Artificial Video Generation for Improved Performance on Polyp Detection

Oda Olsen Nedrejord

Thesis submitted for the degree of
Master in Robotics and Intelligent Systems
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2020

Artificial Video Generation for Improved Performance on Polyp Detection

Oda Olsen Nedrejord

Artificial Video Generation for Improved Performance on Polyp Detection

http://www.duo.uio.no/

Printed: X-press printing house

Abstract

Colorectal cancercolorectal cancer (CRC) is a widespread disease which is a threat to public health. Abnormalities in the colon, like polyps, can become cancerous. It is important to detect polyps early, in order to prevent a potential spread of cancer. Polyps can be overlooked during screening, and typically doctors have a polyp miss rate ranging from 14 to 30%. Several promising computer systems have been developed to help doctors lower their polyp miss rate. Obtaining a large, high quality dataset is important when building such a system, and the **lack of data is perhaps the biggest challenge in the field today.**

Data is arguably the most valuable resource in machine learning. Complex neural networks are dependent on great amounts of data in order to perform well. The colon is full of complicated structures, and a dataset should contain examples of as many examples of both healthy and unhealthy structures as possible. However, medical data is hard to get hold of due to legal restrictions and the cost of performing examinations. Currently a highly qualified, medical expert is needed to annotate data as well, further complicating matters.

We have developed a system which can take an existing dataset and use it to generate new, artificial data which can be added to the dataset. This will make it easier to create a large enough dataset for polyp detection systems. In other words, **we can generate real-looking videos of polyps.**

A total of 41 generated videos was provided to two medical experts, and they were asked to comment on the quality of the videos. Their comments revealed that shapes and colors in the videos look real. They additionally stated that they found these videos relevant for detecting other abnormalities in the colon. We also trained two polyp classifiers on the same dataset, but for one of the classifiers we also added our artificial videos. We found that the results were inconclusive, though we believe that it should be possible for the artificial videos to improve performance.

Acknowledgements

Jeg vil først å fremst takke mine veiledere for god hjelp og støtte gjennom et morsomt og lærerikt år. En spesielt stor takk til Michael Riegler, Pål Halvorsen, Steven Hicks og Vajira Thambawita som alltid har vært tilgjengelig for spørsmål og sparring gjennom hele året. Selv fra hjemme i stua under Corona-pandemien. Uten dere ville jeg ikke klart det!

Tusen takk til Augere Medical for at jeg fikk jobbe med dataene deres, og for et godt samarbeid.

Til slutt vil jeg takke min kjæreste, Thomas Børstad som har hatt troen på meg, helt fra jeg startet min 5-årige utdanning. Tusen takk for at du har ventet på meg og for all den fantastiske støtten du har gitt meg på veien.

Contents

List of Figures

List of Tables

Acronyms

CAD computer aided diagnosis. 1, 11, 12, 30

CE capsule endoscopy. 10, 11

CGAN conditional generative adversarial network. 4, 25, 26, 34, 85

CNN convolutional neural network. 22

CRC colorectal cancer. 1, 8, 9

GAN generative adversarial network. 23–25, 43

GI gastrointestinal. 7, 9, 11

GRU gated recurrent unit. 22

LSTM long short-term memory. 22

MLP multilayer perceptron. 15, 16

MSE mean square error. 68, 69

PCA principal component analysis. 14

PSNR peak signal to noise ratio. 68

RNN recurrent neural network. 21, 22

SSIM structural similarity. 68

VAE variational autoencoder. 23, 24

Chapter 1

Introduction

1.1 Motivation

Cancer is the second leading cause of death in the United States and a significant threat to public health [72]. CRC is the third leading cause of cancer-related deaths for both men and women in the United States [82]. Improvements in the treatment of CRC has given large declines in the overall cancer mortality. The CRC death rate has dropped with 53% for men since 1989 and 57% among females since 1969 [82]. However, in the last decade The American Cancer Society is reporting that progress has been slowing for cancer types that are dependent on early detection through screening [82].

Studies show that a good prognosis is associated with early diagnosis of CRC [36]. Colonoscopy is the gold standard of colorectal cancer screening methods. A colonoscopy is performed by inserting a camera into the rectum. A doctor who specializes in the digestive system, a gastroenterologist, will look for abnormalities in the colon. A colon polyp is an abnormality which a doctor will look for. Polyps can vary in size and can become cancerous. Polyps tend to be overlooked during colonoscopy examinations. van Rijn et al. report a polyp miss rate between 14 to 30% [100] during the examinations.

Researchers have developed several promising methods and systems for automatic detection [105, 74] to improve polyp detection. Such methods are referred to as computer aided diagnosis (CAD). Some of these systems as based on deep learning algorithms. These algorithms are complex and require large amounts of data to perform well. However, getting access to large amounts of endoscopy data is not a trivial task.

There are not many large scale endoscopy datasets available [7, 23, 69]. The available datasets often lack enough data or consist of a limited number of abnormal cases. It is challenging to collect clinical data due to privacy regulations. Furthermore, data annotation is a time consuming task, and it is additionally hard to get hold of the medical experts to evaluate the data. These challenges lead us to the aim of our thesis.

1.2 Problem statement

The ratio between normal and abnormal cases in colon data is often large. The abnormal cases are often present in few and short clips of long videos. While the normal cases are present in the remaining clips. Closing the gap between abnormal and normal cases remains a challenge. We attempt to close the gap by generating artificial videos of abnormal cases.

We aim to develop a system that can generate artificial videos. The goal is to increase the number of abnormal cases in a training dataset. The dataset that we have at our disposal, contains videos of normal cases and abnormal cases where colon polyps represent the abnormal cases. Although our work is limited to colon polyps, we aim for our system to be used in multiple use-cases, for example for generating videos of rarer cases like lymphoma. We want to introduce artificial videos that look realistic. Realistic in terms of the image quality and the camera and tissue movements should look real. Both these two parameters are equally important. The goal is to add a larger number of, and more variety to a set of annotated polyp videos. We want our generated polyp videos to contribute to a higher polyp detection rate. Based on our goals, the thesis aims the answer the following research question:

> *Can generative models be used to generate realistic-looking colon polyps?*

To answer this question, we define three objectives that focus on the practicalities of coming to a solution. Each objective builds on the objective that comes before it, and each objective is given a full chapter to be thoroughly discussed in this thesis.

Objective 1 Prepare the training data in a way that optimizes network leaning and avoids overfitting to specific video frames. This objective stems from the common need for quality training data when developing deep neural networks.

Objective 2 Generate artificial videos of colon polyps using conditional generative adversarial networks (CGANs). This objective is the main contribution of this thesis and comes from the requirement of needing more labeled data in the medical sector.

Objective 3 Perform a thorough evaluation of the generated videos using a quantitative and qualitative approach, in addition to evaluating the fake videos on a real-world use-case. This objective will give us the answer to our research question of whether or not the generated videos are of sufficient quality.

1.3 Scope and Limitations

We have decided to limit the scope of this thesis to focus purely on generating video sequences containing polyps. Although the GI tract contains many other lesions, polyps are among the most common and most dangerous as they may become cancerous if they are not removed in due time. Although the focus is on polyps, we expect that the methods presented here will generalize to other lesions as well. Furthermore, as we are limited by time and resources, the deep

learning models we have trained have been restricted in terms of training time and model size.

1.4 Research Methods

We have based our research method on the Association for Computing Machinery (ACM) method "Computing as a Discipline" [15]. The paper proposes a framework for the discipline of computing which consists of three major paradigms: *Theory*, *Abstraction* and *Design*. The thesis is based on a combination of the three paradigms.

1.4.1 Theory

The theory paradigm consists of four steps that are rooted in the field of mathematics. The fours steps include:

1. Characterize objects of study (definition).
2. Hypothesize possible relationships among them.
3. Determine whether the relationships are true (proof).
4. Interpret the results.

In the thesis, we use the theory paradigm to identify studies and challenges within the field of colorectal cancer. Moreover, we search for solutions and relationships between theories through different areas within machine learning, among others convolutional neural networks and generative modeling.

1.4.2 Abstraction

The abstraction paradigm is rooted in the field of the experimental scientific method and consists of four steps: *i)* Form a hypothesis. *ii)* Construct a model and make a prediction. *iii)* Design an experiment and collect data. *iv)* Analyze results. The design of our experiments falls under the abstraction paradigm. We designed and conducted our experiments in an iterative process which can be described as experimental prototyping.

1.4.3 Design

The third paradigm is design, and is rooted in engineering. The paradigm consists of four steps: *i)* State requirements. *ii)* State specifications. *iii)* Design and implement the system. *iv)* Test the system. Our complete system is based on the design paradigm. The complete system includes the data preparation, artificial video generation, and the final polyp classification, which includes the testing of the system.

1.5 Contributions

Our work is divided into three phases: data preprocessing, sequence generation and evaluating the quality of the generated sequences. Our main contributions is the work contained in first two phases.

Our research question in Section 1.2 was split into three objectives, and here we address the two main objectives:

Objective 1 Prepare the training data in a way that optimizes network learning and avoids overfitting to specific video frames.

We developed a data preprocessing framework that optimizes training videos by removing duplicate frames and large jumps in the videos. The method we used to achieve this was optical flow. We found the preprocessing to greatly improve the quality of our genrated video sequences.

Objective 2 Generate artificial videos of colon polyps using conditional generative adversarial networks (CGANs).

