method, which became widely known and thus triggered many further developments. More than two decades later, gradient-based learning is still a central and essential idea in the application of ANNs. Initially, the weights of neural networks were set manually. However, this approach did not catch on, because it was too difficult to develop target-oriented weight matrices and filters for CNNs. With gradient-based leaning, a network is able to learn all its parameters based on many examples. In the learning process, a distinction is made in the flow of information between the forward-pass and back-propagation, for both of which further mostly nearly equivalent terms exist. First, all parameters are initialized with random values, although there are also more special initialization methods, which are based, for example, on the fact that the occurrence of the values correspond to a distribution, such as the Gaussian normal distribution. This is followed by the first forward-pass. In our example of classification, an image is given to the network as input. Now the individual neurons in the dense and convolutional layers compute the respective layer inputs until finally values are determined in the output layer. In the case of binary classification into dog and cat images, this would result in two outputs, each of which would indicate in a decimal number between 0 and 1 the probability whether the input image contains a dog or a cat. Now, the loss-function is calculated. This measures the error of the predicted output values $y_{pred,\,i}$ compared to the actual or target values y_{GT}, which are referred to as ground truth (GT). One example is the $\mathcal{L}_2$ loss or least squared error calculated according to equation 2.3. The total loss of a output vector is the sum of the individuals losses of all N elements of the vector (eq. 2.4)[27].

$$\mathcal{L}_2(y_{pred,\,i}) = (\; y_{GT} \;-\; y_{pred,\,i}\;)^2 \tag{2.3}$$

$$\mathcal{L}_2(\vec{y}_{pred}) = \sum_{i}^{N} \mathcal{L}_2(y_{pred,\,i}) \tag{2.4}$$

As the ANN is trained on the training data, the goal is to minimize the total loss of each input. Since all parameters were used for the calculation of $\vec{y}_{pred}$ besides the input, they can be seen as arguments of the loss function. For a given input, the set of parameter values that leads to the global minimum of the loss function is to be found. However, it is not possible to calculate this global minimum analytically. Therefore the method of gradient back-propagation is used. Here, the gradient of the loss function with the parameters as argument is calculated. The chain rule is used, because each layer produces as output the input of the next layer. With the negative gradient all parameters are changed individually in the direction that the loss is smaller with a recalculation. How far the parameters are changed is influenced by the gradient and the learning rate, a hyperparameter of the network. This process is now repeated to further reduce the loss with each iteration and is called gradient descent. There are other more complicated types of parameter update such as AdaGrad, AdaDelta or RMSprop [28].

Deep learning: Deep learning (DL) is a development in ML that started to gain momentum in the 2010s. It deals with deep neural networks or deep neural nets (DNNs). Deep means here that the networks have an architecture with high depth, i.e. many layers. DNNs are thus able to form multiple layers of abstraction. In a concrete case, this can mean that the first layer of a net for image classification recognizes edges with a certain orientation in the image, the second layer pays attention to a certain arrangement of edges, and the third layer is able to identify familiar objects, such as an eye. Further on, together with other features, the head of a cat can be recognized and finally a cat as a whole. The meanings of the abstraction layers are not introduced manually during the design of the DNN, but learned by gradient back-propagation. Therefore one speaks of representation-learning methods. DL has already been successfully used in a wide variety of applications, such as speech recognition, CV or even drug discovery and genetics. Deep CNNs can be applied to speech and audio in addition to their use in CV [29].

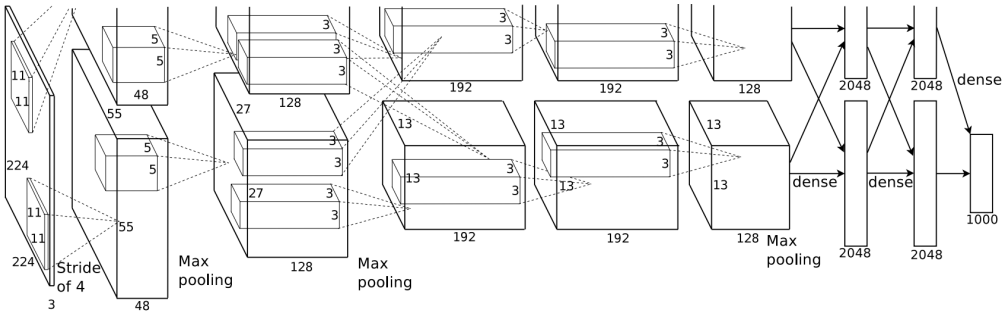

Figure 2.7: The deep convolutional neural network AlexNet classifies images into 1000 classes based on the object depicted by them. Its architecture is split into two parts for the training on two graphical processing units. One tunes the layer-parts above and the other those below, while both communicate with each other only at certain layers. Besides convolutional layers, max pooling and dense layers are also utilized in the architecture. [30].

A prominent example of a DNN is AlexNet. It is a deep CNN, which has 650,000 neurons and 60 million parameters. These are arranged in five convolutional layers, some with max-pooling layers, and three fully connected layers, which finally generate a 1000-dimensional output vector. The network was trained to classify the 1.2 million images of the LSVRC-2010 contest into the 1000 different classes. It was specifically designed to be trained on two graphics processing units (GPUs), which is well illustrated by its architecture depicted in figure 2.7 [30]. With DL, the explanation of the ML fundamentals most relevant to this thesis is now complete. In the next section, methods used for future frame prediction are explained.

2.2.2 Machine Learning Methods for Future Frame Prediction

Problem definition: First, the task at hand is formally defined as follows: The basis of the prediction is a video sequence consisting of n frames. With this as input of the method m frames are to be predicted, which continue this video sequence. Future frame prediction thus builds on a representation learned from input frames, while video generation is mostly unconditioned. During the learning process, the GT target frames serve as labels, which is why it seems that the problem can be associated with supervised learning. However, since human-labeling is not necessary, it is a self-supervised task. Assuming that good prediction requires accurate representation, learning by prediction can be seen as an approach to how well a system has learned the patterns underlying the input data [9].

Application of future frame prediction outside bioinformatics: Perhaps most prominent is the application of video prediction to autonomous driving. To control autonomous vehicles (AVs), the trajectories of pedestrians must be predicted [31]. Considering other traffic participants, future frame prediction can be used to predict the future semantics, geometry and motion of complex real-world urban scenes in traffic. The obtained

representation which does not need to be in pixel space, can be used to control AVs [32]. Widespread use of autonomous driving has the potential to reduce traffic accidents and improve mobility in crowded areas. For this purpose, the safety of these real-time systems must be the highest priority in their development [33]. Other applications are for example in robot navigation or human-machine interaction [9].

An overview over several categories of methods can be found in source [9]. Since a direct pixel synthesis method was used to obtain the results of this work, only methods of this type are explained below. In this category there are several different methods, which often combine a couple of techniques used in ML. Convolutional layers often serve as the basic building blocks of DL architectures for visual reasoning. Therefore, CNNs represent the foundation of the visual prediction literature. The receptive field of convolutional operations is limited by the kernel size. Since classic CNNs lack the capacity for explicit inter-frame modeling, 3D convolutions are often used to capture temporal consistency. Below, recurrent neural networks (RNNs) and generative neural networks (GANs) are explained as two of the most used ANN architectures for video prediction. However, there are also other models for video prediction such as autoencoders (AEs) like the conditionally reversible network (CrevNet) [34]. It has a reversible architecture to build a bijective AE. It serves as both an encoder and decoder. Since it uses 3D convolutions, it is a 3D convolutional encoder-decoder (3D-cED).

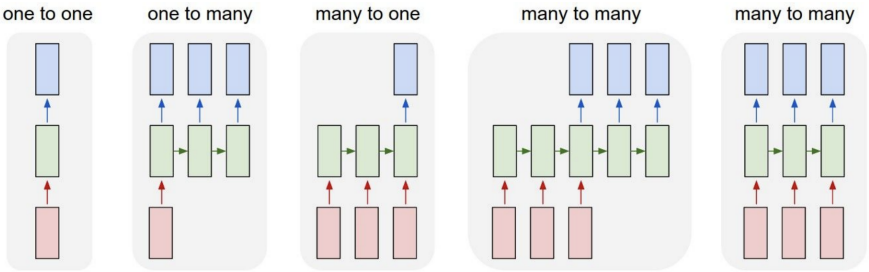

Figure 2.8: Different types of recurrent neural networks can read and return sequential data of different lengths. Hidden layers (green) are applied to the input vector (red) one or more times with the same parameter values. They pass data to the output vectors (blue) and, if it exists, to the next instance of the hidden layers. [35]

Recurrent neural networks: In general, ANNs receive an input of a fixed size and produce an output vector that has a fixed size as well. RNNs are designed to work on sequential data, such as video, speech or text. Therefore, they are able to input and output data of variable size, as shown in figure 2.8. This is achieved by applying the same hidden layers (green) with the same parameters multiple times. From the first to the second to last instance of the network, a hidden state is passed on to the next instance. A certain number of the first instances additionally receive a part of the total input (red), while a certain number of the last ones output a part of the total output [35]. RNNs for video prediction are specially designed to model spatio-temporal data. Using classical

RNNs for this purpose brings problems like vanishing or exploding gradients. This leads to limitations of long-term representations for videos. These can be solved by using more sophisticated models such as long-short term memory (LSTM) or gated recurrent units (GRUs) [9]. Example of RNNs for future frame predictions are the recurrent CNN (rCNN) from Ranzato et al. [36] and the Znet [37], which is a convolutional LSTM RNN. An example of a model with a more exotic RNN architecture is the folded recurrent neural network (fRNN) [38]. It is an AE with shared states between encoder and decoder using bijective gated recurrent units (bGRUs). Since the architecture of this practically half folded AE also uses convolutional layer, it can be called convolutional gated recurrent units autoencoder (cGRU-AE).

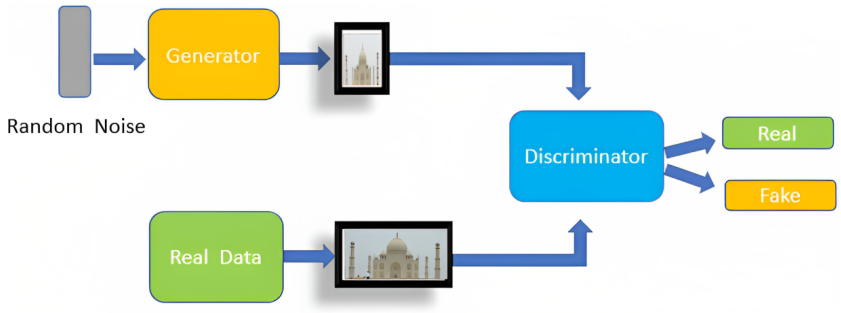

Figure 2.9: In a generative adversarial network, a generator and discriminator are trained simultaneously. The generator generates an image from random noise, or from an input such as a frame sequence in the case of future frame prediction. The goal of the training for the discriminator is to correctly distinguish between generated and real data, which are passed to it alternately. The generator is trained to fool the discriminator. Typically such a type of network is used to generate realistic data similar of the training data. [39]

Generative adversarial network: A generative adversarial network (GAN) is a type of ANN that learns to generate outputs through an adversarial process. Two models are trained simultaneously: a generative model or generator and a discriminative model or discriminator. The former captures the data distribution of the training data and creates samples to reflect it. The latter receives a sample from the training data or from the generator and estimates the probability whether the sample comes from the data or was created by its generative counterpart. This basic concept of a GAN is depicted in figure 2.9. The discriminative model is trained to be correct in its estimate, while the generator is trained to maximize the probability that the discriminator's estimate is false. Therefore, a GAN is based on game theory rather than optimization like other generative model approaches. If both parts of the model are realized by multi-layer percetrons, the whole system can be trained with back-propagation [40]. Convolutional layers can also be used in their architecture. GANs are among the best generative models and produced good results for many different applications (mostly in a research environment). They are especially known for their ability to produce realistic high-resolution images of for

Figure 2.10: Progressive growing generative adversarial networks are able to generate photorealistic images. Growing the generator and discriminator during training speeds up and stabilizes the training process. The depicted 1024 × 1024 images were generated by a model trained on the CALEBA-HQ dataset, which contains images of celebrities. [43]

example faces [41]. An example of using a GAN for future frame prediction can be found in source [42]. It was used to predict the future development of traffic scenes.
In a progressive growing GAN (PGGAN), both the generator and the discriminator grow progressively during the training process. So it is started with a small resolution, such as 4 × 4, and finer and finer details are modelled over time by adding more layers. This is done, for example, by doubling the number of pixels in width and height. Using PGGANs instead of GANs speeds up and stabilizes the training. As a result, images of previously unattainable quality could be generated. Some examples are shown in figure 2.10. The network used in this thesis, called FutureGAN, is an example of a PGGAN and is explained in the next section.

2.2.3 Explanation of the FutureGAN

The FutureGAN [44] is a GAN model that predicts frames of a video sequence conditioned on a set of previous frames. All frames are generated simultaneously. During training, the FutureGAN receives only the raw pixel values of the frames as input. This makes it easy to adapt to various different datasets and still achieve stable results, which was one of the reasons to use it for PDAC growth prediction.

Architecture: While GAN-based models have been used for video prediction before, FutureGAN's utilization of the concepts of the existing PGGAN model [43] is new. It extends the basic PGGAN concept used for the generation of single images to the task of future frame prediction. Here it also helps to avoid an unstable training process. In addition, collapse effects, which cause the generator to learn to fool the discriminator by producing samples of a limited set of modes, are avoided. Furthermore, the FutureGAN is an encoder-decoder GAN model that uses spatio-temporal 3D convolutions to capture both the spatial and temporal components of a video sequence in its internal representation. The generator, which is trained to predict the sequence of future video frames, and the discriminator, which is trained to distinguish between generated and real video

sequences, are trained at the same time. The latter receives alternately generated and real sequences and assigns them a decimal number between 0 (generated/fake) and 1 (real).