For sequence generation, we developed a sequence generator model: Vid2Pix. The model extracts spatiotemporal features by using 3D convolutions and deconvolutions in a conditional generative adversarial network (CGAN). We found that the model was able to successfully generate artificial polyp videos that could reduce the need for collecting data.

Kvasir-Capsule, a video capsule endoscopy dataset **[86]** In addition to the technical work presented in this thesis, we published the paper *Kvasir-Capsule, a video capsule endoscopy dataset* [86]. The paper presents a new dataset consisting of annotated capsule endoscopy videos.

1.6 Thesis Outline

The thesis is organized with a background chapter followed by three chapters that represent the three technical phases of our work. All three phases are described with separate sections consisting of methods, results and discussion. The closing chapter discusses the conclusion and future work. We structure the chapters in the following way:

Chapter 2: Background We give an overview of medical concepts and current machine learning advances within the field of colorectal cancer detection. We further discuss the basic concepts within machine learning and neural networks that build the foundation of our work. Next, we focus on generative modeling and sequence learning. Finally, we introduce related work in the field of image- and video generation which motivates our contributions.

Chapter 3: Preprocessing data for Sequence Generation In this chapter, we introduce the data preprocessing phase. In this phase we aim optimize data for sequence generation. We experiment and discuss different ways of optimizing a dataset before training a sequence generator. We further present and discuss the results from the preprocessing.

Chapter 4: Sequence Generation We introduce the model architectures that are the building blocks of our contribution. We further give a detailed description of our model contribution. Next, we describe and discuss the experimental process that led us to our final results. Finally, we introduce our

system for creating artifical video sequences. The aim is to increase the size and introduce new information to an existing training dataset.

Chapter 5: Sequence Evaluation In this chapter, we will conduct a thorough evaluation of the generated sequences. We will use both a qualitative approach, based on metrics, and a subjective approach relying on the expertise of two medical doctors. Next, we conduct a case study to evaluate our generated videos on a real-world use-case scenario. With this, the aim is to increase the performance of a polyp classifier by adding additional fake polyp videos to our training dataset.

Chapter 6: Conclusion and Future Work Finally, we provide a summary where we will discuss our main contributions and answer our research question and its objectives. We will further suggest ideas for future work.

Chapter 2

Background

In this chapter, we present the medical and technical background needed to understand our work and recognize its importance. First, we present some background on the GI tract and discuss some of the most common lesions that appear in this anatomical system. Then, we give an introduction to the machine learning techniques that are used and will be further mentioned throughout the thesis. We further discuss some of the related work around automatic disease detection and finally we address the benefits and shortcomings of related work within the field of future video prediction.

2.1 Medical Background

This section will give an introduction to the medical concepts and anatomical systems required to understand the remainder of this thesis. This includes a description of the gastrointestinal tract and some lesions commonly found in this organ system. We also cover the current standards for inspecting the GI tract and some related works where deep learning has been used for automatic detection of disease and other lesions.

2.1.1 Gastrointestinal tract

The gastrointestinal (GI) tract is together with the liver, pancreas, and gallbladder a part of the digestive system. The GI tract is divided into two main parts: The upper GI tract and the lower GI tract. The upper GI tract consists of the mouth, esophagus, and stomach. While the lower GI tract consists of the small intestine and the large intestine which includes the colon and rectum. A complete overview of the digestive system is shown in Figure 2.1. In the upcoming sections, we will be focusing on the lower GI tract.

Polyps in the Gastrointestinal tract

Polyps are present in different parts of the human body. A colon polyp is a clump of cells that can form on the inner lining of the colon. The majority of colon polyps are harmless, but some can become cancerous over time. If the colon polyps are not detected and removed early, the consequences can become fatal [11].

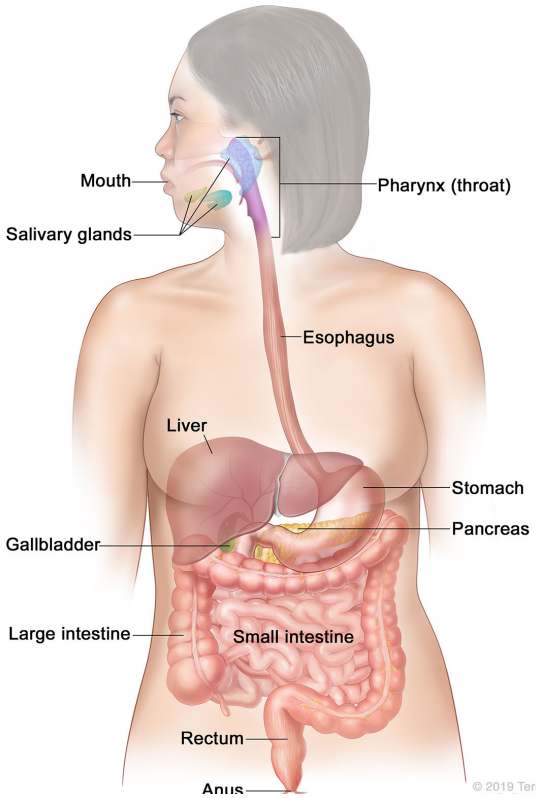


Figure 2.1: An overview of important organs in the gastrointestinal tract. Illustration: For the National Cancer Institute © (2020) Terese Winslow LLC, U.S. Govt. has certain rights

Colon polyps can be divided into two main categories: neoplastic and non-neoplastic. Neoplastic polyps include adenomas and serrated polyps. Adenomas are the most common type of polyps. The non-neoplastic polyps include hamartomatous polyps, inflammatory polyps, and hyperplastic polyps. These three types of non-neoplastic polyps typically do not become cancerous [11]. In this work, we do not distinguish between different types of colon polyps. We instead focus on creating a system that identifies all types of colon polyps to reduce the time for the medical experts and to prevent overlooking polyps. We leave it to the medical experts to decide whether a polyp should be removed or not.

Colorectal Cancer

The GI-tract can be affected by a variety of diseases, among these are CRC. CRC usually starts from a slowly growing polyp in the inner lining of the colon [89]. Cancer which grows from the inner lining of the colorectum is called adenocarcinoma, which accounts for about 96% of all CRC cases in the United States.

Staging of CRC is important for prediction the disease outcome and for determining treatment. A common staging system called: Surveillance, Epidemiology, and End Results (SEER) summary staging system, divides CRC

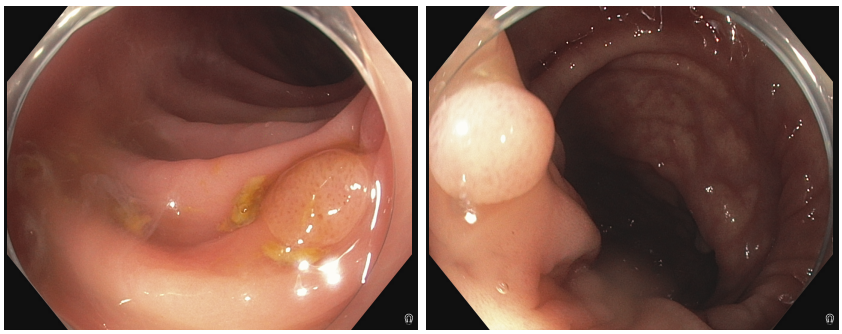


Figure 2.2: The figure shows two images of polyps. The images are a part of the OUS-Study-2019 dataset introduced in Section 3.1.

into four stages: *In situ*, *Local*, *Regional*, and *Distant*. On the *In situ* stage, cancer has not begun to invade the wall of the colon or rectum. The lesions are called preinvasive on this stage and is not accounted for in the cancer statistics. On the *Local* stage, cancer has begun to grow into the wall of the colon or rectum, but has not yet invaded nearby tissue. On the *Regional* stage, cancer has spread to nearby lymph nodes or invaded nearby tissues. On the *Distant* stage, cancer has spread to other parts of the body, such as the lungs or the liver.

Due to the slow growth from preinvasive polyps to invasive cancer, a unique opportunity arises to prevent and detect colorectal cancer at an early stage. To detect and prevent CRC, we need reliable screening methods, which will be introduced in the next section.

2.1.2 Screening Methods

Studies have shown that an early diagnosis is associated with a good prognosis [36]. There exists different screening methods used to detect CRC, and several countries have public health programs to screen for CRC [36]. There are different screening methods used to detect CRC. In Section 2.1.2, we will introduce the gold standard for colorectal cancer screening: gastrointestinal endoscopy. Furthermore, we will look into a more recent method called capsule endoscopy in Section 2.1.2.

Gastrointestinal Endoscopy

During a GI endoscopy examination, the endoscope is inserted directly into the organ, either the mouth (gastroscopy) or rectum (colonoscopy). The gold standard for the detection and removal of polyps is colonoscopy examinations. The endoscope or colonoscope is a long, flexible tube with a small camera at the tip of the tube which enables the doctor to look inside the colon while examining the patient. During an examination, the colonoscope is moving towards the small intestine, through the large intestine as shown in Figure 2.3. Abnormalities can be removed or biopsies can be taken during an examination, if necessary [12].

Colonoscopy examinations are known to be uncomfortable for the patient. According to a study on polyp miss rate [100], researchers found that polyps are often overlooked during colonoscopy examinations. The study revealed that

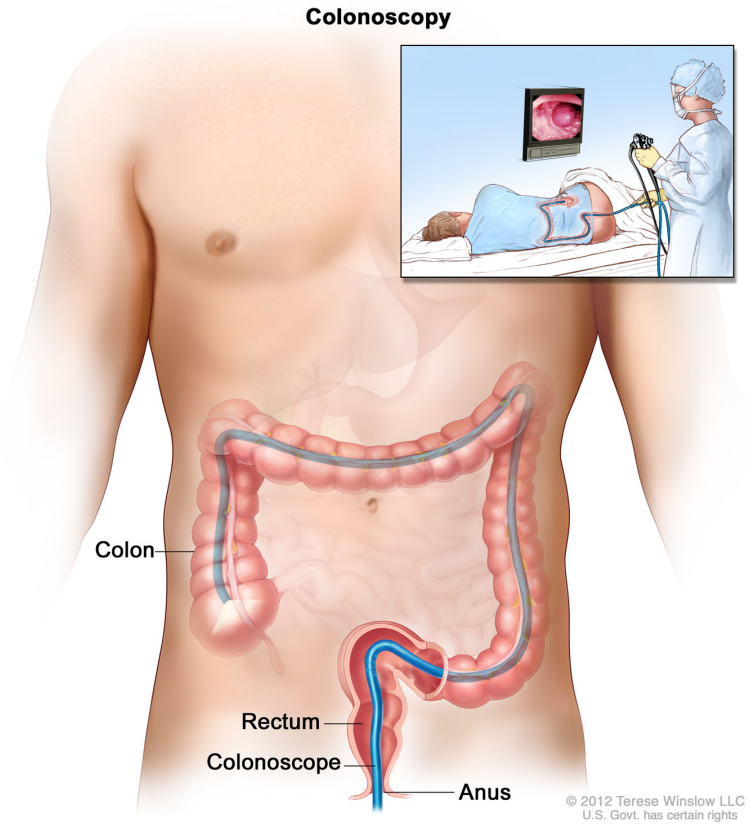


Figure 2.3: Example of a colonoscopy procedure. Illustration: For the National Cancer Institute © (2020) Terese Winslow LLC, U.S. Govt. has certain rights [13].

polyp miss rates were between 14 to 30 % where the type and size of the polyp had a triggering cause [100]. Thus a computer aided system could help to decrease the polyp miss rates.

Capsule Endoscopy

Capsule endoscopy (CE) is a gastrointestinal screening method performed with a wireless pill-sized camera that is swallowed and travels through the digestive system. An example of a capsule camera used in capsule endoscopy is shown in Figure 2.4. The camera is mainly used to examine the small intestine, but cameras have also been developed to examine the large intestine. According to Ding et al., capsule endoscopy has lately revolutionized how we perform examinations in the small intestine. In addition to making examinations less intrusive and easier, CE has also shown good results in detecting different types of cancer in the small intestine [16].

Colonoscopy examinations are time consuming and the qualified medical doctors are hard to get a hold of. In contrast, CE does not require any medical staff to be present while the pill camera travels through the digestive system.

Figure 2.4: The image shows an Olympus EC-S10 endocapsule. Photo: Sigrun Losada Eskeland, Bærum Hospital.

Furthermore, several computer programs have been developed to automatically skip hours of video recordings containing unwanted material to examine in the GI tract.

CE has problems related to low image resolution, low frame rate, and a narrow and uncontrollably field of view. These shortcomings can lead to abnormalities not being detected, especially those that initially are difficult to spot.

2.1.3 Computer Aided Diagnosis

Computer aided diagnosis systems are working as an aid for medical experts to reduce examination time and decrease miss rates for polyps and other abnormalities in the gastrointestinal tract. Automatic detection of polyps is a well researched area. There are currently numerous papers on CAD-systems that are developed to classify polyps or other abnormalities in the GI-tract[104, 41, 105, 74]. Wang et al. developed a real-time software system called "PolypAlert" [105]. The system assists endoscopists to detect polyps by giving visual feedback during colonoscopy examinations. The system managed to detect 97.7% of the polyps in 53 videos. Riegler et al. developed multimedia system named "Efficient computer aided diagnosis" (EIR) [74], the system managed to outperform existing systems in terms of real-time performance and resource consumption.

2.1.4 Deep Learning for Colorectal Disease Detection

Deep learning has shown promising results in classifying video endoscopy data in recent years [10, 99, 92, 65, 112, 101]. Research within the area has been conducted to develop both systems for post examination detection and real-time detection which is closely related to CAD-systems.

Owais et al. developed a deep learning system for video classification. The system is connected in three stages. First, they used an established deep convolutional neural network(CNN) model named ResNet18 for spatial feature extraction, followed by a Long-Short-Term Memory (LSTM) [28] model for temporal feature extraction and at last, a fully connected layer followed by softmax on the output of the LSTM. The data consists of 37 different classes from Gastrolab[1] and the Kvasir dataset[2] [69]. The system was able to achieve AUC value of 97.057%. Compared to existing baseline models, the overall sensitivity performance of the proposed method was higher based on average accuracy, f1 score, mAP, and mARendoscopic videos [65]. The proposed method additionally outperformed other handcrafted feature-based methods they compared it to.

Ding et al. attempts to establish another deep learning based system that can differentiate between two classes: abnormal images and normal images using small intestine capsule endoscopy data. To classify the images, the researches used a state-of-the-art deep CNN called ResNet152. 77 medical centers provided data from 6970 patients. The system was tested and compared to evaluations done by experienced gastroenterologists. With ResNet152, the researchers achieved a 99.88% sensitivity for identifying abnormalities in the per-patient analysis, and a 99.90% sensitivity for identifying abnormalities in the per-lesion analysis. In comparison, gastroenterologists achieved a sensitivity detecting abnormalities in the per-patient analysis with 74.57% and a sensitivity of 76.89% in the per-lesion analysis [16]. Reading time for the CNN was additionally drastically reduced compared to the gastroenterologists.

Ding et al. and Owais et al. work show promising results in classifying image and video endoscopy data. According to Kaminski et al., increasing polyp detection has been proven to decrease the risk of colorectal cancer [48]. Such systems could contribute to improve early detection, and further decrease the risk of colorectal cancer. Still, the papers express challenges about access to large datasets and the lack of data containing abnormal cases. In the next section, we will address the issue of medical data.

2.1.5 The Data Problem

There are few public large scale datasets available from the gastrointestinal tract [7, 23, 69]. In Table 2.1, Borgli et al. provide an overview of the existing GI datasets as of 2019. As we can see from Table 2.1, the available data is greatly sparse. Most of the datasets consists of few and low resolution images, and fewer of them consist of videos. The lack of data is mainly due to large annotation costs and legal restrictions for accessibility.

The lack of a common, high quality dataset makes it challenging to compare different approaches and to prove their promising results. The abnormal cases

[1] `http://www.gastrolab.net/`

[2] `https://datasets.simula.no/kvasir/`

Dataset	Focus	# Images	# Videos	Availability
CVC-356 [5]	Polyps	356	0	Not available
CVC-612/CVC-ClinicDB [6]	Polyps	612	0	Open academic
CVC-12k [5]	Polyps	11 954	0	Not available
ASU-Mayo polyp database [91]	Polyps	18 781	0	Not available
ETIS-Larib Polyp DB [83]	Polyps	196	0	Open academic
Kvasir-SEG [45]	Polyps	1 000	0	Open academic
GIANA'17 [4]	Angiectasia	600	0	By request
KID [52]	Multiple	2 500+	47	Not available
GASTROLAB [24]	Multiple	100+	360+	Open academic
WEO Endoatlas [108]	Multiple	152	0	By request
Colonoscopy Dataset [22]	Multiple	76	0	By request
Endoatlas [94]	Multiple	1 295	0	Not available
GastroAtlas [23]	Multiple	0	5 071	Open academic
Kvasir [69]	Multiple	8 000	0	Open academic
Nerthus [70]	Stool	0	21	Open academic

Table 2.1: Overview of existing GI datasets [7].

are especially crucial generalize a classifier to a variety of cases. We aim to solve these challenges by generating artificial polyp videos from annotated real polyp videos. This can potentially contribute to increase and generalize a training dataset. In the next section, we give a brief overview of the most typical groups within machine learning and describe some typical use-cases for the different groups.

2.2 Machine Learning

Machine learning is the study of statistical modeling and algorithms that enables machines to learn to do tasks based on patterns and inference instead of explicit instructions. There are many types of machine learning, but it is typically divided into three groups: Supervised learning, unsupervised learning and reinforcement learning.

2.2.1 Supervised Learning

Supervised learning involves learning a function to map inputs to its corresponding outputs by training the function on input-output pairs. Each training example consists of a pair, typically an input vector with a corresponding output value or supervisory signal which tells you what kind of feature your input is.

Classification and regression are two categories which fall under supervised learning. With regression, we try to learn a function to map from input variables to numerical or continuous output variables. While with classification we try to learn a function to map from input variables to discrete or categorical output variables.

Labeled data is necessary to perform supervised learning. Typical applications are: Image classification with the classic example of classifying cat images from dog images. In sentiment analysis, an application example is to teach a machine to determine the attitude or emotions in a text message.

2.2.2 Unsupervised Learning

Unsupervised learning is a self-organizing learning method that traditionally is used to find underlying structures or patterns in the input data. The model does not need any corresponding output variables to learn the patterns.