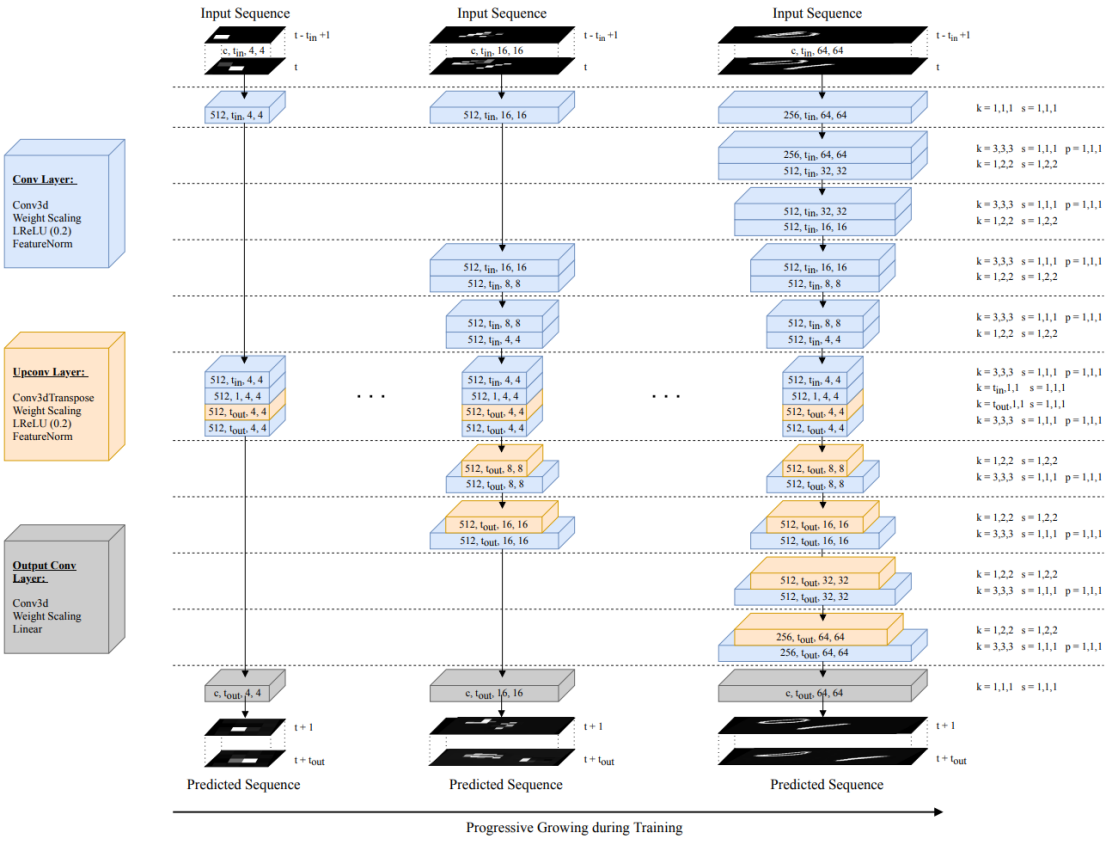

Figure 2.11: During training, FutureGAN's generator network grows progressively. The model is initialized with the same resolution of 4 × 4 for the input and predicted frames. After a specific number of iterations, this resolution if doubled by adding four convolutional layers. Therefore, the network grows progressively. The image illustrates the growth from 4 × 4 px to 64 × 64 px while the steps for 8 × 8 and 32 × 32 are left out for visual clarity . [44]

Generator Network: The generator reads the input frames $\boldsymbol{z} = (\boldsymbol{x}_{t-t_{in}+1}, \ldots, \boldsymbol{x}_{\boldsymbol{t}})$ and transforms them into the output frames $\tilde{\boldsymbol{x}} = G(\boldsymbol{z}) = (\tilde{\boldsymbol{x}}_{t+1}, \ldots, \tilde{\boldsymbol{x}}_{t+t_{out}})$. t_{in} and t_{out} describe the length of the input and output sequence respectively. The generator network consists of the encoder in the upper part of figure 2.11, which learns the representation of the input, and the decoder in the lower part, which generates the prediction. As can be seen, the encoder mirrors the decoder except that the spatial up-sampling layers are replaced by spatial down-sampling layers. To match the number of input frames t_{in} to the number of output frames t_{out}, the bottleneck layers in the middle of figure 2.11 perform temporal up- and down-sampling. The whole process starts with an input and

output resolution of 4×4, which is progressively increased during training after a specific number of iterations by adding two convolutional layers each to the en- and decoder, thus doubling the resolution. A LReLU is used as a non-linear activation function.

Discriminator Network: The architecture of the discriminator network is very similar to that of the encoder part of the generator except for the bottleneck layers. An important difference is that there is no pixel-wise feature vector normalization layer that limits the signal magnitudes in the generator. To prevent a mode collapse, a mini-batch standard deviation layer was added as one of the last ones. The final output of the discriminator is generated using a dense layer with a linear activation function.

More details about the generator, discriminator and the training procedure can be found in the FutureGAN publication [44]. This paper also describes experiments conducted on three datasets: MovingMNIST, KTH Action and Cityscapes.

2.3 Machine Learning in Bioinformatics

Bioinformatics is an interdisciplinary field that deals with the development of methods for retrieving, analyzing, storing and organizing biological data. Objectives include understanding gene functionality along with cell regulation, drug target selection, drug design and disease identification. Often, quantitative analysis is necessary to interpret the enormous amounts of data generated by biological systems [45]. Since ML algorithms such as ANNs require large amounts of data for successful training, bioinformatics is predestined for the use of such methods. Besides network-based approaches, ML techniques for clustering, prediction classification or embedded methods can also be utilized. These are applicable to problems such as gene expression clustering, patient classification, brain network analysis and identification of biomarkers [46]. More unusual applications are also conceivable. The clustering of protein sequences has already been combined with text mining of bioinformatics literature to extract protein-protein interactions [47]. But arguably the best-known and perhaps most groundbreaking ML system in the branch of bioinformatics in recent years is AlphaFold [48][49], developed by Google's sister company DeepMind.

AlphaFold: Determining the three-dimensional structure of proteins from their amino acid sequence by prediction has been an important research problem for more than 50 years. This is mainly because a proteins 3D composition largely determines its function. Through huge experimental efforts, the 3D structure of about 100,000 unique proteins could be determined, typically by X-ray crystallography or nuclear magnetic resonance (NMR) spectroscopy [50]. However, these represent only a small fraction of the billions of known protein sequences because it is often difficult to determine a protein structure experimentally. It takes months to years to determine a single one. AlphaFold is a deep learning algorithm with the goal of predicting the 3D shape of a protein based solely on its amino acid sequence. The model was trained using the over 100,000 known

structures of the protein data bank (PDB) [51]. Genetic information is leveraged by analyzing homologous sequences. This is done by one of the two representations used by this method, the multi sequence alignment (MSA). The second is a distogram (distance histogram) which encodes a probability distribution for the distance between two residue pairs. By gradient descent, an ANN is trained to make a prediction about these distances. AlphaFold can regularly predict protein structures with atomic accuracy even in cases where no similar structure is known. It thus resulted in increased accuracy for protein structure prediction, which could enable new insights into the function and malfunction of proteins [48][49].

Growth prediction: Biology deals with the living and consequently by definition with things that grow, which can be individual animal or plant cells, but also entire trees. Therefore, growth prediction has the potential to be used in virtually any subfield of bioinformatics. Natural plant growth cycles can be extremely slow. DNNs have the potential to predict the expected growth of plants by predicting future leaf and roots systems. This could enable ML-based high-throughput phenotyping and thereby accelerate experimental research cycles involving the quantitative assessment of anatomical, biochemical and physiological plant traits. Results to date show strong performance with the capacity to match expert annotation in the future [52].
Furthermore, tumor growth has also already been predicted with CNNs by Zhang et al. [53]. Their method demonstrated higher accuracy than previous mathematical models, which can lead to better results in tumor treatment management and surgical planning. A convolutional LSTM in the spatio-temporal domain (ST-convLSTM) was used to extract a tumor's static imaging appearances, capture its temporal dynamic changes and learn the longitudinal dynamics from multiple patient studies. Both cell density and CT intensity number could be predicted.
Already before and also during the writing of this thesis at the Chair of Cellular Biophysics (E27) [54] at the Technical University of Munich PhD student Fabian Englbrecht [55] and master student Sandra Andrusca [56] also worked on ML based growth prediction for biological cell culture experiments. However, they worked with 3D organoids. As a basis for their prediction they use microscopy data that have been segmented into binary black-and-white frames so that only organoid and non-organoid pixels are distinguished. Sandra Andrusca used an input sequence with three different representations: a binary mask, a growth map and an optical flow representation. With these, a CNN extrapolates the prediction of the growth. Fabian Englbrecht, on the other hand, used the FutureGAN. Besides the prediction of untreated organoids, they both also make some for cell cultures after a treatment to compare them with the actual development of the organoids after the application of the agent.

With this short overview of the use of ML in bioinformatics and especially in growth prediction, the theory part is now finished. In the next chapter, the methods used to obtain the results of this thesis will be described.

3 Methods

This chapter explains the experimental techniques and evaluation methods that were used to collect and process the data. All the code can be found at in the *GitHub* repository at `https://github.com/MatthiasSagerer/growth-prediction`.

3.1 Experimental Techniques

In the following, the experimental techniques used to acquire the data of the growth of PDAC cells will be described. First the cell culture and then the microscopy will be explained in more detail.

3.1.1 Cell Culture

The PDAC cells used at our lab originate from the 9591 mouse of Prof. Dr. med. Dieter Saur's group at the *Klinikum rechts der Isar*. For this thesis, cells were taken from an existing cell line and were grown in flasks in cell culture. When a flask is removed from the incubator, where the cells grow at 37°C under a 5% CO_2 atmosphere in ideal growth conditions, for inspection, they are examined for their cell density. If it is observed that the cells in the center of cell clusters are small and round, this is an indication that the cell density is too high. In this case, not all cells have sufficient access to nutrients of the medium anymore. This means that they need to be split, which is done about every 2-3 days. Depending on the specifications of the cell line and the experiment, they may need to be split less or more frequently. In the splitting process, also called passaging, the cells are transferred with new medium into one or more new flasks to reduce the cell density. Some of the cells may be discarded during this process. The new medium added during passaging is *Dulbecco's Modified Eagle's Medium (DMEM) Mixture F-12* from *Sigma*. It was supplemented with 10% *Fetal bovine serum (FBS)* growth serum and 1% of *Penicillin/Streptomycin (P/S)* antibiotic, which were also both from *Sigma*. Even if the cell density is not too high, the medium must be changed every two days to provide sufficient nutrients in order to ensure cell growth. However, changing the medium only does not influence the number and density of the cells. A low nutrient concentration can be recognized, for example, by the fact that the otherwise reddish cell-medium mixture turns orange. For imaging, the cells were seeded in a *2 well µ-slide coverslip* from *ibidi* at a density of $2 \times 10^5 \frac{cells}{well}$ together with the supplemented medium.

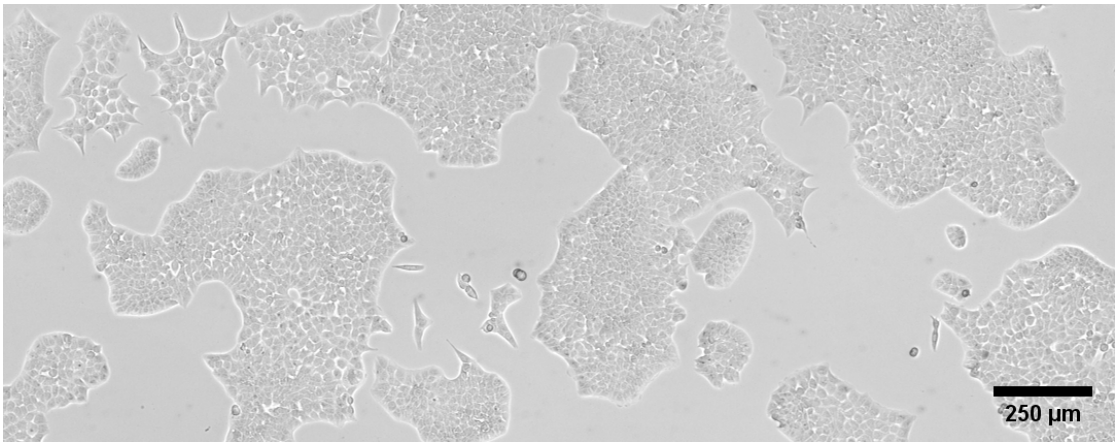

Figure 3.1: Pancreatic ductal adenocarcinoma cells grow in a 2D cell culture. A section of a microscope image with a size of 2808.0 μm × 1104.4 μm is depicted. It was cut out of the first frame of the video of the left well.

3.1.2 Microscopy

For imaging, the coverslip with the cells was placed in a *Tokai Hit* live-cell microscope incubation system. In this system, the cells continue to have ideal growth conditions with 37°, 100% humidity and a 5% CO_2 atmosphere during microscopy. A *Leica Thunder Imager Live Cell* microscope with the *Leica Application Suite X (LAS X)* software was used for imaging. It was equipped with a *N PLAN 5x/0.12 PH0*, a Leica air objective with 5× magnification and a numerical aperture of 0.12. Once per hour, both wells were automatically scanned by the microscope by passing over 64 positions per well and stitching the individual images together. Brightfield microscopy was used to capture the images and the entire process took approximately only one minute for both wells. Each hour, the stitching process resulted in an image with a resolution of 7446 × 7440 per well. The images are scaled at 0.3846 $\frac{pixels}{\mu m}$, which means that a whole image covers an area of 19359.62 μm × 19344.02 μm. Thus, most of the area of a well, which has dimensions (w × l × h) of 21.6 × 23.8 × 9.3 mm^3, was pictured. For 3D organoids, this methods is extended to the third dimension by additionally mapping different planes at each position by changing the focus. Thus, instead of one image, a z-stack of images is generated for each point in time. The growth of the 2D cell culture used for the experiments in this thesis was microscoped over a period of 78 hours. This generated two *Leica Image Files (LIFs*, each containing 78 7446 × 7440 images. They were then converted to the *Tagged Image File Format (TIFF or TIF)* for further processing. A small cutout of the first frame of the left well is depicted in figure 3.1.

3.2 Data Preprocessing & Postprocessing for Dataset Parameter Variations

An exploratory approach was taken to optimize the ML growth prediction for cell culture experiments. By changing several parameters of the dataset used for prediction, their effects on prediction accuracy were analyzed. In the first part, the number of videos and frame resolution in the training set of the FutureGAN model were varied simultaneously. In the second part, the time intervals between images and the preprocessing of the dataset with respect to segmentation were varied. Finally, in the last part, the influence of two different data augmentation methods on the prediction accuracy was investigated. In this section, the methods used for preprocessing and postprocessing for the three different dataset variations are explained. The dataset processing here was mainly done with a *Microsoft Surface Laptop 2*, which has an *Intel Core i5-8250U @ 1.60 GHz* central processing unit (CPU) with 4 cores and 8 GB of random-access memory (RAM). Most of it was done with *Python (3.9.12)* scripts, but some steps in preprocessing were done with the *ImageJ* image processing software. The execution of the *Python* scripts took mostly only a few seconds to a few minutes and in rare cases a little over 5 minutes. Particularly computationally intensive tasks in ImageJ were run on a PC with an *Intel Core i9-10900X @ 3.70 GHz* CPU with 20 cores and 64 GB RAM. As a result, each individual processing step took less than a minute for the entire data set.