Traditionally, the most commonly used unsupervised learning task has been cluster analysis. Cluster analysis involves learning objects that are similar to each other and group or cluster them by similarity. This means that within a clusters, the cluster members need to have a smaller distance between each other. In cases where we have lots of unlabeled data, we can create clusters from the data which indicates which datapoint belongs to the same classes. Principal Component Analysisprincipal component analysis (PCA) is another popular unsupervised learning method that aim to reduce the dimensionality of the data by keeping important features of the data[110].

In recent years, generative modeling has become very popular, especially for generating images. Two methods that has shown great success within the field are Variational Autoencoders[51] and Generative Adversarial Nets[29] these methods will be further discussed in Section 2.4.3.

2.2.3 Reinforcement Learning

Reinforcement learning algorithms involve agents that are given a goal in a given environment and they are trying the maximize a reward. A reward is given if the agents manage to reach their goal or some sub-goals. If the agents do not manage to reach their goal or sub-goals, they will get penalized. Agents are in other words learning from the consequences of their actions.

Today's most common reinforcement learning algorithms are State-Action-Reward-State-Action (SARSA) [79], Q-learning [107] and a Deep Q Network (DQN) [31], which is based on Q-learning but also introduces a neural network.

Common applications within reinforcement learning are robot motion control, swarm intelligence [49] or solving games like chess with AlphaZero [84] and Atari that manged to outperform humans [62].

2.3 Feed Forward Neural Networks

A feed forward neural network is a type of artificial neural network. The information in this type of network is only moving forward. This network consists of a layer of input nodes, one or more layers of hidden nodes, and a layer of output nodes. To understand how deep neural networks work, we first need to look into how a simple neural network is built up. First, we will look into how the simplest kind of neural network called the perceptron is built in Section 2.3.1. Then we move further to the Multilayer perpectron in Section 2.3.2.

2.3.1 The Perceptron

The perceptron algorithm was first invented by Rosenblatt and funded by The United States Office of Naval Research in 1958 [76]. The perceptron is an algorithm for learning binary classifiers. A binary classifier is a type of linear classifier. This classifier is a function that takes a set of elements and decides which out of two groups each element belongs to. It is the simplest kind of

neural networks and it consists of only a single layer of output nodes and does not contain any hidden layers. The inputs are fed directly to the output nodes through their corresponding weights as shown in Figure 2.6. The function shown in Equation 2.1 is an example of a binary function that we use in the perceptron algorithm. The function maps its input x to a binary output which is either 0 or 1.

$$f(x) = \begin{cases} 1, & \text{if } \boldsymbol{w} \cdot \boldsymbol{x} + b > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

Figure 2.5: A function that maps its input x to a binary output value 0 or 1

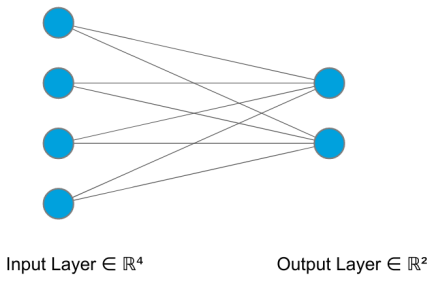


Figure 2.6: Example of single layer perceptron

2.3.2 Multilayer Perceptron

While a single layer perceptron can only learn linear functions, a multilayer perceptron (MLP) can also learn functions that are not linearly separable. If an MLP is using a linear activation function, according to linear algebra, all the layers can be reduced to a single layer model containing only one layer of output nodes. The Multilayer perceptron was first described by McCulloch and Pitts in 1943 [59]. In contrast to the single layer perceptron algorithm, the MLP algorithm consists of a minimum of three layers of nodes which is the input layer, one or more hidden layers, and the output layer. We train an MLP by using backpropagation [78].

2.3.3 Training a Neural Network

In the previous section, we presented how a simple neural network is built, first through the perceptron. Further, we looked into how an MLP is introducing non-linearity to a neural network. In this section we will cover the important parts that are missing to understand how a neural network learns through updating weights between connections.

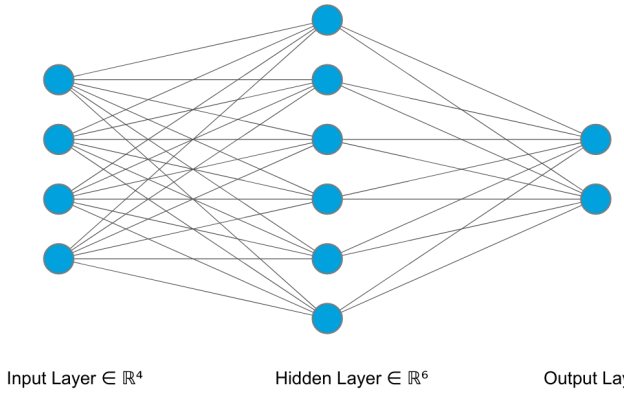


Figure 2.7: The figure shows an example of a multilayer perceptron consisting of an input layer with four neurons, a hidden layer with six neurons, and an output layer with two neurons.

Activation Functions

The objective of an activation function is to determine whether a neuron should fire or not. An activation function introduces non-linearity to our network and with that, prevents our network from doing linear mapping from input to output. Several activation functions are commonly used within the MLP domain and there are pros and cons of usage, often depending on the problem to be solved.

The sigmoid activation function is a differential function shown in Equation 2.2. It maps its input to a value between 0 and 1. The function can be useful if we want to predict the probability as an output.

$$g(x) = \frac{1}{1 + e^{-}x} \tag{2.2}$$

The rectified linear unit(ReLU) activation function is a differential function shown in Equation 2.3 and Equation 2.4. ReLU maps its input to 0 if the input is negative. If the input is above or equal to zero, the output will be the same as the input. A weakness with the functions is that all negative values become zero. This may affect the training of a neural network. As an attempt to solve the problem, Leaky Rectified Linear Unit Activation(Leaky ReLU) was developed.

$$g(x) = max(0, x) \tag{2.3}$$

or

$$g(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

Leaky Rectified Linear Unit (Leaky ReLU) was first introduced by Maas, Hannun, and Ng in 2013 [58]. Leaky ReLU is similar to ReLU when it comes to the objective of suppressing large negative values. While ReLU is using a hard zero when x is negative, Leaky ReLU uses a value close to zero: 0.01 that

is multiplied by the input x which is shown in equation (2.5) and (2.6). With this function, we get small negative values when x is negative, instead of a hard zero when x is negative.

$$g(x) = max(0.01x, x) \tag{2.5}$$

or

$$g(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.01x, & \text{otherwise} \end{cases} \tag{2.6}$$

To help our neural network to learn, we need a way to measure how it performs. A loss function is measuring the deviation between the true output and the predicted output for one sample of the data.

Cross Entropy

Cross entropy is a commonly used algorithm that calculates the difference between the ground truth and the predicted output. From Equation 2.7 on the left shows how we can calculate the cross-entropy for a multi-class classification problem. Since $N = 2$, the calculation becomes a binary cross-entropy problem, hence we can derive it into the equation shown to the right.

$$CE = -\sum_{i=1}^{N=2} \tilde{y}_i \, log(\hat{y}_i) = -(\tilde{y}_1 \, log(\hat{y}_1) + (1 - \tilde{y}_1) \, log(1 - \hat{y}_1)) \tag{2.7}$$

Where:

N	number of classes
log	natural log
$\tilde{y}_i$	true output of class i
$\hat{y}_i$	predicted output of class i

Optimizers

Our goal is to minimize the loss function by optimizing the weights in our network. With this, we use backpropagation. Backpropagation is a recursive computation technique that calculates the gradient of each node in a network by using the chain rule [78]. First, we initialize the weights, often by random initialization. Furthermore, the weights are updated based on how different or close the predicted output is to the ground truth. The method consists of a forward pass and a backward pass. There are several ways to optimize weights through the calculation of gradients. We will briefly present the two of the algorithms we used in our experiments.

Learning rate is an important tuning parameter that is used in optimization algorithms. The learning rate controls how much the weights should be adjusted with respect to the optimization algorithm. The value range between 0.0 and

1.0, where 1 indicates that the network should learn very fast, and 0.0 indicates that the network will not learn anything.

Stochastic gradient descent is another efficient optimization algorithm. The algorithm replaces the actual gradient with an estimate of the gradient for the entire dataset. When updating the gradient for each iteration, we get a computationally fast algorithm[53]. With SGD, we find the gradient of the cost function with a single sample of the data. One random sample from the data is chosen for each iteration. The SGD equation is shown below.

$$\theta = \theta - \eta \cdot \Delta_\theta J\left(\theta; x^{(i)}; y^{(i)}\right)$$

Where:

θ	model parameter
η	learning rate
$x^{(i)}$	training sample
$y^{(i)}$	label for each training sample
$\Delta_\theta J$	gradient of cost function

Adaptive moment estimation(Adam) [50] is an efficient stochastic optimization algorithm. It is an adaptive learning rate algorithm that by its name, adapts its learning rate for each weight in the network. More information on the optimizer can be found in the paper "Adam" [50].

Now that we have introduced the main components that are needed to train a neural network, we can describe the training process. The aim of training a neural network is to build a model of some data, in order to predict the output on new data. We first input a vector to the layer of input nodes. The input is fed forward through the network. The last layer in the network consists of an activation function, for instance a sigmoid function that decides whether nodes in the layer should fire or not. The output from the activation function is further used as input to a loss function, for instance the cross entropy algorithm. The loss function computes the error between the output from the network and the ground truth. Finally, through backpropagation, the loss is propagated back through the network in order to optimize the weights.