3.2.1 Training Set Size and Frame Resolution

For this part, an already preprocessed dataset of the microscopy images was utilized. During his preprocessing, the images of each well were split into a 10×10 grid, resulting in a total of 200 videos. Here, each cutout has a resolution of 660×660, so a total resolution of 6600×6600 was used. Due to shadows at the edges of the microscope images, not all their resolution could be used. For the 200 videos, some of the first and last frames couldn't be used due to spatial incoherence and only every third frame was held, resulting in videos with 25 frames and a 3h time interval between frames. The FutureGAN, as used for this thesis, works with videos of 12 frames For this reason, when training and testing the models, each video was automatically split in two. The dataset has also already been divided into a training set with 180 videos and a test set with 20 videos, and all frames were segmented into binary images. An example of a video sequence containing 12 frames is depicted in figure 3.2. The white areas always represent the cells and the black areas the background. In the first row are the first six and in the second row are the last six frames, ordered from left to right. For the segmentation during preprocessing a procedure almost identical to the one described in section 3.2.2 was used.

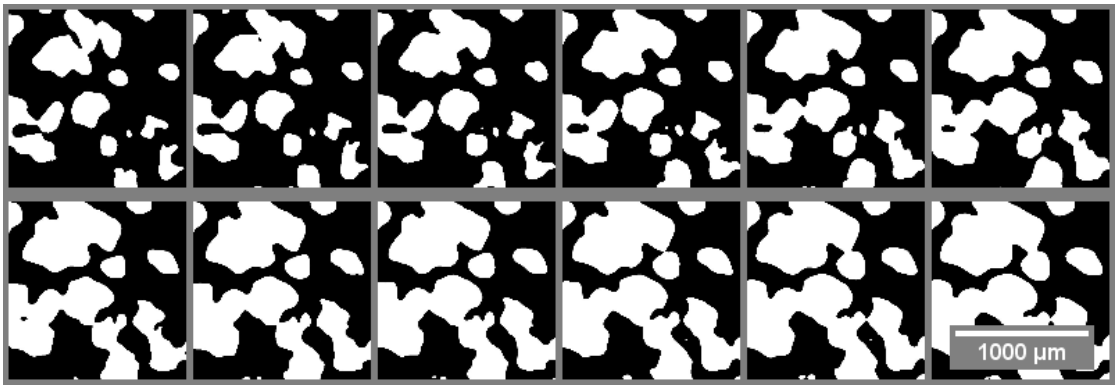

Figure 3.2: Each video sequence that the FutureGAN works with contains 12 frames. This is one example sequence from the dataset for the training set size and frame resolution variation. The white areas represent cells and the black areas represent the background. The top row shows the first half and the bottom row the second half of the frame sequence. They are ordered from left to right with a gap of 3 hours between each two frames.

Preprocessing: Four copies with 18, 45, 90 and 180 videos of the training set of the dataset described above were created. This represents 10%, 25%, 50% and 100% of the available training set. This division was chosen because of the expectation of diminishing returns for the improvement of the prediction as the size of the training set increased. The largest change was suspected to be in the range between 10% and 50%. With the chosen sizes, the effects of training set size on prediction accuracy could be better resolved in this range. Of each of these four copies another four copies were made. For three of these per training set size, i.e. a total of 12 training sets, the individual frames were downscaled to the resolutions 128×128, 256×256 and 512×512. This resulted in 16 different training sets with four different numbers of videos and four different frame resolutions. These resolutions were chosen because the FutureGAN, as used in this thesis, only produced frames with a resolution of 128×128 and it was assumed that the prediction would only improve with an increase in the resolution of the frames in the training set. Due to the expectation of diminishing returns as in the training set size case, with the exception of the jump from 512×512 to 660×660, the resolution was also progressively increased. All the preprocessing was done with *Python*. The models trained with the 16 training sets were all tested on the same test set. This was the test set from the dataset, which was already received preprocessed, downscaled to a 128×128 resolution.

Postprocessing: During testing of the models, they predicted six output frames for six input frames. Although you could distinguish very clearly between the black and white areas in these output frames, they were not binary images, but RGB images. However, because of this clear difference, it was very simple to segment the images. For this, the *Python* function depicted in figure 3.3 was used.

```
1   import numpy as np
2   import cv2
3   def simpleSegmentImg(img_input):
4       img_copy = img_input.copy()
5       lower_white = np.array([127, 127, 127], dtype='uint16')
6       upper_white = np.array([255, 255, 255], dtype='uint16')
7       thres_mask = cv2.inRange(img_copy, lower_white, upper_white)
8       img_copy[thres_mask > 0] = (255, 255, 255)
9       img_copy[thres_mask < 1] = (0, 0, 0)
10      return img_copy
```

Figure 3.3: A rather simple *Python* function was used to segment the majority of the real and predicted frames during postprocessing. This function was used for the results of the models trained on segmented data, since they were comparable easy to segment. In line 4, a copy of the input image is created to work with. In line 7, a mask is created with the upper and lower bounds for the white areas set in lines 5 and 6. Then, in line 8, all pixels values inside this range are set to white (values for red, green and blue equal to 255, the highest possible value), and in line 9, all values outside this range are set to black (value 0 for all color channels). Finally, the segmented image copy is returned. The function uses the *OpenCV (cv2)* and *NumPy* library.

In line 4, a copy of the input image is created. In lines 5 and 6, threshold values for white are defined. The mask for segmentation is created in line 7 and applied to the input image copy in lines 8 and 9. Finally, in line 10 the segmented image is returned. In the function the two libraries *NumPy* and *OpenCV (cv2)* are used.

Limitations: The range of the frame resolution was limited upwards to 660 × 660. Thus, no prediction could be made with frames of higher resolution. One reason why higher resolutions were not used was that this one corresponded to the resolution of the already preprocessed dataset. The dataset was also not regenerated from the microscopy images with higher resolution, as this would have resulted in fewer videos. The choice of range of resolutions of the training set allowed to examine their influence and that of the training set size on the prediction within reasonable boundaries. Since the number of videos generated is indirectly proportional to the square of the pixel width of a generated frame, there is a trade-off between these two parameters. Increasing the number of videos by microscoping more samples would also be a possibility. Since this would have meant a considerable increase of working time and effort due to cell culture and microscopy, this was not done. In addition, the usage time of the labs and microscopes is shared among many Bachelor, Master and PhD students as well as postdoc researches in the research group [54].

3.2.2 Time Interval and Segmentation

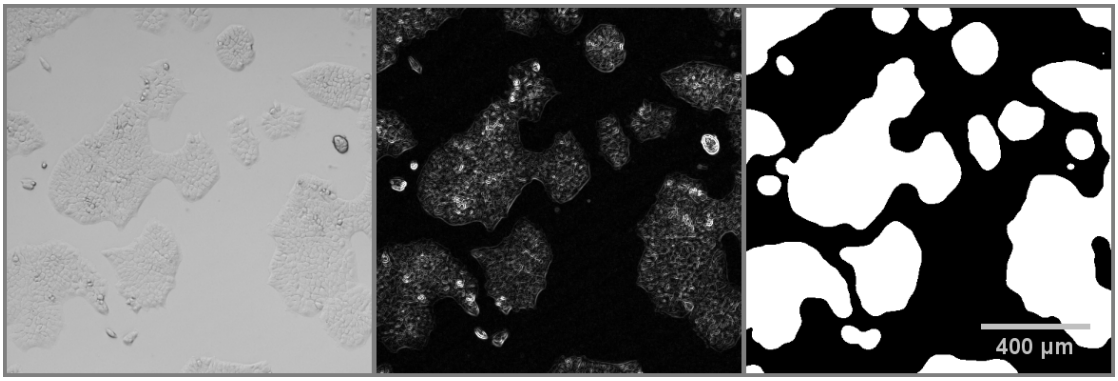

Figure 3.4: Three different levels of preprocessing were used for training. From left to right: original, edge detection, segmented. Each frame shows the same area of 1331.25 μm × 1331.25 μm at a resolution of 512 × 512.

Preprocessing: The microscope data after a drift correction, also called registering, was applied to them was used as a starting point for this dataset variation. The images were cropped to a resolution of 7168 × 6656 because parts of the edges were not usable. This corresponds to (14x512) × (13x512) and thus 14 columns and 13 rows of cutouts with a resolution of 512 × 512. Next, the first 6 frames were deleted because they contained a jump between two frames, had partially bad lighting conditions, and 72 frames were more suitable for later processing. Afterwards, they were segmented. This and the steps described so far were done in *ImageJ*. First, edge detection was applied to the whole image stack (one per well), followed by a bandpass filter using fast Fourier transform (FFT). This filtered large/small structures down/up to 400/5 pixels with a directional tolerance of 5% and simultaneously scaled them automatically with saturation. Afterwards, the images were processed with a Gaussian blur with radius 5 to reduce noise.
A copy was made after each processing step of the segmentation. Three datasets were used for further processing: the registered original data with the target resolution and number of frames, the data processed with edge detection and the fully segmented images. An example of a frame in all three processing stages is depicted in figure 3.4. This selection was made for the following reasons: The original data should serve as a control to evaluate the prediction of an almost unprocessed dataset. In the edge detection images, it is easier to distinguish between the cells and the background, but they still contain information about individual cells. The best distinction between the cells and background can be made in the segmented binary images.
Next, four data sets were created from each of the three datasets with time intervals of 1h, 2h, 3h, and 4h between the frames. From each of these, 182 videos were created with 12 images per well. The images of each of them were rotated once by 90 degrees counter-clockwise and the unrotated as well as the rotated videos were used. The process of generating more data from the available data by changing it in some way is called

data augmentation. Through this data augmentation method, 728 videos with 12 frames could be obtained for each time interval. Since according to the time interval (except for 1h) a certain number of the 72 frames were not used, between one and six videos could be produced per cutout. Thus, depending on the number of hours, a different number of cutouts had to be used to generate the same number of videos. Table 3.1 lists how many videos could be generated per cutout, how many columns and rows of the 14 × 13 grid were used and how many of the videos from those cutouts had to be omitted to get to 182 videos per well before using data augmentation.
All 12 datasets were finally divided into 644 training and 84 test videos. This roughly corresponds to a split of 88.5 : 11.5.

Time Interval	Videos per Cutout	Columns Used	Rows Used	Videos Omitted
1h	6	8	4	10
2h	3	9	7	7
3h	2	13	7	0
4h	1	14	13	0

Table 3.1: The number of available videos for training and testing per cutout depends on the time intervals between two frames. The microscopy images were divided into a grid of 14 columns and 13 rows of cutouts with a resolution of 512 × 512. For the time intervals of 1h, 2h, 3h, and 4h a different number of columns were needed to be used to acquire the the same number of videos for each variation. Additionally, some of the videos of the first two time intervals needed to be omitted.

Postprocessing: Since the prediction results with the original data were of such a bad quality, it made no sense to try to segment them. The predictions with the segmented data were segmented in the same way as those of the dataset size and frame resolution variation (see figure 3.3). However, when segmenting the frames produced by the models that worked with the edge detection data, a more sophisticated *Python* function had to be used. The code of this function is shown in figure 3.5. A copy of the input image is first scaled up from 128 × 128 to 512 × 512. Then *OpenCV*'s canny edge detection, dilation, Gaussian blur and thresholding are applied. Finally, the image is scaled back down and converted to a binary image with only 0 and 255 as pixel values before it's returned.

```python
import cv2
import numpy as np
def segmentImg(img_input):
    img_copy = img_input.copy()
    dil, can_l, can_u, gauss, thrs_min, thrs_max  = 2, 12, 20, 37, 0, 255
    kernel_dil = np.ones((dil, dil), np.uint8)
    img_res = cv2.resize(img_copy, (512, 512))
    img_canny = cv2.Canny(img_res, can_l, can_u)
    img_dilate = cv2.dilate(img_canny, kernel_dil, iterations=3)
    img_gauss = cv2.GaussianBlur(img_dilate, (gauss, gauss), 0)
    ret, img_thres = cv2.threshold(
        img_gauss, thrs_min, thrs_max, cv2.THRESH_BINARY+cv2.THRESH_OTSU)
    img_seg = cv2.resize(img_thres, (128, 128))
    img_seg[img_seg > 192] = 255
    img_seg[img_seg < 192] = 0
    return img_seg
```

Figure 3.5: For the segmentation of the real and predicted frames from the models trained with the edge detection datasets a more sophisticated *Python* function was used. Lines 4 to 6 contain basic setup: creation of an image copy to work with, assigning of variables for the parameters of following functions and the initialization of a kernel for the dilate method. In line 7, the image is upscaled for better processing. The main processing with the canny edge detector, dilation, Gaussian blur and thresholding happens in lines 8 to 12. In the following three lines the image is downscaled back to 128 × 128 and converted into a binary image. Finally, the processed image is returned in line 16.

Limitations: The range of the different time intervals was limited downwards to 1h, as this corresponded to the time between the frames of the microscopy data. Upwards, it was limited to 4h due to the training time of the networks. For the same reason, more types of preprocessing of the frames were also not used. Adding more types of image preprocessing or time intervals would have resulted in three or more additional datasets. The successful training of a model took about one working day and sometimes even several days due to hardware difficulties, which for example required an adjustment of the training script. Therefore, an extension of the investigated variations of the dataset could have quickly caused weeks of additional training time. Another limitation of the experiment was that the measured prediction accuracy depends on the accuracy during the segmentation in pre- and postprocessing. Since the segmentation of the edge detection results was particularly difficult, it was performed reasonably well, but not as good as for the other results and thus does not completely truthfully represent the prediction and GT data.

3.2.3 Data Augmentation

Preprocessing: For the last part, four datasets were generated to investigate the effect of data augmentation on prediction accuracy. All of them contain segmented frames with a resolution of 512×512 and three hours between each two frames. The segmented dataset from section 3.2.2 before further processing into the individual datasets was used as the starting point. Thus, the microscope images were again divided into a grid with 14 columns and 13 rows. The datasets differ as follows:

- **Control:** All 14 columns and 13 rows were used in this dataset.
- **Rotated:** Here, with 7 columns and 13 rows, half of the available data was used. Each video was also added to the dataset as a 90 degree counter-clockwise rotated copy.
- **Mirrored:** Also 7 columns and 13 rows were used, but this time a copy mirrored on the vertical axis was additionally added to the dataset.
- **No augmentation:** Finally, this dataset only consists of 7 columns and 13 rows from the available data.

Due to the fact that two videos with 12 frames were generated per cutout and well, the first three datasets consist of 728 (training set: 644, test set: 84) and the last one of 364 (training set: 322, test set: 42) videos. All models were tested on the control test set. However, in order to simulate the use of one dataset by itself, all four were divided into a test and training set and only the latter was used for training.

Postprocessing: Since only segmented data was used for training and testing, the segmentation of the predicted frames were very simple and could be done again using the *Python* function from figure 3.3.

Limitations: The decision to use mirroring and rotation of the frames as means of data augmentation was made because they are among the easiest to apply and thus adapt. They could have been applied more than once, but there is no obvious reason to expect that a second, or third application would have a different effect than the first. Implications about the effects of advanced data augmentation methods, such as frame interpolation, cannot be derived from the used research approach. However, the effects of specifically rotation as a method of data augmentation are most relevant to this thesis, since it was the only one used in the previous dataset variation.