2.4 Deep learning

Deep learning is a field within machine learning that is based on neural network techniques. A deep learning algorithm represents data points by abstraction. Deep learning algorithms can be used within several machine learning fields, like supervised, unsupervised, and reinforcement learning. Deep learning networks are called "deep" due to the multiple layers that are included in the networks.

In this section, we will cover methods like generative modeling, sequence learning, and convolutional neural networks that we used actively in our experiments.

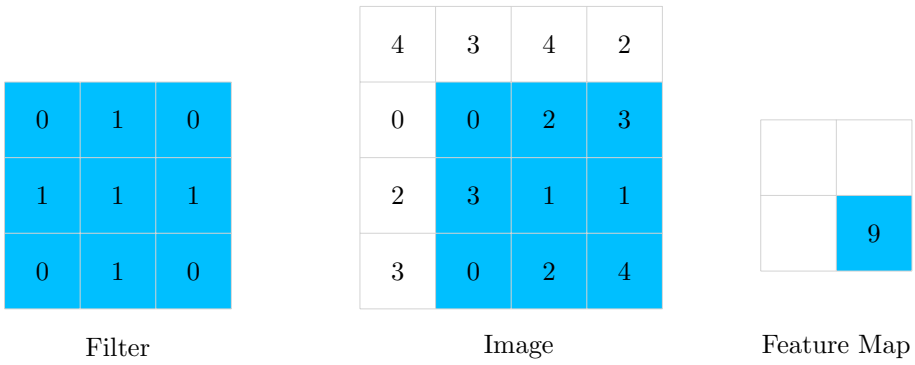


Figure 2.8: This diagram shows an example of a convolutional operation on a $4 \times 4 \times 1$ image using a kernel size of $3 \times 3 \times 1$ and a stride of 1.

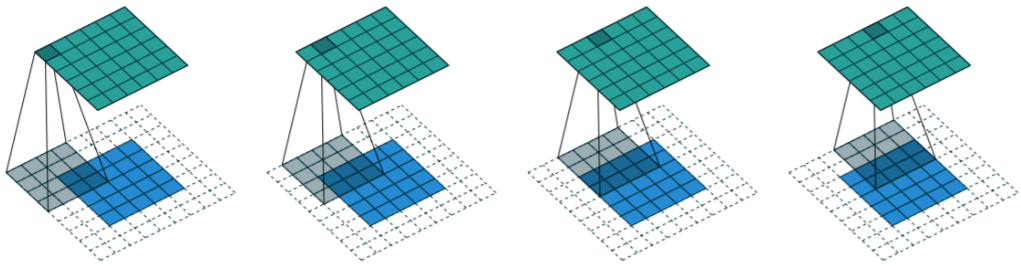


Figure 2.9: The figure shows an example of a 2D convolution with strides=2 and padding=2 [18].

2.4.1 Convolutional Neural Networks

Convolutional neural network is a subcategory within deep neural networks. CNNs are often applied to solve image classification tasks. The difference between a multilayer perceptron and a CNN is that neurons in a multilayer perceptron are fully connected. This means that each neuron or node in one layer is fully connected to all activations from the previous layer. While convolutional layers are only connected to a local region of the input. In addition, many neurons in the convolutional layers share parameters. We use convolutional layers on images to avoid the large number of parameters FCs are causing which can lead to overfitting.

Convolutional Layers

The convolutional layer is the main component of a convolutional neural network. An example of the operation conducted on one image is shown in Figure 2.8. During a forward pass we convolve the filters across the width and height of the whole input volume. In a convolution operation, we calculate the sum of element-wise multiplications between the image and the filter kernel.

Strides

We use strides to decide how we should slide or convolve our filter through our images. If stride is set to 1, we move the filter with one pixel at a time. If we set stride to 2, we skip two pixels at a time when we move our filter. This will produce a reduced spatial size of the output. For deconvolutional layers, we use strides in the opposite way by increasing the spatial size of the input if strides is set to be larger than 1. Strides can additionally be used in 3D convolutions or deconvolutions to reduce or increase the temporal size of the output respectively. An example of how an image is computed when stride is set to 2 is shown in Figure 2.9.

Deconvolutional Layers

A deconvolutional layer or a transposed convolution is used when we want our network to learn an optimized way to upsample our data [113]. An example of a deconvolutional operation is shown in Figure 2.10. Deconvolutions are for example used to reconstruct images.

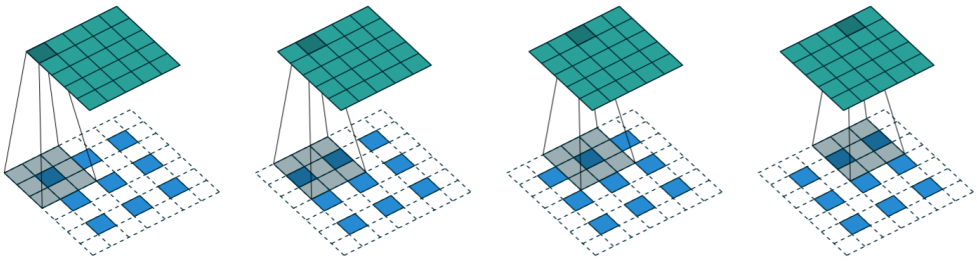
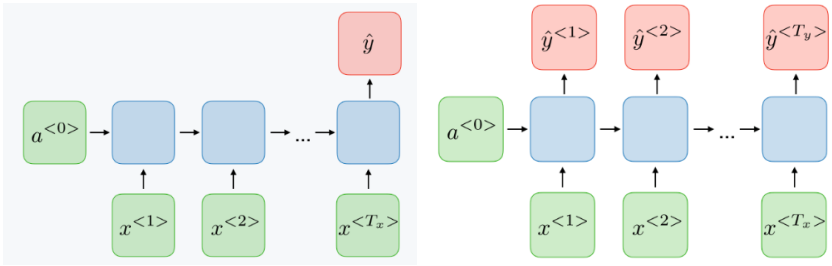


Figure 2.10: The figure shows an example of a 2D deconvolution using a 3×3 filter kernel on a 5×5 input with strides=2 and zero padding=1 [18].

Transfer Learning Transfer learning is a common method used in machine learning[66, 73], and it is often related to having an insufficient size of data. Transfer learning is a method of re-training a trained network. For example, we can train a model to learn to classify animals and transfer this knowledge to classify cats. Instead of training a CNN from scratch by randomly initializing weights, we can train a CNN on a large and generic image dataset like ImageNet(Section 5.2.1) and use the pre-trained weights as initialization or as a feature extractor for our new task. CNN as a feature extractor [66] holds a variety of ways to be performed. One way is to take a CNN pre-trained on for instance ImageNet(Section 5.2.1). Then we freeze n number of layers, which means that they will not be updated during training and will be treated as a feature extractor. Moreover, we remove the last fully connected layer and replace it with a few layers that are appropriate for the new task.

By fine-tuning CNN [66], we use the pre-trained weights as initialization to our new task. In the new task, we do not freeze any layers, but we continue to update weights through backpropagation. With this, we are optimizing all parameters for the new dataset. According to Yosinski et al., fine-tuning has



(a) Many-to-one architecture (b) Many-to-many architecture

Figure 2.11: Example of the common architecture of an RNN. The left figure (a) shows a many-to-one structure and the right figure (b) shows a many-to-many structure [14].

shown great improvement in data generalization, even on transferring features from distant tasks[111].

2.4.2 Sequence Learning

Recurrent Neural Networks

A classical problem within machine learning and statistics is the modeling of data sequences. There are various methods in machine learning that models data sequences. An example of a deep learning method for time series classification is recurrent neural network (RNN).

An RNN is a type of neural network that is often used when we have data sequences where the length of the sequence is varying or the order of the sequence is important. RNNs have a temporal dynamic behavior which is important when we want to learn sequences. For instance, an RNN can learn to predict the next word in a sentence, or the next image in a video. The main objective of an RNN is to learn what comes next in a sequence.

Vanilla RNN A vanilla RNN is the simplest kind of RNNs. What sets the vanilla RNN apart from more complex RNNs is that it does not have a cell state. In a video classification problem, we want to use previous knowledge h_{t-1} together with the current knowledge h_t that takes the current frame in a video x_t as input to determine for instance whether a person is moving or not.

The formula shows how the current hidden state in a simple RNN is calculated:

$$h_t = f_W(h_{t-1}, x_t)$$

Where:

h_t current hidden state

f_W function with parameters W

h_{t-1} previous hidden state

x_t input vector with time step t

RNNs are effective for learning shorter sequences, but they suffer from short-term memory which leads to a major drawback of learning longer sequences. Researchers have managed to solve this problem by introducing Long Short-Term Memory networkslong short-term memory (LSTM)[28] and Gated Recurrent Unit networksgated recurrent unit (GRU) [30]. Although LSTMs and GRUs have proven great performance in learning sequences, they are not as efficient and easy to combine with generative models. Generative models often use layers for downsamling and upsamling of data, which can be computationally heavy to combine with a LSTM or a GRU. In Figure 2.4.2, we can see two examples how we can structure an RNN.