3.3 Application of the FutureGAN

Google's cloud computing service *Google Colab* was used to train the models for the dataset size and frame resolution variation. There, the training was performed on a *Nvidia Tesla V100* with 16 GB RAM using *Python 3.8*. The same platform was used for a

part of the second variations data before switching to a PC, also running *Python 3.8* with 2 *Nvidia GeForce RTX 2080 Ti* with 12 GB of RAM for the training of the remaining datasets, including those for the data augmentation. The datasets were organized into a test and train folder, each containing the videos as folders. In each video folder are the respective frames as *jpg* files. The FutureGAN was trained using mostly the default settings of the training file from the *GitHub* repository [57]. However, when training with the dataset with 1h between the frames and original images (no edge detection of segmentation), a batch size of 8 instead of 16 had to be selected for the 4×4 resolution due to GPU memory limitations. Checkpoints were created for the dataset size and frame resolution variation after 20 epochs and for the rest after 40 epochs. Their creation frequency was reduced, because the checkpoints need a lot of storage and only served as a backup, but never had to be used. During the testing of the models, the GT frames, which were later used for comparison during the evaluation, were automatically downscaled to 128×128, since the FutureGAN produced frames with this resolution. Due to a bug, the last video of each test set was not tested when testing the FutureGAN models. Thus, the results of the first variation were tested on 39 instead of 40 and those of the other two on 83 instead of 84 videos. More details about the training and testing procedure can be found in the description and code of the FutureGAN repository on *GitHub* [57].

3.4 Analysis Procedure

All predictions were analyzed by using four metrics: Intersection over Union (IoU), predicted cell area normalized to GT, precision and recall. They quantify under different aspects how well the predicted frames match the GT frames. After segmentation, the GT and predicted videos were stored in separate folders. The analysis was automated with *Python* on the *Microsoft Surface Laptop 2*. Figure 3.6 shows the rough structure of the scripts schematically.

Dataset size and frame resolution: The predictions of the first variation were evaluated by using four scripts, which have the same structure and differ mainly only in the applied metric. The repeated execution of a code block for a certain number of passes is made possible with a so-called for loop. It was looped over a nested list of 32 directories corresponding to the four dataset sizes, four frame resolutions and GT and predicted frames. The first loop was over the dataset sizes, the second over the frame resolutions, the third over the individual videos and the last over the frames. The predicted frame and corresponding GT frame were read in and then the value of the metric was calculated.

Time interval and segmentation: Eight scripts, four for the edge detection and four for the segmented datasets, were used. This time, it was looped over a nested list of 16 directories. While the first loop ran over the different time intervals between the frames,

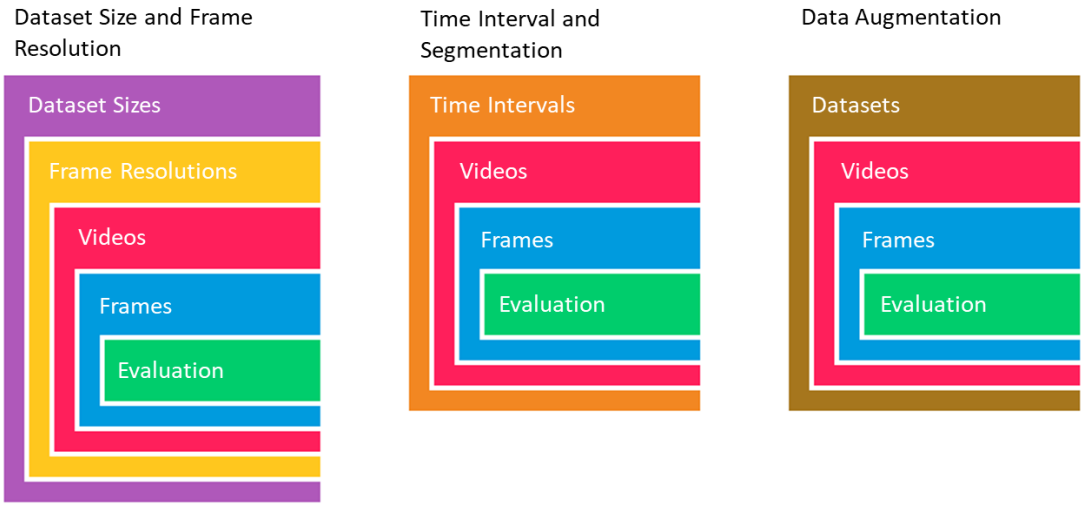

Figure 3.6: The predictions of the three different variations were evaluated using *Python* scripts with three types of sequential structure. Every colored shape, except the outermost, represents a similar block of code that gets repeated for each element of the category named in the surrounding block. The three innermost blocks represent the automatic evaluation of all the predicted frames of all videos of a specific dataset and follow the same structure for all three variations. On the left, the two outermost blocks represent that for each dataset size, the datasets with different resolutions are run through one after the other. For the second variation, the datasets with different time intervals between frames were processed one by one. With the script for the data augmentation variation, the predicted frames of the datasets with different data augmentation methods and of the control datasets were evaluated in sequence.

the other two match the inner two for the dataset size and frame resolution variation, so that the respective metric for the individual frame pairs could be calculated again.

Data augmentation: For the last part, only one script was used to calculate all metrics. It looped over a nested list of eight directories, corresponding to the GT and predicted frames of the ”Control”, ”Mirrored”, ”Rotated” and ”No Augmentation” dataset. The first loop went over these datasets, while the other two nearly matched the two innermost ones of the dataset size and frame resolution variation. This time, however, all metrics were calculated directly for each frame pair.

With each script the metric values were written to an excel file. Then the data from the excel files were read into *MatLab* to create the figures.

4 Results

4.1 Evaluation Metrics

In figure 4.1, six predicted frames are compared with the corresponding GT frames. Each pixel is assigned to one of four categories (tab. 4.1).

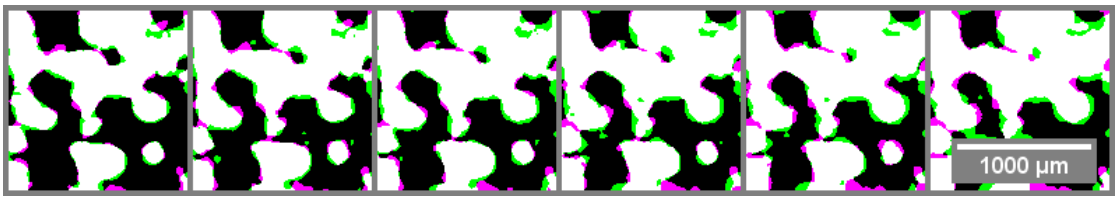

Figure 4.1: The comparison between six predicted frames and the respective real frames shows where too much or too little growth was predicted. For the black (true negatives) and white (true positives) areas, the prediction matches with the ground truth. The green (false positive) areas correspond to incorrectly predicted growth. Areas where growth actually occurred but was not predicted are colored in magenta (false negative). The images were generated from the predicted and real frames of a video from the control dataset for the data augmentation variation.

Positives correspond to pixels with cells, while negatives correspond to pixels without cells, i.e. only the background. True positives (TP, white) are correctly predicted cells and false positives (FP, green) are incorrectly predicted cells. Pixels of the category true negatives (TN, black) correctly and those of the category false negatives (FN, magenta) incorrectly contain only the background.

Ground Truth	**Prediction**	
	Positive (P)	Negative (N)
Positive (P)	True Positive (TP)	False Negative (FN)
Negative (N)	False Positive (FP)	True Negative (TN)

Table 4.1: Pixels of the predicted frames are classified into one of four categories. Those that contain cells are considered positive, while pixels that contain only the background count as negative. Using the number of pixels in each category for a given frame, the metrics for that frame can be calculated to measure how well the prediction matches the real frames.

Four different metrics were used to evaluate the predictions. Each quantifies in a different way how well the predicted frames match the GT frames. The intersection of the predicted cell area and the GT cell area divided by the union of the the two areas gives the Intersection over Union (IoU). The IoU (eq. 4.1) can also be calculated using the TP, FP and FN. It always has a value between zero (no match at all between prediction and GT) and one (prefect match). Next, the predicted cell area was normalized to the GT cell area (eq. 4.2). The precision (eq. 4.3) is a measure of how many of the predicted pixels with cells contain a cell in the GT frame. It therefore indicates what proportion of the predicted growth was correctly predicted. Recall (eq. 4.4) measures how much of the GT growth was predicted. It is calculated as the ratio of the number of TP to the sum of TP and FN.

$$IoU = \frac{TP}{TP + FP + FN} = \frac{intersection}{union} \tag{4.1}$$

$$normalized\ predicted\ cell\ area = \frac{TP + FP}{TP + FN} = \frac{predicted\ cell\ area}{ground\ truth\ cell\ area} \tag{4.2}$$

$$precision = \frac{TP}{TP + FP} \tag{4.3}$$

$$recall = \frac{TP}{TP + FN} \tag{4.4}$$

4.2 Dataset Parameter Variations

In this chapter, the evaluations of the growth prediction accuracy for the different dataset variations are presented. For the training set size and frame resolution variation, the training time of the respective models was also examined.

4.2.1 Training Set Size and Frame Resolution

All networks for this variation were evaluated with the same test set, which contains 20 videos with 25 frames each. Since FutureGAN worked with sequences of 12 frames, it used the first and subsequent 12 frames of each video as a separate sequence for training and testing, while the 25th frame was not used. Due to a bug, predictions were only made for 39 of the possible 40 videos. The videos of the test set have a resolution of 128×128 to match the resolution of the prediction. Because the test set and training set only contained segmented frames, the predicted frames look similar to the ones for the segmented data for the time between frames and level of segmentation depicted in on the right side in figure 4.6. One can see that the predicted frame is paler than the corresponding GT frame.

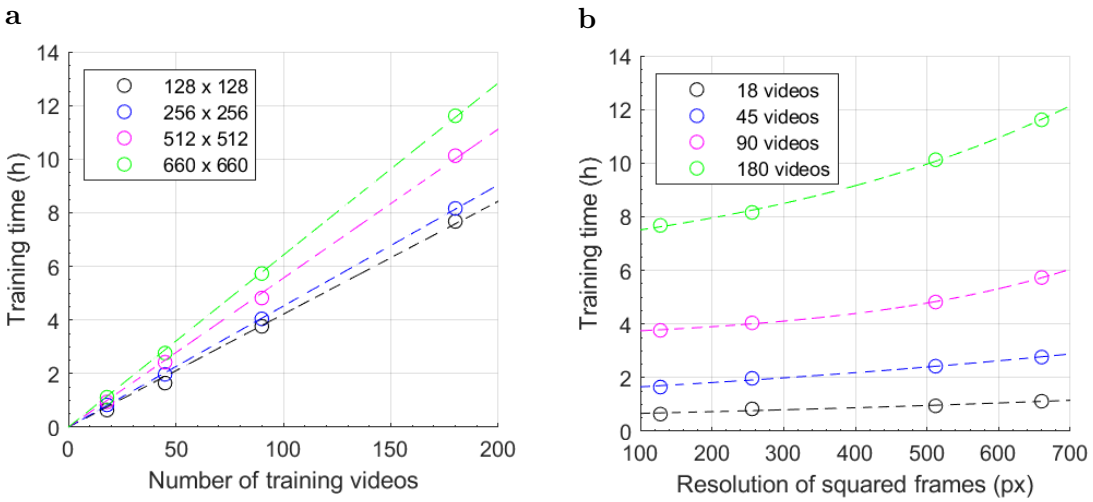

Figure 4.2: The training time for the FutureGAN model changes with number and resolution of the training videos. a With up to about eleven hours difference, the training time increases significantly with training set size. Fitting equation: $a \cdot x$ (128 px: a = 0.042; 256 px: a = 0.045; 512 px: a = 0.056; 660 px: a = 0.064). **b** The increase in training time with frame resolution is significantly smaller for the selected range. Fitting equation: $a \cdot \exp(b \cdot x) + c$ (18 videos: a = 0.61, b = 0.0010, c = 0; 45 videos: a = 1.5, b = 0.00092, c = 0; 90 videos: a = 0.030, b = 0.0031, c = 3.3; 180 videos: a = 1.7, b = 0.0020, c = 5.5)

Training times:

A simple *Python* script was used to measure the training times plotted in figure 4.2. Figure 4.2**a** shows the change of the training time with the increase of the number of videos in the training set for different frame resolutions. Since each video contains 25 frames, it is important to note that each one was used as two videos by FutureGAN, as for testing. Figure 4.2**b** shows the change with frame resolution for different training set sizes. Training times increased significantly more with an increase in the size of the training set than with an increase in the resolution. For the smallest training set, it increased from the smallest to the highest resolution from 38m 21s to 67m 9s. This corresponds to an increase of 75.1%. For the largest training set, it increased from 7h 40m 33s to 11h 36m 51s, an increase of 51.3%. This means that the training time increased by 1100.9% at the lowest resolution and by 937.8% at the highest.
However, the progression for change with the dataset size was best fitted with linear graphs, while that for the frame resolution was best fitted with exponential graphs. For the latter, a linear and quadratic fit were also tested. This type of growth for the respective parameter would mean, for values above the examined range, that the training time increases significantly faster for higher resolutions than for larger training sets. However, a higher resolution and number of training videos is only desirable up to a certain point. There is far more room for reasonable increase in training set size

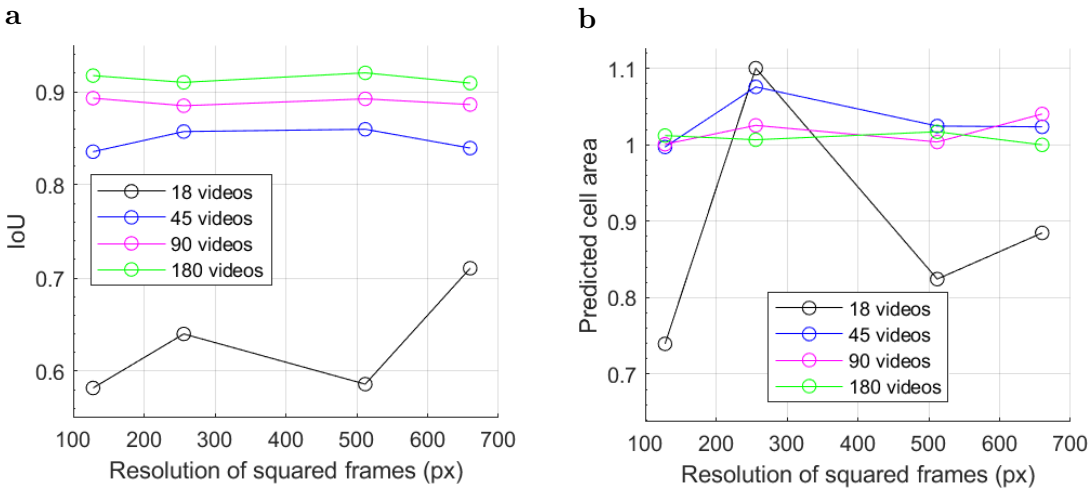

Figure 4.3: The average metric values for the prediction accuracy change with training set size and resolution. Tables 7.1 and 7.2 list the plotted values and associated standard deviations. **a** Intersection over Union (IoU). **b** Predicted cell area normalized to the ground truth cell area.

than resolution. More importantly, from the available data, it is not possible to conclude with certainty that training time grows exponentially with frame resolution, since the difference from linear and quadratic growth is not significant enough in the considered range.

Growth prediction accuracy:

In figures 4.3 and 4.4, the changes in the metric values with the frame resolution are plotted for the different training set sizes. Figure 4.3 depicts the IoU and the predicted cell area normalized to the GT cell area. For both, one is a perfect value, but unlike the IoU, the cell area can be greater than one. Figure 4.4 shows the values for the precision and recall, which, like the IoU, take on values between zero and one with the latter as the optimal value. The mean standard deviations of the results within the test set are 0.0878 for the IoU, 0.0963 for the cell area, 0.0552 for the precision and 0.0723 for the recall. Individual standard deviations are listed in tables 7.1, 7.2, 7.3 and 7.4 along with the associated metric values.
The results for the smallest training set fluctuate a lot, show no obvious trends and also have the highest standard deviations across the test set. For this reason they should be treated with caution. In order to get a better impression of the underlying trends, if there are any, the corresponding networks would have to be evaluated with larger test sets. However, it is also possible that this training set size is not large enough for meaningful results. As expected, all results improve on all metrics with larger training sets, but shows diminishing improvements. In general, by far the biggest improvement is from 18 to 45 videos, which is an increase from 10% to 25% of the available data.

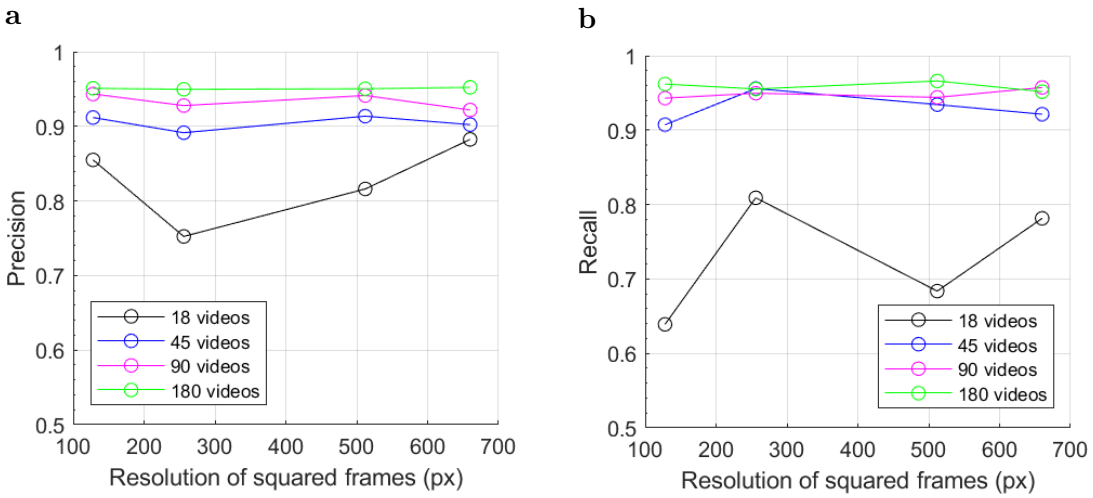

Figure 4.4: The average metric values for the prediction accuracy change with training set size and resolution. Tables 7.3 and 7.4 list the plotted values and associated standard deviations. **a** Precision. **b** Recall.

There are a few outliers to this trend: the cell area for 256 × 256 and precision for 128 × 128 and 660 × 660 for the training set with 18 videos are close to the corresponding values for larger training sets. However, since these are significantly better than the ones for the same dataset size but lower or higher resolution, they're likely the result of randomness. To confirm or deny the validity of these improvements, further experiments would have to be made. Almost all values for the three larger training sets overlap within their standard deviations. However, between the training set with 45 videos and 180 videos there are still improvements of up to 0.0817. The change in metric values with resolution is by far the largest for the smallest training set, but no obvious trend can be seen as all values sometimes improve and other times worsen with higher resolution. For the larger training sets there are no significant changes with frame resolution, especially considering the standard deviations. As with the small training set, there is no clear trend present in these small changes, since all metrics for all training set sizes worsen and improve at least once with greater resolution.

4.2.2 Time Intervals and Segmentation

For each training set, an individual test set was used to evaluate the respective model. This was necessary because each dataset (training set and test set) differs from the others either in the time interval between frames, the degree of segmentation or both. It was assumed that a model could not be successfully applied to a test set that differs in this way from the training set of the model. All test sets included 84 videos with 12 frames, but due to a bug, one video was not used for testing and thus predictions were only made for 83 videos. The frames of the test sets have a resolution of 512 × 512 like the ones in the training sets. However, they were automatically downscaled to 128 × 128

by the FutureGAN during testing to get the same resolution as the predicted frames.

Training set and test set with original frames

Figure 4.5 depicts GT frames in the first row and the corresponding predicted frames in the second row. The first predicted frames are shown. The columns differ in the time intervals between the frames in the dataset used for the prediction. From left to right: one hour, two hours, three hours and four hours. The prediction tends to get better with longer time intervals between the frames. For one hour, no structure corresponding to the cells in the GT frame is visible in the prediction. This improves to the four hour prediction where a structure similar to the GT cell area is present. However, the frames predicted for four hours are very difficult to segment correctly. This is mainly because there is no clear boundary between the cells and the background: Both smoothly blend into each other. For this reason and because the models for the other time intervals made very poor predictions, no quantitative analysis of the results from the models trained with original data was conducted.

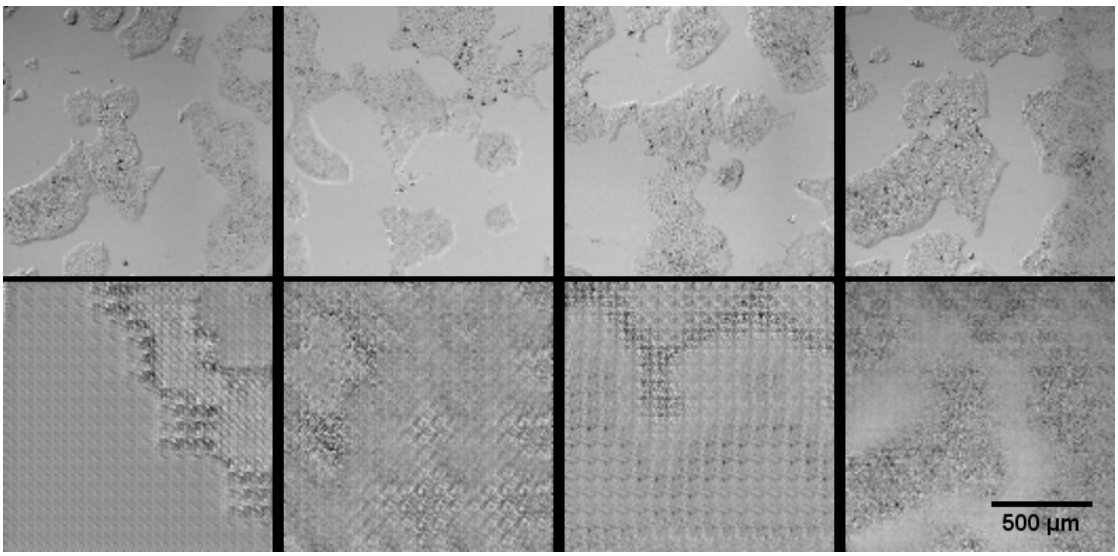

Figure 4.5: Models trained and tested with the original images predicted the cell growth very poorly. The lower row contains the first predicted frames and the upper row the corresponding real frames. The columns differ by the time interval between the frames of the videos in training set and test set. From left to right: one hour, two hours, three hours and four hours. Since there cannot be a video that is included in all datasets, the frames of the columns belong to different videos.

Segmented training set and test set

On the right side in figure 4.6 a predicted frame and in the middle right the corresponding GT frame are depicted. The prediction is much paler than the GT, which is the case for

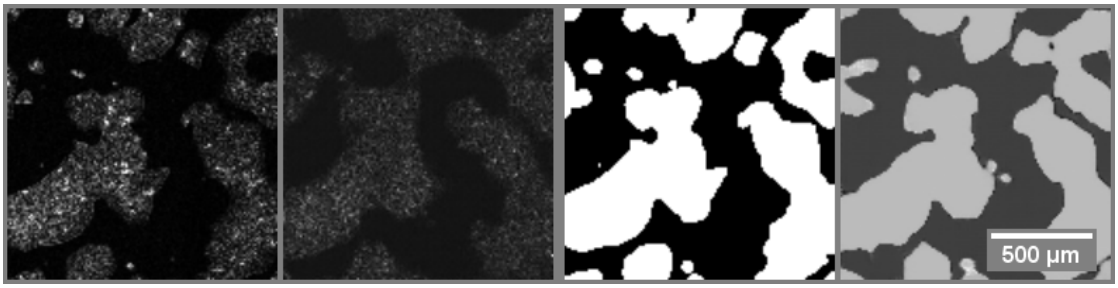

Figure 4.6: Predicted frames are paler than the corresponding real frames. On the left is depicted a real frame belonging to the first predicted frame in the middle left. The model was trained and tested with a dataset whose frames were preprocessed with edge detection. On the right a first predicted frame from a model that worked with segmented data is depicted. In the middle right the corresponding real frame is shown.

all predictions for segmented datasets. However, the boundaries between the cells and the background are very clear, so segmentation for evaluation was simple.
In figure 4.7 the results for the IoU and the cell area normalized to the GT cell area are shown. The precision and recall of the prediction are depicted in figure 4.8. Metric values for the individual predicted frames are plotted in each case. The average standard deviations within the test sets are 0.0357 for the IoU, 0.0186 for the cell area, 0.0231 for the precision and 0.0175 for the recall. Individual standard deviations are listed

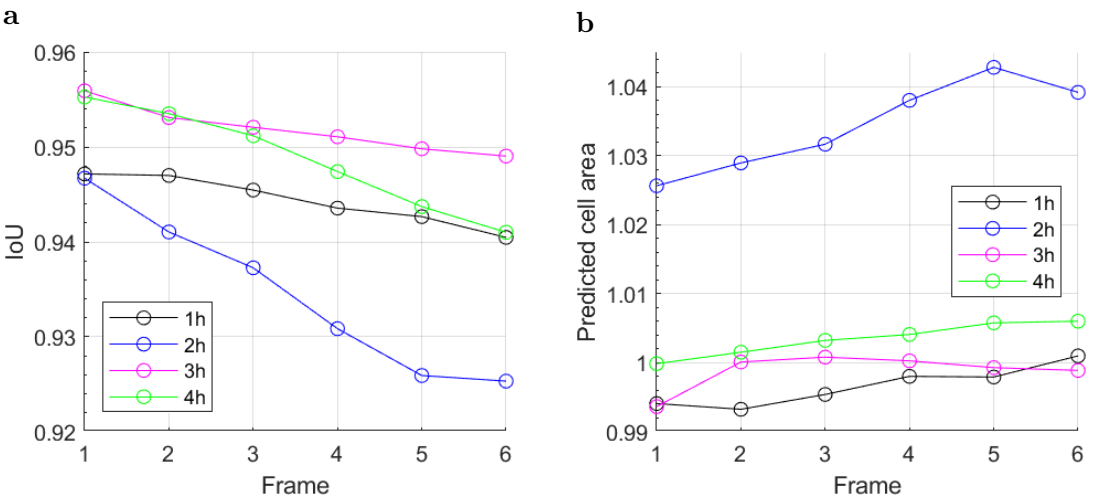

Figure 4.7: The metric average values for the prediction accuracy change from the first to the last predicted frame. For each of the datasets with different time intervals between the frames a graph is depicted. The training set and test set were segmented during preprocessing. Tables 7.5 and 7.6 list the plotted values and associated standard deviations. **a** Intersection over Union (IoU). **b** Predicted cell area normalized to the ground truth cell area.

along with their corresponding metric values in tables 7.5, 7.6, 7.7 and 7.8. Except for considering the predicted cell area for time intervals of one and three hours, each prediction has the highest accuracy for the first predicted frame and gets worse with each subsequent frame. This is reasonable because the errors in the prediction add up with each frame. The deterioration is greatest for the IoU, but mostly not really significant for the other metrics. The prediction accuracy for the two hour time intervals is clearly different from the other three. Except for recall, it is always worse than the others. Since more cell area than was present in the GT was predicted for two hours, a better recall makes sense, because with a larger overall cell area it is likely that a larger proportion of the GT cells was correctly predicted. Why this dataset deviates so clearly from the others is not obvious and could have happened by chance. To find the reasons for this, further experiments would have to be conducted. All results of the other datasets are close to each other and, like those of the two hour interval, better than those from the training set size and frame resolution variation. Thus, the trend of increased prediction accuracy with larger training sets is continued because all models were trained with 644 videos, which is 78.9% more than were included in the largest training set of the training set size and frame resolution variation. There is no clear trend for the influence of the length of time intervals between frames on the growth prediction accuracy, especially when the standard deviations are taken into account. If the two hour time interval is excluded, the results of the different datasets are all quite equal and there is no consistent order of time intervals across all metrics in terms of prediction accuracy.

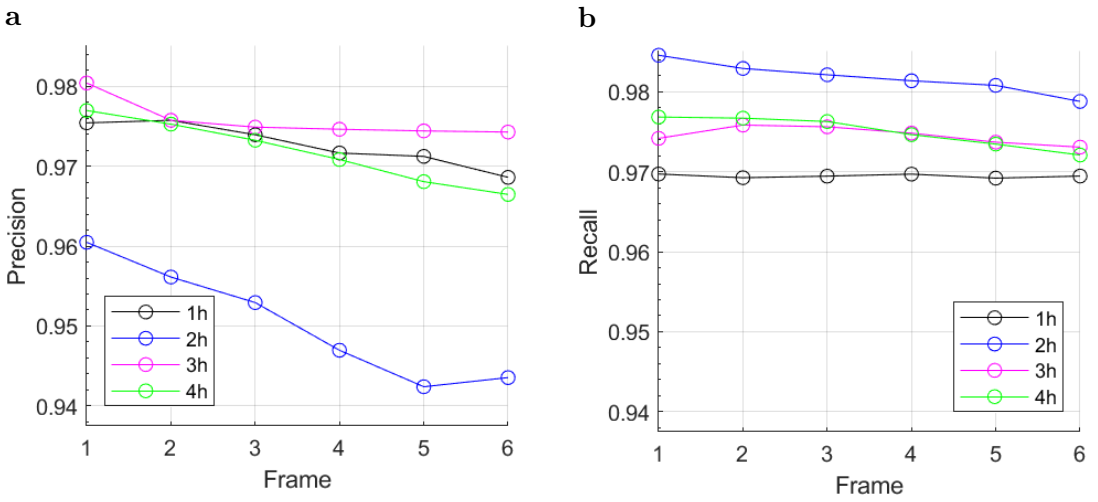

Figure 4.8: The metric average values for the prediction accuracy change from the first to the last predicted frame. For each of the datasets with different time intervals between the frames a graph is depicted. The training set and test set were segmented during preprocessing. Tables 7.7 and 7.8 list the plotted values and associated standard deviations. **a** Precision. **b** Recall.