3D Convolutional Neural Networks

3D convolutional neural networks(3DNNs) are closely related to 2D convolutional neural networks. The main difference is that 3DCNNs are built up of 3D convolutional layers instead of 2D convolutional layers. The 3D layers introduce an additional temporal dimension to the network.

With 2D convolutional neural networks (CNNs), convolutions are computed on 2D feature maps from the spatial dimensions only. 2D CNNs are often used to solve single image classification tasks [90, 9, 85, 38, 115, 81]. To solve a video classification task, we are additionally interested in the motion information from multiple contiguous frames.

With 3D CNNs we can extract both spatial and temporal feature maps from multiple frames. This means that in addition to learn the spatial shape of an object, we can learn the temporal shape and movement of an object based on temporal information with 3D CNNs. Ji et al. explain 3D convolutions in this way: "The 3D convolution is achieved by convolving a 3D kernel to the cube formed by stacking multiple contiguous frames together" [46]. A 3D convolution is computed by a kernel formed cube. As shown in Figure 2.12, the cube extracts features in the spatial and temporal dimensions by stacking multiple contiguous frames together.

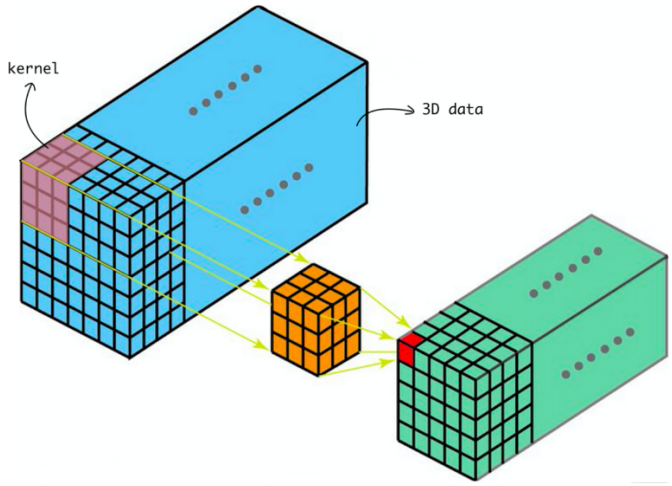


Figure 2.12: Example of a 3D convolution operation [98]

3D CNNs has proven to outperform 2D CNNs in various of video analysis tasks [95]. Applications like action recognition [46, 95] and medical 3D image segmentation [97, 19] have been developed using 3D CNNs. In the paper "Learning Spatiotemporal Features with 3D Convolutional Networks", Tran et al. suggest using $3 \times 3 \times 3$ filter kernels to achieve the best performance on action recognition.

2.4.3 Generative Modeling

> *A generative model describes how a dataset is generated, in terms of a probabilistic model. By sampling from this model, we are able to generate new data*
>
> *David Foster [20, p. 1]*

Generative modeling is an unsupervised learning method that is gaining popularity. Generative models are trained by learning the true data distribution of the training dataset so that it can generate new data with a similar distribution, also known as artificial or fake data. To be categorized as generative, the model must be probabilistic rather than deterministic, which means that it must include a stochastic element that can influence the individual samples to make them different from each other.

Generative adversarial networks (GANs) and variational autoencoders (VAEs) have recently given good results [51, 29]. Generative models are used in several different applications like music generation, text-to-image translation, and generating images of human faces.

Suppose we have a set of images of paintings. These paintings are all painted by the Norwegian artist, Edvard Munch. We want our generative model to learn the general rule that governs Edvard Munch's artistic style. The goal is to generate new images that look like real paintings by Edvard Munch. This is an example of a goal that we can work towards by using generative modeling.

Autoencoders

Autoencoders were first introduced in the 1980s by Hinton [35]. An autoencoder is a neural network that is trained to recreate a copy of its input. The purpose of an autoencoder is to discover low dimensional representations of the input data to capture some underlying structure in the high dimensional data.

A generic autoencoder consists of three main parts: an encoder network, a latent space representation, and a decoder network. The task of the encoder network is to compress high dimensional input data into a lower-dimensional, latent space representation. The decoder network is responsible for decompression of the low dimensional data back to its original shape. The goal is to learn a good encoding-decoding scheme through an iterative optimization process.

There are several variants of autoencoders, with different use cases. Some popular variants are the sparse autoencoder and the denoising autoencoder. The main idea behind the sparse autoencoder is to introduce a sparsity constraint in the hidden layer to prevent the output layer from copying the input data, this is done by allowing only a subset of hidden nodes to fire at the same time. The denoising autoencoder adds noise to the input data with the purpose to prevent the model from copying the input data, but instead learn the features of it.

Autoencoders show major flaws when it comes to introducing completely new data. Another, more promising variant of autoencoders when it comes to introducing new data, is the variational autoencoder which will be described in further detail in the section below.

Variational Autoencoders

The variational autoencoder was first introduced by Kingma and Welling in 2014 in the publication "Auto-Encoding Variational Bayes" [51]. Unlike the autoencoder that maps its input to a single point, the variational autoencoder maps the input to a multivariate Gaussian distribution in the latent space [20]. The steps are: the input is encoded to a multivariate Gaussian distribution containing the mean value μ and the variance σ^2 in the latent space. From the distribution, a point is sampled as a latent representation and is thereafter decoded to a reconstruction of the input. Finally, a reconstruction error is computed and backpropagated through the network

Variational autoencoders are using a statistical approach called variational inference to approximate complex distributions. This involves setting a parameterized Gaussian distribution, where the parameters are the mean and the covariance, and then search for the best approximation of the ground truth distribution among the Gaussian distribution.

VAEs are often used to encode data from the latent space. They have limitation on commonly producing blurry images and they are additionally not great on introducing new data samples. [71].

Generative Adverserial Networks

The GAN was first introduced by Ian Goodfellow et al. in 2014 [29]. The general idea of a GAN comes from game theory, the GAN is a type of minimax game between two adversaries, the discriminator D and the generator G. The discriminator can be seen as a police officer who tries to disclose which dollar bills are real or fake. In contrast, the generator can be seen as a counterfeiter. The generator's goal is to generate fake dollar bills to fool the police officer into mistaking fake dollar bills to be real.

The GAN architecture consists of three main parts: real training data, a generator model, and a discriminator model. The generator's goal is to create real looking data that can fool the discriminator, while the discriminator's goal is to distinguish between real and generated data. The generator and discriminator are trained simultaneously while trying to reach their conflicting goals.

Generator . The generator G is a neural network with a generator distribution $p_g(x)$ over the input data x. To learn the generator distribution $p_g(x)$, we introduce an input noise distribution $p_z(z)$, from this distribution we sample a random noise vector z. We use z to generate an image $x = G(z)$.

In mathematical terms: We have a mapping function that is a differentiable function $G(z; \theta_g)$ that takes a random noise vector z and a set of weights θ_g and maps z from the latent space to data space x. The output of the generator is of the same size as the samples from the real data distribution p_{data}. The goal of the generator is to fool the discriminator into predicting what is fake to be real. The objective of training the generator is to minimize $\log(1 - D(G(\boldsymbol{z})))$.

Discriminator The discriminator D is a neural network whose purpose is to correctly classify samples from the generator distribution $p_g(x)$ to be fake, and samples from the real data distribution $p_{data}(x)$ to be real. Throughout the training process, the discriminator learns to identify features that contribute to the real images. The discriminator is a function D that maps x to $D(x;\theta_d)$ where θ_d are the adaptive parameters.

The objective of training the discriminator is to maximize the log-likelihood of the probability $P(Y = y|x)$, where $y = 1$ if x comes from $p_{data}(x)$ and $y = 0$ if x comes from $p_g(x)$ [29].

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \quad (2.8)$$

Mode Collapse When we generate data from a GAN, we usually expect it to produce a wide variety of outputs. If a generator model shows a tendency to produce one, or a small set of similar outputs repeatedly, we can assume that the model is suffering from mode collapse. Regardless of what input noise we feed into the generator, there is a fixed optimal point that the generator repeatedly generates. This happens when the discriminator gets stuck in a local minimum. This causes the generator to over-optimize for a specific discriminator, and the discriminator never manages to get out of the local minimum [60].

Oscillating Loss When we train a GAN, we expect small oscillations in loss between batches. Moreover, we expect that the loss gradually stabilizes, decreases or increases during training. A common challenge with GANs is that the loss of the generator or the discriminator starts to oscillate wildly.

A successful approach that attempts to prevent the generator from optimizing for a particular discriminator, is to implement Wasserstein loss [2]. Wasserstein loss replaces the binary cross-entropy loss that is used in a GAN. Other approaches, like reducing the complexity of the discriminator or adding noise to the input of the discriminator has turned out to greatly stabilize a GAN [1, 77].

Conditional Generative Adversarial Nets

While GANs can learn to map from a noise distribution $p_z(z)$ to an output image x, CGANs can learn to map from a noise distribution $p_z(z)$ and an input image y to an output image x.

The CGAN works as follows: the condition y is fed as an additional input layer into both the generator and the discriminator. In the generator, the condition y and the input noise distribution $p_z(z)$ are combined together as a hidden representation. In the discriminator x and y are both represented as inputs [61].