Training set and test set preprocessed with edge detection

On the left side in figure 4.6 a GT frame and in the middle left the corresponding predicted frame for the dataset preprocessed with edge detection are depicted. The predicted frame is again much paler, but there are still clear boundaries between the cells and the background visible. This made segmentation for evaluation challenging, but doable. It should be noted that because the segmentation was difficult, it was likely not done correctly for a significant portion of the predicted and GT frames. Therefore, the following figures probably do not reflect the raw prediction accuracy, but rather both prediction and segmentation accuracy.

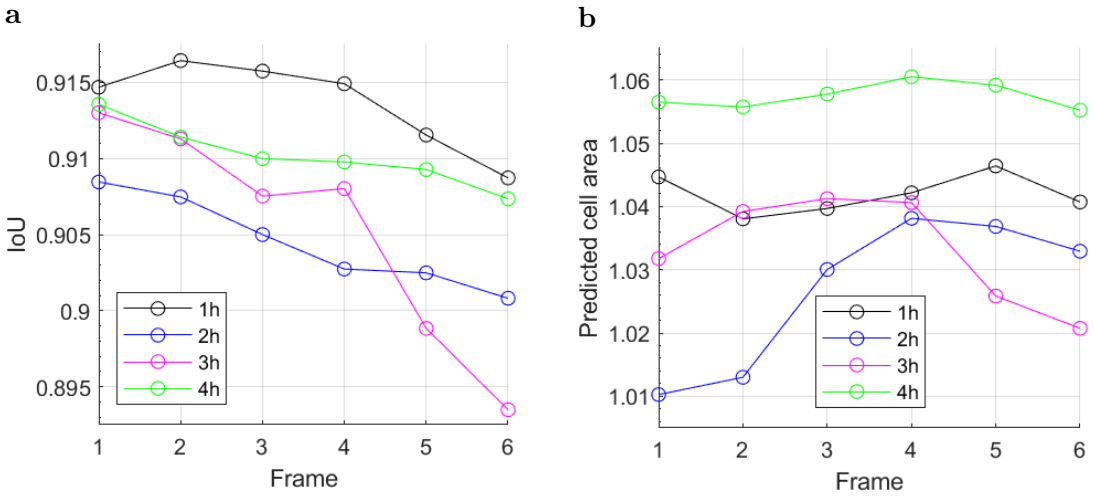

Figure 4.9: The metric average values for the prediction accuracy change from the first to the last predicted frame. For each of the datasets with different time intervals between the frames a graph is depicted. The training set and test set were preprocessed with edge detection. Tables 7.9 and 7.10 list the plotted values and associated standard deviations. **a** Intersection over Union (IoU). **b** Predicted cell area normalized to the ground truth cell area.

IoU and the predicted cell area normalized to the GT cell area are depicted in figure 4.9. Figure 4.10 shows the precision and recall. Metric values for the individual predicted frames are plotted again. The average standard deviations within the test set are 0.0522 for the IoU, 0.0531 for the cell area, 0.0461 for the precision and 0.0215 for the recall. Individual standard deviations and the respective metric values are listed in tables 7.9. 7.10, 7.11 and 7.12. In contrast to the results of the segmented dataset, there is no clear deterioration of the prediction from the first to the last frame across all metrics. While IoU and precision worsen slightly, predicted cell area and recall sometimes improve and other times worsen with the next frame. In general, the predictions for the dataset preprocessed with edge detection are slightly worse than those for the segmented data. All metric values differ by about 0.01 - 0.07 when comparing the results for both datasets. The influence of cell area on recall is visible, as the recall graphs loosely recompose the

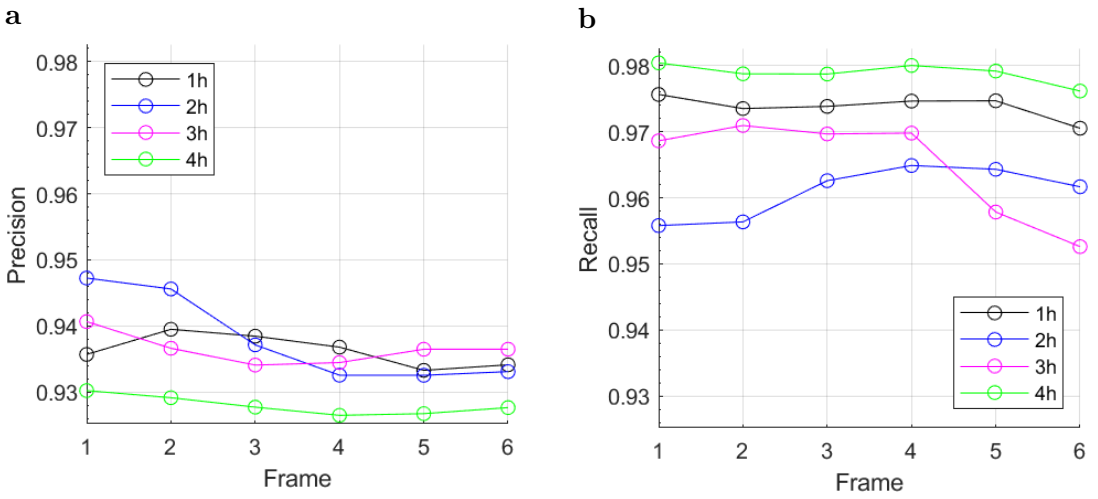

Figure 4.10: The metric average values for the prediction accuracy change from the first to the last predicted frame. For each of the datasets with different time intervals between the frames a graph is depicted. The training set and test set were preprocessed with edge detection. Tables 7.11 and 7.12 list the plotted values and associated standard deviations. **a** Precision. **b** Recall.

structure of the cell area graphs, only at different values. This is again because it becomes more likely that more of the GT cell area is predicted when more area is predicted overall. On the other hand, more predicted cell area negatively affects precision because a greater proportion of that area tends to be predicted incorrectly. This can be seen in the precision graphs, as they have approximately an inverse structure to that of the cell area graphs. However, there is again no consistent order across all metrics for the different time intervals in terms of prediction accuracy.

4.2.3 Data Augmentation

Lastly, the impact of using two data augmentation methods on prediction accuracy is discussed. Results for the ”Mirrored” and ”Rotated” training sets are compared with those of the two training sets without data augmentation. Each model was evaluated with the test set of the ”Control” dataset, as this sets the benchmark for the desired performance. It contains 84 videos with 12 frames each. Due to a bug, predictions were again only made for 83 sequences. All frames of the training sets and the test set are segmented, have a time interval of three hours between each two frames and a resolution of 512×512. During testing, the FutureGAN automatically downscaled the GT frames to 128×128 so that they have the same size as the predicted ones. Figure 4.11 depicts the results for all metrics and training sets. They are grouped by the different training sets and color-coded by the metrics. The average standard deviations within the test sets are 0.0436 for the IoU, 0.0284 for the cell area, 0.0274 for the precision and 0.0261 for the recall. Individual standard deviations and the associated metric values are listed

in table 7.13. All metric values for the ”No augmentation” training set are very slightly worse than those for the ”Control” training set. With about 0.01, the biggest difference is in the IoU. This is significantly lower than the corresponding standard deviations and thus the differences are not that substantial. A deviation that small is somewhat surprising, since the ”No augmentation” training set contains only half the videos of the ”Control” training set.

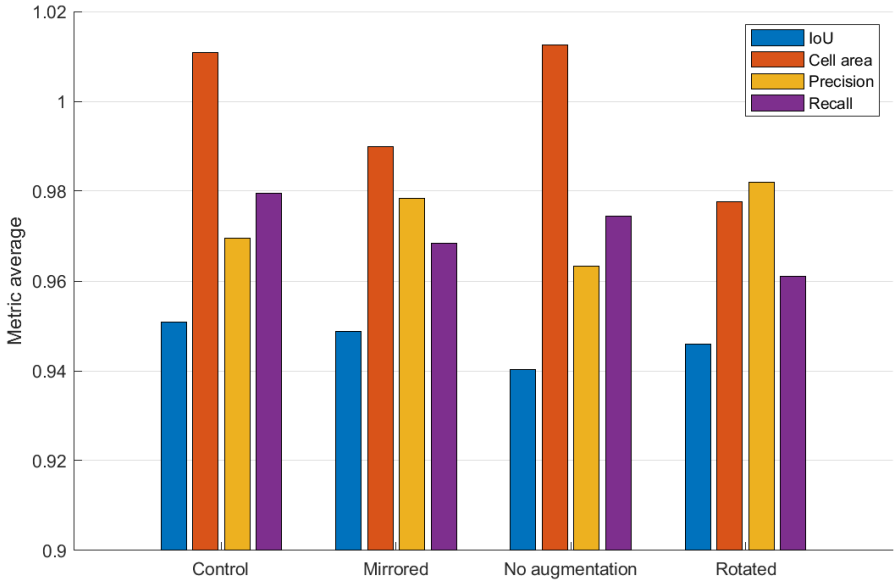

Figure 4.11: The average values have changed for the different training sets of the data augmentation variation for most metrics. Intersection over Union (IoU), normalized cell area, precision and recall averages of prediction for models trained on the ”Control” (whole dataset), ”Mirrored” (half the dataset; original and mirrored along vertical axis), ”No augmentation” (half the dataset) and ”Rotated” (half the dataset; original and rotated 90 degrees counterclockwise) training set are depicted. All models were tested on the same test set. Table 7.13 lists the plotted values and associated standard deviations.

While the models trained without data augmentation predicted too much cell area, the other two predicted too little. This is most likely the reason why the precision increased with data augmentation and even exceeded that for the ”Control” training set. On the other hand, recall dropped with data augmentation, likely due to the lower cell area. These trends for cell area, precision, and recall were the same for both augmentation methods, but more pronounced for the rotated training set. The data augmentation also slightly increased the IoU to nearly match the one for the ”Control” training set.

5 Discussion

Limitations of the used approach

Method used for growth prediction: Cell growth was optimized for one method of future frame prediction in this thesis. The FutureGAN only read in six frames at a time and predicted another six. Other lengths for the input and output sequence would be conceivable and could possibly provide better results. Furthermore only frames with a resolution of 128 $\times$ 128 were predicted, because with the used hardware better predictions would have taken much longer. Thus, less variations of the dataset could have been trained in the same time.

Dataset for training and testing: Compared to typical sizes of datasets for ML, a small dataset was used. Dataset for future frame predictions often contain more than hundreds of thousands of frames [9]. The quality of the dataset was also not optimal. There was a high cell density, which is why some cutouts contained almost exclusively cells. Thus, the variation of the dataset was increased. The higher the variation of a dataset, the larger it should be to lead to good results. In addition, there were generally only minor changes between two images, which raises the question of whether ML was even necessary for the used dataset. It could be possible that extrapolation would have been sufficient. However, the use of ML was still justified because on the basis of the results obtained, the method can be extended and further improved. There are many ways to increase the change between two frames. In addition, the quality of the data was suboptimal because unwanted shadows, brightness variations and foreign particles were present in the microscope images. The segmentation accuracy also influenced the quality of the datasets that were finally used for training and testing of the ML models.

Conclusion

For the examined parameters of the dataset with values in the selected ranges, it can be said that the size of the training set has by far the greatest influence on the accuracy of the growth prediction. In contrast, the resolution of the frames has no significant influence on the quality of the prediction if a sufficiently large training set is used. It therefore makes sense to create many videos with low resolution rather than a few with high resolution during preprocessing. Furthermore, if the FutureGAN was trained and tested with original data, it was mostly unable to generate any meaningful frames. However, the quality of the prediction improved with a larger time interval between the frames. As a consequence, it is not unreasonable to think that good results could also be achieved

with the original data with a larger training set and even larger time intervals between the frames.
For this thesis, the best results were obtained with segmented data. The degree of segmentation was the second largest influencing factor on the prediction accuracy after the training set size. However, it cannot be ruled out that the results for data preprocessed with edge detection could not be as good or even better than those of the segmented data, if larger training sets and a better segmentation technique for postprocessing are used. For this reason, the assumption that individual cells in cell clusters can provide information about how the boundaries of these clusters will grow should not be rejected. It could neither be confirmed nor dismissed with the results of the experiments of this thesis, since the segmentation was difficult in postprocessing and the resulting inaccuracies affected the metric values of the results. Varying the time interval between frames did not cause a clear improvement or worsening of the results for the segmented or with edge detection preprocessed data. Whereas the prediction tends to become slightly less accurate with each additional predicted frame. Furthermore, apart from the network's recall, a statistically significant improvement could be achieved through the use of data augmentation. Thus, the use of it to the extend tested is legitimate if recall is not critical.

6 Outlook

Improvement of the approach More data could be acquired from the beginning by growing cells in more than two wells. This would result in a larger training set and presumably better results. A further extended use of data augmentation methods would increase the training set size as well. However, these should be used with caution, because if the training set deviates too far from real data, undesirable and unrealistic results could be generated. If a lower cell density is used, it would probably also lead to an improvement of the prediction, as it is less likely that almost exclusively cells are in the frames of a train sequence. Thus, the variation of the training set is reduced. To enhance the segmentation quality in pre- and postprocessing methods like contouring could be applied. This would avoid the creation of false background patches in cell clusters during segmentation.
In the case that the time intervals between the frames of the training videos are higher than the ones of the original microscopy data, instead of only every third frame, for example, the two sequences shifted to these frames could also be added to the training data. Since this way no frames are used twice, but they all come from the same video, the improvement could be quite small. Another way to increase the amount of training data is to add the previously predicted frames to the training set. This should as well be done with caution and not too extensively, as the predictions already contain small inaccuracies.

Extension of the approach First, the entire process of preprocessing, training, prediction and postprocessing could be automated. Next, it could be ensured that there are more differences between the frames by using a higher magnification for microscopy or by splitting the images into smaller sections. A way to achieve at least more absolute pixel difference is to predict frames with higher resolution of for example 512×512 or even of 1024×1024. Instead of brightfield microscopy, fluorescence microscopy could also be used. In this technique, the sample to which fluorophores have been added is illuminated with light of a specific wavelength. It is absorbed by the fluorophores and they emit light with a longer wavelength. Consequently, the fluorescent light can be separated from the illumination light by spectral emission filters. As a result, there is a much greater contrast between the cells and the background than in brightfield microscopy, which makes segmentation easier. However, as a disadvantage this type of microscopy can cause cell death due to phototoxicity. For smoother videos motion interpolation, a method where one or more additional frames are inserted between each two frames, could be applied to the predicted frame sequences.
In addition to growth prediction for 2D cell cultures, predictions could also be made for organoids or other 3D biological structures. This is conceivable either by projecting

them onto a 2D plane or using a 3D representation. For the latter case, perhaps spatio-temporal 4D convolutions could be used. In addition to the FutureGAN, other GAN models or other types of networks should be tested for growth prediction for cell culture experiments. For example, RNNs or AE that are not trained in an adversarial manner could be used for the future frame prediction [9].

Since some of the mentioned methods for improvement and extension of the approach involve an increase in computational effort, attempts should be made to speed up training for a faster iteration process. Possibilities for this are the use of even more GPUs, tuning of suitable hyperparameters of the network, like the learning rate, or using a different method for the network parameter updates.

Applications Growth prediction can assist in the investigation of organoid structure formation. As a consequence this technique can help to improve the understanding of tumor morphogenesis. In addition, machine learning prediction could be used as a digital control group for treatment testing on organoids in the future. Furthermore, the development of other biological organisms, such as bacteria could be investigated with this method. Improved methods for growth prediction in cell culture experiments also have the potential to be transferred to other applications, such as growth modeling of tumors *in vivo*. This modeling as a part of predictive medicine could potentially lead to better treatment management and surgical planning [53]. Depending on the application and interest, the cutout size of each video should be adjusted. It is conceivable that in one case the development of a certain region and in another the progression of a sample as a whole is crucial.
In conclusion, machine learning methods and especially various types of artificial neural networks have the potential to provide accurate growth predictions for many types of biological cell culture experiments and applications beyond.

Bibliography

[1] A. M. TURING. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX(236):433–460, 10 1950. ISSN 0026-4423. doi: 10.1093/mind/LIX.236.433. URL https://doi.org/10.1093/mind/LIX.236.433.

[2] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[3] Artificial intelligence & autopilot. https://www.tesla.com/AI, 2023.

[4] Abstracts written by ChatGPT fool scientists. https://www.nature.com/articles/d41586-023-00056-7, 2023.

[5] Filippo Amato, Alberto López-Rodríguez, Eladia Peña-Méndez, Petr Vaňhara, Aleš Hampl, and Josef Havel. Artificial neural networks in medical diagnosis. *J Appl Biomed*, 11:47–58, 12 2013. doi: 10.2478/v10136-012-0031-x.

[6] World health organization - cancer fact sheets. https://www.who.int/news-room/fact-sheets/detail/cancer, 2023.

[7] International agency for research on cancer - cancer fact sheets. https://gco.iarc.fr/today/fact-sheets-cancers, 2023.

[8] Peter Nagle, John Plukker, Christina Muijs, Peter van Luijk, and Robert Coppes. Patient-derived tumor organoids for prediction of cancer treatment response. *Seminars in Cancer Biology*, 53, 06 2018. doi: 10.1016/j.semcancer.2018.06.005.

[9] Sergiu Oprea, Pablo Martinez-Gonzalez, Alberto Garcia-Garcia, John Alejandro Castro-Vargas, Sergio Orts-Escolano, Jose Garcia-Rodriguez, and Antonis Argyros. A review on deep learning techniques for video prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(6):2806–2826, jun 2022. doi: 10.1109/tpami.2020.3045007. URL https://doi.org/10.1109%2Ftpami.2020.3045007.

[10] Kenneth P. Olive, Michael A. Jacobetz, Christian J. Davidson, Aarthi Gopinathan, Dominick McIntyre, Davina Honess, Basetti Madhu, Mae A. Goldgraben, Meredith E. Caldwell, David Allard, Kristopher K. Frese, Gina DeNicola, Christine Feig, Chelsea Combs, Stephen P. Winter, Heather Ireland-Zecchini, Stefanie Reichelt, William J. Howat, Alex Chang, Mousumi Dhara, Lifu Wang, Felix Rückert, Robert Grützmann, Christian Pilarsky, Kamel Izeradjene, Sunil R. Hingorani, Pearl Huang, Susan E. Davies, William Plunkett, Merrill Egorin, Ralph H. Hruban, Nigel Whitebread, Karen McGovern, Julian Adams, Christine Iacobuzio-Donahue, John Griffiths, and David A. Tuveson. Inhibition of hedgehog signaling enhances delivery

of chemotherapy in a mouse model of pancreatic cancer. *Science*, 324(5933):1457–1461, 2009. doi: 10.1126/science.1171362. URL `https://www.science.org/doi/abs/10.1126/science.1171362`.

[11] Jörg Kleeff, Murray Korc, Minoti Apte, Carlo Vecchia, Colin Johnson, Andrew Biankin, Rachel Neale, Margaret Tempero, David Tuveson, Ralph Hruban, and John Neoptolemos. Pancreatic cancer. *Nature Reviews Disease Primers*, 2:16022, 04 2016. doi: 10.1038/nrdp.2016.22.

[12] Fuuka Minami, Norihiko Sasaki, Yuuki Shichi, Fujiya Gomi, Masaki Michishita, Kozo Ohkusu-Tsukada, Masashi Toyoda, Kimimasa Takahashi, and Toshiyuki Ishiwata. Morphofunctional analysis of human pancreatic cancer cell lines in 2- and 3-dimensional cultures. *Scientific Reports*, 11:6775, 03 2021. doi: 10.1038/s41598-021-86028-1.

[13] Linda C. Chu, Zhen J. Wang, Avinash Kambadakone, Elizabeth M. Hecht, Jin He, Amol K. Narang, Daniel A. Laheru, Hina Arif-Tiwari, Priya Bhosale, Candice W. Bolan, Olga R. Brook, Abraham F. Bezuidenhout, Richard K.G. Do, Samuel J. Galgano, Ajit H. Goenka, Alexander R. Guimaraes, David M. Hough, Naveen Kulkarni, Ott Le, Lyndon Luk, Lorenzo Mannelli, Michael Rosenthal, Guillermo Sangster, Zarine K. Shah, Erik V. Soloff, Parag P. Tolat, Marc Zins, Elliot K. Fishman, Eric P. Tamm, and Atif Zaheer. Postoperative surveillance of pancreatic ductal adenocarcinoma (pdac) recurrence: practice pattern on standardized imaging and reporting from the society of abdominal radiology disease focus panel on pdac. *Abdominal Radiology*, 2022. ISSN 2366-004X. doi: 10.1007/s00261-022-03693-0. Publisher Copyright: © 2022, The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature.

[14] Samuel Randriamanantsoa, A. Papargyriou, H. Maurer, K. Peschke, M. Schuster, G. Zecchin, Katja Steiger, R. Öllinger, D. Saur, C. Scheel, R. Rad, E. Hannezo, M. Reichert, and A. Bausch. Spatiotemporal dynamics of self-organized branching in pancreas-derived organoids. *Nature Communications*, 13, 09 2022. doi: 10.1038/s41467-022-32806-y.

[15] Serafeim Loukas. What is machine learning: Supervised, unsupervised, semi-supervised and reinforcement learning methods. `https://towardsdatascience.com/what-is-machine-learning-a-short-note-on-supervised-unsupervised-semi-supervised-and-aed1573ae9bb`, 2020.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[17] Classic pong - android app. https://www.chip.de/downloads/Classic-Pong-Android-App_57466163.html, 2022.

[18] Brij Rokad. Machine learning approaches and its applications. https://medium.datadriveninvestor.com/machine-learning-approaches-and-its-applications-7bfbe782f4a8, 2019.

[19] Zied HY. Deep learning from scratch. https://ziedhy.github.io/Introduction_Deep_Learning.html, 2018.

[20] Joe Greener, Shaun Kandathil, Lewis Moffat, and David Jones. A guide to machine learning for biologists. *Nature Reviews Molecular Cell Biology*, 23, 09 2021. doi: 10.1038/s41580-021-00407-0.

[21] Siddharth Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *International Journal of Engineering Applied Sciences and Technology*, 04:310–316, 05 2020. doi: 10.33564/IJEAST.2020.v04i12.054.

[22] Stanford cs231n module 1: Neural networks - neural networks part 1: Setting up the architecture. https://cs231n.github.io/neural-networks-1/, 2022.

[23] Cecbur. File:convolutional neural network neuralnetworkfeaturelayers.gif. https://commons.wikimedia.org/wiki/File:Convolutional_Neural_Network_NeuralNetworkFeatureLayers.gif, 2019.

[24] Stanford cs231n module 2: Convolutional neural networks - convolutional neural networks: Architectures, convolution / pooling layers. https://cs231n.github.io/convolutional-networks/, 2022.

[25] Saad Albawi, Tareq Abed Mohammed, and Saad ALZAWI. Understanding of a convolutional neural network. 2017 International Conference on Engineering and Technology (ICET), 2017.

[26] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[27] Katarzyna Janocha and Wojciech Czarnecki. On loss functions for deep neural networks in classification. *Schedae Informaticae*, 25, 02 2017. doi: 10.4467/20838476SI.16.004.6185.

[28] Stanford cs231n module 1: Neural networks - neural networks part 3: Learning and evaluation. https://cs231n.github.io/neural-networks-3/, 2022.

[29] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015. doi: 10.1038/nature14539.

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[31] Apratim Bhattacharyya, Mario Fritz, and Bernt Schiele. Long-term on-board prediction of people in traffic scenes under uncertainty, 2017. URL https://arxiv.org/abs/1711.09026.

[32] Anthony Hu, Fergal Cotter, Nikhil Mohan, Corina Gurau, and Alex Kendall. Probabilistic future prediction for video scene understanding, 2020. URL https://arxiv.org/abs/2003.06409.

[33] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37 (3):362–386, apr 2020. doi: 10.1002/rob.21918. URL https://doi.org/10.1002%2Frob.21918.

[34] Wei Yu, Yichao Lu, Steve Easterbrook, and Sanja Fidler. Efficient and information-preserving future frame prediction and beyond. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1eY_pVYvB.

[35] Stanford cs231n student-contributed posts: Recurrent neural networks. https://cs231n.github.io/rnn/, 2022.

[36] MarcAurelio Ranzato, Arthur Szlam, Joan Bruna, Michael Mathieu, Ronan Collobert, and Sumit Chopra. Video (language) modeling: a baseline for generative models of natural videos, 2014. URL https://arxiv.org/abs/1412.6604.

[37] Jianjin Zhang, Yunbo Wang, Mingsheng Long, Wang Jianmin, and Philip S Yu. Z-order recurrent neural networks for video prediction. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 230–235, 2019. doi: 10.1109/ICME.2019.00048.

[38] Marc Oliu, Javier Selva, and Sergio Escalera. Folded recurrent neural networks for future video prediction, 2017. URL https://arxiv.org/abs/1712.00311.

[39] Gan graphic. https://inseclab.uit.edu.vn/idsgan-tiem-nang-cua-he-thong-phat-hien-xam-nhap-ids-dua-tren-mang-sinh-doi-khang-gan/, 2020.

[40] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. URL https://arxiv.org/abs/1406.2661.

[41] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Commun. ACM*, 63(11):139–144, oct 2020. ISSN 0001-0782. doi: 10.1145/3422622. URL https://doi.org/10.1145/3422622.

[42] Osamu Shouno. Photo-realistic video prediction on natural videos of largely changing frames, 2020. URL https://arxiv.org/abs/2003.08635.

[43] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2017. URL https://arxiv.org/abs/1710.10196.

[44] Sandra Aigner and Marco Körner. Futuregan: Anticipating the future frames of video sequences using spatio-temporal 3d convolutions in progressively growing gans. https://arxiv.org/abs/1810.01325, 2018.

[45] K. Aditya Shastry and H. A. Sanjay. *Machine Learning for Bioinformatics*, pages 25–39. Springer Singapore, Singapore, 2020. ISBN 978-981-15-2445-5. doi: 10.1007/978-981-15-2445-5_3. URL https://doi.org/10.1007/978-981-15-2445-5_3.

[46] Angela Serra, Paola Galdi, and Roberto Tagliaferri. Machine learning for bioinformatics and neuroimaging. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8, 01 2018. doi: 10.1002/widm.1248.

[47] Zhiqiang Zeng, Hua Shi, Yun Wu, and Zhiling Hong. Survey of natural language processing techniques in bioinformatics. *Computational and Mathematical Methods in Medicine*, 2015:1–10, 10 2015. doi: 10.1155/2015/674296.

[48] Andrew Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander Nelson, Alex Bridgland, Hugo Penedones, Stig Petersen, Karen Simonyan, Steve Crossan, Pushmeet Kohli, David Jones, David Silver, Koray Kavukcuoglu, and Demis Hassabis. Improved protein structure prediction using potentials from deep learning. *Nature*, 577:1–5, 01 2020. doi: 10.1038/s41586-019-1923-7.

[49] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon Kohl, Andrew Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596:1–11, 08 2021. doi: 10.1038/s41586-021-03819-2.

[50] Kurt Wüthrich. The way to nmr structures of proteins. *Nature structural biology*, 8:923–5, 12 2001. doi: 10.1038/nsb1101-923.

[51] Rcsb protein data bank. https://www.rcsb.org/, 2023.

[52] Robail Yasrab, Jincheng Zhang, Polina Smyth, and Michael Pound. Predicting plant growth from time-series data using deep learning. *Remote Sensing*, 13:331, 01 2021. doi: 10.3390/rs13030331.

[53] Ling Zhang, Le Lu, Xiaosong Wang, Ronald Summers, Jianhua Yao, and Mohammadhadi Bagheri. Spatio-temporal convolutional lstms for tumor growth prediction by learning 4d longitudinal patient data. *IEEE transactions on medical imaging*, 2019:1–13, 09 2019. doi: 10.1109/TMI.2019.2943841.

[54] Bauschlab chair of Cellular Biophysics. `https://www.bauschlab.org/`, 2023.

[55] Bauschlab - Personal site of PhD student Fabian Englbrecht. `https://www.bauschlab.org/fabian`, 2023.

[56] TUM - Personal site of master student Sandra-Hannelore Andrusca. `https://www.ph.tum.de/about/people/vcard/30E64D3495C7EC16/`, 2023.

[57] TUM-LMF/FutureGAN GitHub repository. `https://github.com/TUM-LMF/FutureGAN`, 2018.

Appendix

Dataset Size and Frame Resolution variation

The following tables show the averages for the following metrics: IoU, Cell area (normalized to GT cell area), precision and recall. Training was performed on 18/45/90/180 videos and testing on 39 videos. Each training video counted as two because they each contained 25 frames and the FutureGAN was used with 12 frames per video. The different resolutions were 128×128/256×256/512×512/660×660.