The method was first proposed by Mirza and Osindero, and has become popular for doing image-to-image translation. For example by mapping segmented images to photos. As with a GAN, the generator in a CGAN also tries to fool

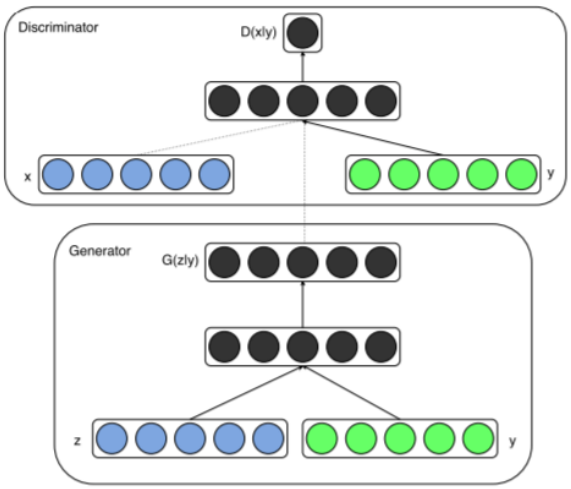


Figure 2.13: The figure shows an example of the structure of a simple conditional generative net [61].

the discriminator. The objective function of the two-player minimax game for a CGAN is shown in Equation 2.9.

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))] \tag{2.9}$$

[25] [44]

2.4.4 Data Handling

Dataset

In the context of this thesis, a dataset is referred to as a collection of videos X that corresponds to a set of classes Y that are either polyp or normal mucosa. The dataset is divided into three independent datasets that all have different use cases.

Training set. In supervised learning, the training set consists of a set of X and corresponding Y pairs that has the purpose of training a model to learn to predict new Ys without any help. In unsupervised learning, a training set consists of a set of X that is used to train a model to learn a pattern.

Validation set. A validation set is usually used during the training of a neural network. It is independent of the training and test set and has the purpose of evaluating generalization and overfitting during training. It is additionally used for hyperparameter tuning.

Test set The test set is independent of the training and the validation set. It should not be used during training and has the purpose of evaluation the model on completely new or unseen data.

Regularization

A common challenge for image classification tasks is the lack of large and generalized datasets. When a model performs well on training data but is generalizing poorly to new data, we refer to it as overfitting[32]. Hawkins describes that overfitting often occurs when our model is more complex than what is needed for the dataset we have. Overfitting can occur if we lack training data or if the training data is not generalized enough. We will now present some methods for avoiding overfitting

Data Augmentation Data augmentation can be beneficial to increase the amount of data by making copies that are changed slightly from their original. One image can be turned into multiple images by for example cropping, rotating, or flipping the image.

Batch Normalization A common challenge when training a deep neural network is to keep the weight values within a reasonable range. If the weights grow too large, it can lead to an exploding gradient problem [43]. To prevent this, batch normalization is often used[43]. With batch normalization we calculate the mean and standard deviation of the input across the batch. We then normalize the output of the layer by subtracting the mean from the input and dividing by standard deviation. Finally, we scale and shift the normalized input by the learned parameters gamma and beta respectively [20, p. 51-52].

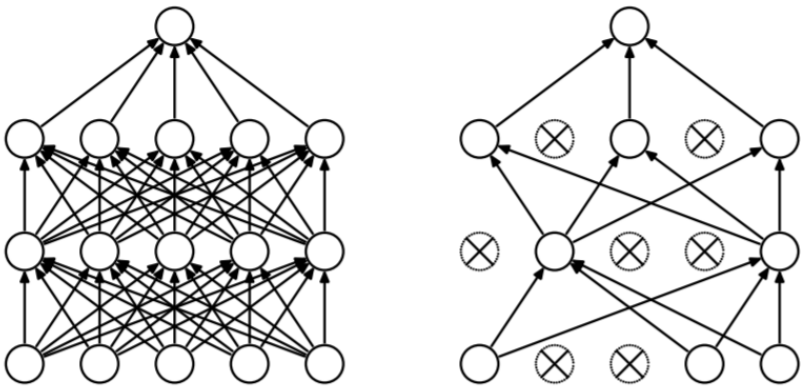
Dropout A common way to regularize and prevent a model from overfitting in a deep neural network is by using dropout [87]. The term dropout refers to dropping out nodes in a neural network. On the left in Figure 2.14, we see a standard neural network where all nodes are fully connected. On the right, we see a thinned neural network where the crossed nodes have been dropped. With dropout, a random set of units are temporarily deactivated and with that, all incoming and outgoing connections are stopped. The remaining units pass unchanged.

2.5 Related Works

There are different approaches and goals for video generation. Some are trying to develop self-driving cars and others are trying to create systems for activity recognition in smart homes. Research within the field is developing fast.

2.5.1 Image-to-image Translation

Recent research in the field of image-to-image translation [39, 44, 56] has shown great success. The method is used to translate an input image in one domain to a corresponding output image in another domain. The input image and the translated output image are often related. With this method, we can for



(a) Standard Neural Net (b) After applying dropout.

Figure 2.14: The figure shows a standard neural net with two hidden layers on the left On the right we see an example of a thinned neural net by applying dropout. Crossed nodes are dropped [87].

example translate an image of a young person to an image of the same person that looks old. Our approach is based on this method, but is in contrast opposed to generating videos. The results of an image-to-image translation task by Pix2Pix [44] are shown in Figure 2.15.

2.5.2 Unconditional Video Synthesis

is a method based on regular GANs, but is different in the way that it is generating videos. The method takes latent vectors as input to the generator network and tries to generate videos. There are no conditional images used as input here.

Tulyakov et al. developed a Motion and Content decomposed Generative Adversarial Network (MoCoGAN) which is mapping a sequence of random vectors to a sequence of video frames. The researchers developed both a motion and an image discriminator to learn content and motion. The model achieves competitive results compared state-of-the-art approaches [96]. Due to being unconditional, images tend to result in low-resolution videos and lower image quality [96, 17] compared to conditional generative neural nets.

These methods for image generation have successfully been used to generate single images [21]. By generating data based on a single input image, the generated image will not depend on a previous *sequence*, it will only be dependent on the previous *image*. Hence, the generated image will not necessarily achieve the same natural direction of motion, or a natural variation of the structure as with a sequence generator.

2.5.3 Video-to-video Synthesis

uses a sequence of images in one domain as a conditional input and generates a sequence of images in another domain as output. For example, with vid2vid

Figure 2.15: The figure shows two examples of image-to-image translation by Pix2Pix [44]

Wang et al. are using a sequence of segmentation masks as input and outputs a sequence of photorealistic images [102].

Coherent online video style transfer (COVST) [8] is closely related to the video-to-video synthesis method and has proven competitive results by generating temporally coherent stylized video sequences in near real-time. A comparison of three state-of-the-art methods is shown in Figure 2.16

The objective of video-to-video synthesis is to translate a video in one domain to a video in another domain. While our objective is to translate a video in one domain to future video frames in the same domain, this leads us to the method of future video prediction.

2.5.4 Future Video Prediction

Future video prediction uses a sequence of images to predict future frames in the same sequence. Several promising methods have been developed to predict future frames [63, 68, 47, 26, 55, 57, 54].

PredNet is one state-of-the-art algorithm for predicting future frames in a video sequence [57]. The method uses convolutional LSTM to robustly learn to predict object movements from both synthetic and natural image sequences. The method additionally proves to scale to other applications like capturing movements of objects from a car-mounted camera.

An algorithm developed by Lee et al. combines latent variational variable models that explicitly model underlying stochasticity together with adversarially-trained models that aim to produce naturalistic images. The two combined methods are used to generate videos of a moving robotic arm and different human activities. The method manages to produce realistic looking predictions

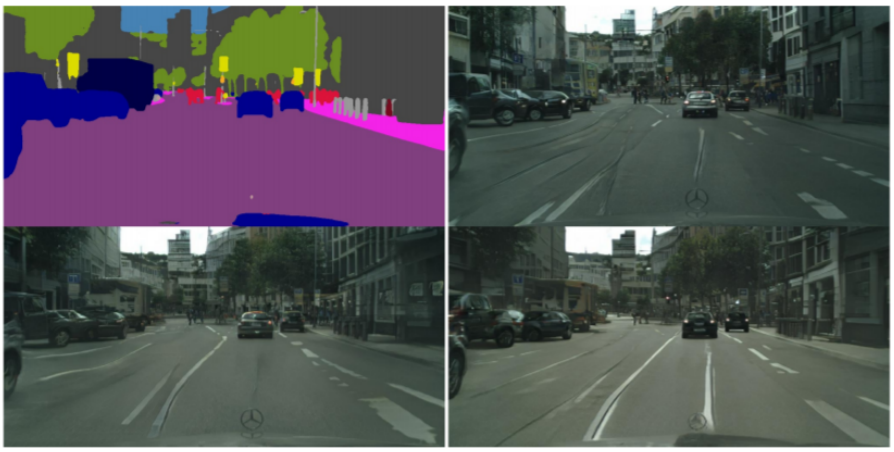


Figure 1: Generating a photorealistic video from an input segmentation map video on Cityscapes. Top left: input. Top right: pix2pixHD. Bottom left: COVST. Bottom right: vid2vid (ours). *Click the*

Figure 2.16: The figure shows comparison of three state-of-the-art video-to-video synthesis methods[8]

to human raters and by similarity measurements. The algorithm manages to outperform prior work in the field of future video prediction.