Resolution		**128**	**256**	**512**	**660**
18 Videos	**IoU Avg.**	0,5820	0,6400	0,5861	0,7105
	St. Dev.	0,2080	0,0962	0,2844	0,1441
45 Videos	**IoU Avg.**	0,8356	0,8572	0,8597	0,8395
	St. Dev.	0,0728	0,0639	0,0667	0,0733
90 Videos	**IoU Avg.**	0,8931	0,8850	0,8924	0,8862
	St. Dev.	0,0510	0,0555	0,0531	0,0549
180 Videos	**IoU Avg.**	0,9173	0,9101	0,9203	0,9093
	St. Dev.	0,0447	0,0476	0,0410	0,0482

Table 7.1: IoU average and standard deviation by resolution for different dataset sizes.

Resolution		**128**	**256**	**512**	**660**
18 Videos	**Cell Area Avg.**	0,7394	1,1002	0,8242	0,8847
	St. Dev.	0,2477	0,1663	0,4206	0,1591
45 Videos	**Cell Area Avg.**	0,9971	1,0757	1,0245	1,0235
	St. Dev.	0,0622	0,0646	0,0592	0,0619
90 Videos	**Cell Area Avg.**	1,0007	1,0254	1,0037	1,0403
	St. Dev.	0,0435	0,0519	0,0429	0,0499
180 Videos	**Cell Area Avg.**	1,0121	1,0066	1,0170	0,9999
	St. Dev.	0,0247	0,0315	0,0259	0,0300

Table 7.2: Predicted cell area (normalized to the GT cell area) average and standard deviation by resolution for different dataset sizes.

Resolution		**128**	**256**	**512**	**660**
18 Videos	**Precision Avg.**	0,8550	0,7524	0,8160	0,8825
	St. Dev.	0,1153	0,1160	0,1137	0,0572
45 Videos	**Precision Avg.**	0,9118	0,8915	0,9136	0,9023
	St. Dev.	0,0519	0,0544	0,0473	0,0532
90 Videos	**Precision Avg.**	0,9433	0,9278	0,9413	0,9219
	St. Dev.	0,0352	0,0440	0,0324	0,0439
180 Videos	**Precision Avg.**	0,9510	0,9496	0,9504	0,9523
	St. Dev.	0,0303	0,0308	0,0282	0,0303

Table 7.3: Prediction precision average and standard deviation by resolution for different dataset sizes.

Resolution		**128**	**256**	**512**	**660**
18 Videos	**Recall Avg.**	0,6391	0,8091	0,6838	0,7815
	St. Dev.	0,2295	0,0277	0,3404	0,1505
45 Videos	**Recall Avg.**	0,9073	0,9559	0,9343	0,9214
	St. Dev.	0,0506	0,0295	0,0453	0,0479
90 Videos	**Recall Avg.**	0,9430	0,9496	0,9441	0,9573
	St. Dev.	0,0340	0,0310	0,0385	0,0281
180 Videos	**Recall Avg.**	0,9619	0,9553	0,9661	0,9517
	St. Dev.	0,0233	0,0293	0,0219	0,0301

Table 7.4: Prediction recall average and standard deviation by resolution for different dataset sizes.

Time intervals and Segmentation variation

The following tables show the averages for the following metrics: IoU, Cell area (normalized to GT cell area), precision and recall. Training was performed on 644 videos and testing on 83 videos. The resolution was 512×512 in each case.

Segmented training set and test set

Frame		**1**	**2**	**3**	**4**	**5**	**6**	**Avg.**
1h	**IoU Avg.**	0,9472	0,9470	0,9455	0,9436	0,9427	0,9405	0,9444
	St. Dev.	0,0348	0,0347	0,0345	0,0357	0,0360	0,0374	0,0356
2h	**IoU Avg.**	0,9467	0,9410	0,9373	0,9309	0,9259	0,9253	0,9345
	St. Dev.	0,0363	0,0413	0,0430	0,0485	0,0511	0,0515	0,0453
3h	**IoU Avg.**	0,9559	0,9531	0,9521	0,9511	0,9498	0,9490	0,9518
	St. Dev.	0,0274	0,0287	0,0293	0,0297	0,0301	0,0308	0,0293
4h	**IoU Avg.**	0,9552	0,9535	0,9512	0,9474	0,9437	0,9410	0,9487
	St. Dev.	0,0289	0,0298	0,0304	0,0334	0,0353	0,0379	0,0326

Table 7.5: IoU average and standard deviation by predicted frame for different time intervals between frames. The training set and test set were segmented during preprocessing.

Frame		**1**	**2**	**3**	**4**	**5**	**6**	**Avg.**
1h	**Cell Area Avg.**	0,9941	0,9933	0,9954	0,9980	0,9979	1,0010	0,9966
	St. Dev.	0,0112	0,0125	0,0139	0,0163	0,0187	0,0199	0,0154
2h	**Cell Area Avg.**	1,0257	1,0290	1,0317	1,0380	1,0428	1,0392	1,0344
	St. Dev.	0,0216	0,0289	0,0306	0,0403	0,0453	0,0426	0,0349
3h	**Cell Area Avg.**	0,9937	1,0001	1,0008	1,0003	0,9993	0,9989	0,9989
	St. Dev.	0,0094	0,0102	0,0123	0,0126	0,0133	0,0179	0,0126
4h	**Cell Area Avg.**	0,9999	1,0015	1,0033	1,0041	1,0058	1,0060	1,0034
	St. Dev.	0,0071	0,0092	0,0109	0,0133	0,0144	0,0149	0,0116

Table 7.6: Predicted cell area (normalized to the GT cell area) average and standard deviation by predicted frame for different time intervals between frames. The training set and test set were segmented during preprocessing.

Frame		1	2	3	4	5	6	Avg.
1h	**Precision Avg.**	0,9754	0,9757	0,9739	0,9716	0,9712	0,9686	0,9728
	St. Dev.	0,0167	0,0156	0,0180	0,0188	0,0183	0,0209	0,0180
2h	**Precision Avg.**	0,9605	0,9561	0,9529	0,9470	0,9424	0,9435	0,9504
	St. Dev.	0,0279	0,0335	0,0350	0,0417	0,0450	0,0437	0,0378
3h	**Precision Avg.**	0,9804	0,9757	0,9749	0,9746	0,9744	0,9743	0,9757
	St. Dev.	0,0147	0,0157	0,0169	0,0173	0,0162	0,0180	0,0165
4h	**Precision Avg.**	0,9770	0,9753	0,9732	0,9708	0,9681	0,9665	0,9718
	St. Dev.	0,0157	0,0173	0,0191	0,0214	0,0227	0,0238	0,0200

Table 7.7: Prediction precision average and standard deviation by predicted frame for different time intervals between frames. The training set and test set were segmented during preprocessing.

Frame		1	2	3	4	5	6	Avg.
1h	**Recall Avg.**	0,9697	0,9692	0,9694	0,9697	0,9692	0,9695	0,9695
	St. Dev.	0,0219	0,0229	0,0214	0,0226	0,0241	0,0238	0,0228
2h	**Recall Avg.**	0,9845	0,9829	0,9821	0,9814	0,9808	0,9788	0,9817
	St. Dev.	0,0121	0,0127	0,0136	0,0134	0,0140	0,0170	0,0138
3h	**Recall Avg.**	0,9742	0,9758	0,9756	0,9748	0,9737	0,9731	0,9745
	St. Dev.	0,0157	0,0163	0,0165	0,0166	0,0183	0,0190	0,0171
4h	**Recall Avg.**	0,9768	0,9767	0,9763	0,9746	0,9734	0,9721	0,9750
	St. Dev.	0,0157	0,0157	0,0145	0,0159	0,0170	0,0190	0,0163

Table 7.8: Prediction recall average and standard deviation by predicted frame for different time intervals between frames. The training set and test set were segmented during preprocessing.

Training set and test set preprocessed with edge detection

Frame		1	2	3	4	5	6	Avg.
1h	**IoU Avg.**	0,9147	0,9164	0,9157	0,9149	0,9115	0,9087	0,9137
	St. Dev.	0,0460	0,0452	0,0462	0,0453	0,0468	0,0461	0,0459
2h	**IoU Avg.**	0,9085	0,9075	0,9050	0,9027	0,9025	0,9008	0,9045
	St. Dev.	0,0530	0,0558	0,0597	0,0618	0,0618	0,0617	0,0590
3h	**IoU Avg.**	0,9130	0,9113	0,9075	0,9080	0,8988	0,8935	0,9054
	St. Dev.	0,0455	0,0493	0,0506	0,0491	0,0550	0,0577	0,0512
4h	**IoU Avg.**	0,9136	0,9114	0,9100	0,9098	0,9093	0,9073	0,9102
	St. Dev.	0,0495	0,0522	0,0545	0,0528	0,0524	0,0538	0,0525

Table 7.9: IoU average and standard deviation by predicted frame for different time intervals between frames. The training set and test set were preprocessed with edge detection.

Frame		1	2	3	4	5	6	Avg.
1h	**Cell Area Avg.**	1,0447	1,0380	1,0397	1,0422	1,0464	1,0407	1,0419
	St. Dev.	0,0465	0,0448	0,0460	0,0428	0,0473	0,0446	0,0453
2h	**Cell Area Avg.**	1,0103	1,0130	1,0300	1,0381	1,0368	1,0329	1,0269
	St. Dev.	0,0368	0,0415	0,0556	0,0599	0,0520	0,0479	0,0489
3h	**Cell Area Avg.**	1,0318	1,0392	1,0412	1,0406	1,0258	1,0207	1,0332
	St. Dev.	0,0496	0,0551	0,0625	0,0597	0,0745	0,0864	0,0646
4h	**Cell Area Avg.**	1,0565	1,0557	1,0578	1,0605	1,0592	1,0552	1,0575
	St. Dev.	0,0517	0,0476	0,0523	0,0545	0,0542	0,0615	0,0536

Table 7.10: Predicted cell area (normalized to the GT cell area) average and standard deviation by predicted frame for different time intervals between frames. The training set and test set were preprocessed with edge detection.

Frame		1	2	3	4	5	6	Avg.
1h	**Precision Avg.**	0,9357	0,9395	0,9384	0,9368	0,9333	0,9341	0,9363
	St. Dev.	0,0416	0,0406	0,0420	0,0403	0,0424	0,0400	0,0411
2h	**Precision Avg.**	0,9472	0,9456	0,9371	0,9325	0,9325	0,9331	0,9380
	St. Dev.	0,0399	0,0440	0,0523	0,0560	0,0529	0,0503	0,0493
3h	**Precision Avg.**	0,9406	0,9366	0,9341	0,9344	0,9365	0,9365	0,9364
	St. Dev.	0,0420	0,0464	0,0497	0,0470	0,0467	0,0471	0,0465
4h	**Precision Avg.**	0,9302	0,9291	0,9277	0,9264	0,9267	0,9276	0,9280
	St. Dev.	0,0463	0,0464	0,0497	0,0478	0,0466	0,0483	0,0475

Table 7.11: Prediction precision average and standard deviation by predicted frame for different time intervals between frames. The training set and test set were preprocessed with edge detection.

Frame		1	2	3	4	5	6	Avg.
1h	**Recall Avg.**	0,9757	0,9735	0,9739	0,9747	0,9747	0,9705	0,9738
	St. Dev.	0,0160	0,0161	0,0151	0,0155	0,0165	0,0204	0,0166
2h	**Recall Avg.**	0,9558	0,9564	0,9626	0,9649	0,9643	0,9617	0,9609
	St. Dev.	0,0252	0,0251	0,0221	0,0193	0,0223	0,0262	0,0234
3h	**Recall Avg.**	0,9686	0,9710	0,9697	0,9698	0,9578	0,9526	0,9649
	St. Dev.	0,0200	0,0185	0,0198	0,0215	0,0425	0,0525	0,0291
4h	**Recall Avg.**	0,9804	0,9788	0,9788	0,9801	0,9792	0,9762	0,9789
	St. Dev.	0,0139	0,0149	0,0128	0,0163	0,0186	0,0236	0,0167

Table 7.12: Prediction recall average and standard deviation by predicted frame for different time intervals between frames. The training set and test set were preprocessed with edge detection.

Data Augmentation variation

The following table shows the averages for the following metrics: IoU, Cell area (normalized to GT cell area), precision and recall. The dataset from which the used training sets were generated consisted of 644 training videos and 83 test videos. The training was performed on the control (whole dataset), mirrored (half the dataset, original and mirrored along vertical axis), no augmentation (half the dataset) and rotated (half the dataset, original and rotated 90 degrees counter-clockwise) training set. All models were trained on the same test set containing all 83 videos.

Dataset		**Control**	**Mirrored**	**Rotated**	**No Augmentation**
IoU	**Average**	0,9508	0,9488	0,9458	0,9402
	St. Dev.	0,0368	0,0364	0,0566	0,0445
Cell Area	**Average**	1,0108	0,9899	0,9777	1,0126
	St. Dev.	0,0194	0,0160	0,0467	0,0316
Precision	**Average**	0,9695	0,9785	0,9819	0,9633
	St. Dev.	0,0263	0,0214	0,0274	0,0345
Recall	**Average**	0,9796	0,9685	0,9611	0,9745
	St. Dev.	0,0153	0,0205	0,0500	0,0184

Table 7.13: Prediction IoU, normalized cell area, precision and recall averages for models trained with control (whole dataset), mirrored (half the dataset, original and mirrored along vertical axis), no augmentation (half the dataset) and rotated (half the dataset, original and rotated 90 degrees counter-clockwise) training set. All models were trained on the same test set.

List of Acronyms

PDAC	pancreatic ductal adenocarcinoma
AI	artificial intelligence
NLP	natural language processing
ML	machine learning
RL	reinforcement learning
ANN	artificial neural network
CNN, ConvNet	convolutional neural network
RNN	recurrent neural network
fRNN	folded recurrent neural network
GRU	gated recurrent unit
bGRU	bijective gated recurrent unit
rCNN	recurrent convolutional neural network
AE	autoencoder
CrevNet	conditionally reversible network
cGRU-AE	convolutional gated recurrent unit autoencoder
3D-cED	3D convolutional encoder-decoder
GAN	generative adversarial network
PGGAN	progressive growing generative adversarial network
DL	deep learning
DNN	deep neural network
ReLU	rectified linear unit
LReLU	leaky rectified linear unit
LSTM	long-short term memory
RGB	red, green and blue
CV	computer vision
GPU	graphics processing unit
AV	autonomous vehicle
NMR	nuclear magnetic resonance
MSA	multi sequence alignment
PDB	protein data bank
ST-convLSTM	convolutional LSTM in the spatio-temporal domain

DMEM	Dulbecco's Modified Eagle's Medium
FBS	fetal bovine serum
P/S	Penicillin/Streptomycin
LAS X	Leica Application Suite X
LIF	Leica Image File
TIFF, TIF	Tagged Image File Format
CPU	central processing unit
RAM	random-access memory
FFT	fast Fourier transform
P	positive
N	negative
TP	true positive
FP	false positive
TN	true negative
FN	false negative
IoU	Intersection over Union

List of Figures

List of Tables