2.6 Summary

In this chapter, we have addressed medical concepts and screening methods for detecting colorectal cancer and other abnormalities in the colon. We have further described the importance of early detection of polyps to lower the risk of cancer mortality. We discovered the benefits of CAD-systems for aiding doctors in decreasing polyp miss rate during colonoscopy examinations. By adding more data, we found a large potential of existing advances in CAD-systems. Moreover, we discovered that medical data is sparse and that there is a great need for more medical data. The lack of data further motivates to build a system that can generate more data. In the last part of the chapter, we address machine learning algorithms that are used to build our systems. Finally, we look into work that is closely related to ours. We have seen that the most critical issue for improving these systems is the lack of data.

In the upcoming chapters, we will address three separate phases that together make up our overall solution. The phases are divided into three chapters. In the next chapter, we will introduce the OUS-Study-2019 dataset 3.1 that we use throughout our experiments. Then, we will describe challenges with our dataset, and we will introduce the optical flow 3.3.3 method which is used to improve our data. Moreover, we will look at experiments where we optimize the training data in order to generate realistic looking sequences.

Chapter 3

Preprocessing Data for Sequence Generation

In the previous chapters, we discussed the challenges concerning the need for more data and the large gap between data containing healthy tissue and data containing abnormalities in the colon. We have further seen the importance of discovering abnormalities at an early stage in order to prevent cancer from spreading. Furthermore, we have looked into previous studies on the detection of polyps and other abnormalities that have proven promising results. A common weakness for these studies is the accessibility to datasets and the amount of data with an emphasis on the lack of data containing abnormalities.

To solve these challenges, we have developed a system that can create more data from data we already have. Specifically, we aim to generate entirely new videos of colon polyps from our existing videos of colon polyps. Our system can be broken down into four distinct steps:

1. Remove duplicate frames from a dataset before sequence generation.
2. Train a model to generate video sequences of polyps.
3. Create fake video sequences from the generator model.
4. Train and test a polyp classifier based on real and fake data.

In this chapter, we will first give a full overview of the datatset that have been broken into two separate datasets. We will further give a detailed description of the first part of the dataset that we are using in this chapter. Moreover, we will look into the methods we use on data preprocessing, followed by the experimental process and results from this phase. The aim is to optimize data for sequence generation, by removing duplicates in colonoscopy videos.

3.1 OUS-Study-2019

In this section, we will introduce the in-house dataset that we have used throughout our thesis. The dataset is divided into two separate parts. We name the first part: *GEN-DAT* dataset that is used to train a polyp sequence generator. We name the second part: *CLSF-DAT* that is used to train a polyp classifier.

Each part of the dataset are treated as two separate datasets. In this section we will describe the dataset as a whole before we go into details on the *GEN-DAT* dataset that we are using in this chapter.

High resolution colonoscopy video data is hard to get hold of. Public available and high resolution datasets like HyperKvasir [7] and Kvasir [69] mainly consists of images and not videos. GastroAtlas is a dataset consisting of multiple videos, but with lower resolution [23]. The lack of video data and high resolution images, led us to search for an in-house video colonoscopy dataset. OUS-Study-2019 is a confidential dataset provided by the Norwegian health technology company: Augere Medical[1]. The dataset contains colored videos taken from the GI tract, which were collected between August 2019 and February 2020 from two hospitals in Oslo; The National Hospital (Rikshospitalet) and Ullevål University Hospital. All videos are recorded with endoscopic equipment and annotated by 31 experienced medical doctors.

The full OUS-Study-2019 dataset consists of a 83088 polyp images and 83088 normal mucosa images. The video resolution is 1380×1080 pixels with 3 color-channels. Due to the limited time and resources to run all experiments, we downsize and crop the images to 128×128 pixels with 3 channels. The videos in the dataset have varying lengths. The video lengths vary from a minimum of 6 minutes, and up to 2 hours. The average length of a recording is about 20 minutes and the frame rate is of 50 frames per second.

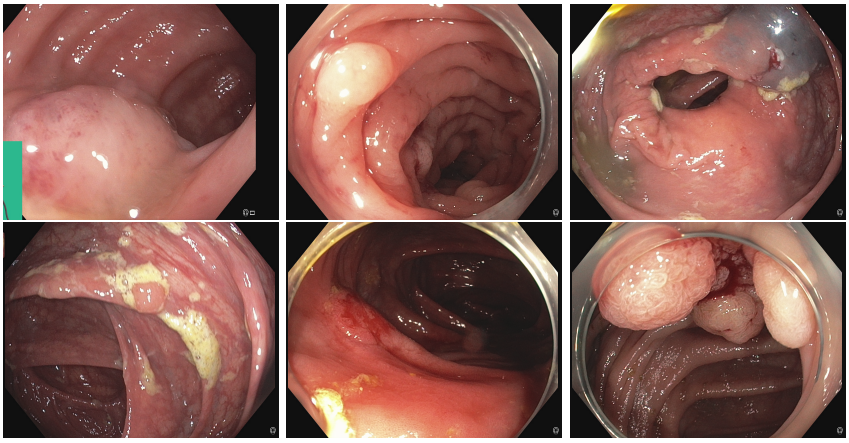


Figure 3.1: The figure shows six images from the polyp class. The images are obtained from the OUS-Study-2019 dataset.

The content of the dataset is equally split into two classes: polyp and normal mucosa. All videos in the normal mucosa class can be characterized as pseudo normal since the videos have not been manually annotated. To be included in the pseudo normal mucosa class, the videos must meet the following requirements:

- The patient is below age 45.
- The examination has reached the Cecum.

[1] Augere Medical `https://augere.md/`

- There must be no findings during examination.
- No polyps were found during the examination.

A few samples obtained from the normal mucosa class of the OUS-Study-2019 dataset is shown in Figure 3.2.The polyp class includes videos that exclusively contain colon polyps which are manually annotated by experienced medical doctors. A few samples obtained from the polyp class of the dataset are shown in Figure 3.1.

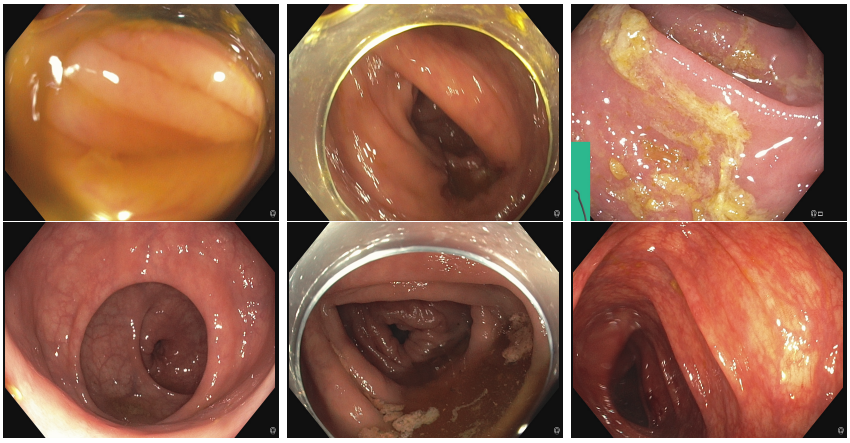


Figure 3.2: The figure shows six images of the normal mucosa class. The images are obtained from the OUS-Study-2019 dataset.

For the preprocessing experiments we use the *GEN-DAT* dataset. Our model requires a fixed input size. Based on the model requirements, we have split the videos in the dataset into 8 frames long video sequences. If some video sequences are shorter than 8 frames, they are removed from the dataset. From the 8 frames, we use 4 frames as input to the model and 4 frames as ground truth. This means that all videos that are shorter or do not add up with the fixed length are removed.

3.2 Problem and Goal

One challenge is that the motion contained in 8 frames of video, can be very different from video to video. If the camera is standing still, then all 8 frames will be nearly identical. The faster the camera moves, the less overlap there will be from frame to frame. This is further complicated by the fact that the operator can stop the camera in order to inspect an area of interest.

A high frame rate of 50 frames per second, combined with inconsistent movements causes some neighbor frames to look identical and others to appear more different. This leads to some video clips containing a lot of duplicate frames, while others contain few. These outcomes are based on two scenarios:

1. When the recorder is not moving, we get continuous frames that are very similar.

2. When the recorder is moving, continuous frames are more different and we get fewer duplicates.

We developed a prototype CGAN to test if duplicate frames would cause errors on the output. We discovered that irregular motions in the videos posed challenges for predicting the next frame. We tested this by inputting duplicate continuous frames as conditional input to our network. This resulted in duplicate output frames. We discovered a tendency that by using the unrefined data as input to our generator, the network generated videos that were too similar to the input. With this, it did not introduce any new data samples. Instead, we attempt to introduce new data from previous frames by making the difference between each frame in a video somewhat larger.

To create a more diverse dataset, we performed a series of preprocessing experiments to optimize the dataset for more consistent motions by removing duplicates and large jumps in videos. Our goal is to preprocess a dataset that contribute to introduce new and consistent data samples. The data samples will be used in our future sequence generator experiments that are introduced in Section 4.4. To solve this, we decide to use a popular method for estimating motion in videos, called optical flow.

3.3 Preprocessing Experiments and Results

To solve our challenges concerning duplicate frames, we experimented with different approaches. This resulted in a data processing framework that is a combination of some of the approaches.

In this section, we discuss the results from the three methods we tested for preprocessing of the dataset. Furthermore, we discuss their advantages and limitations and how we proceed to our final solution of using dense optical flow.

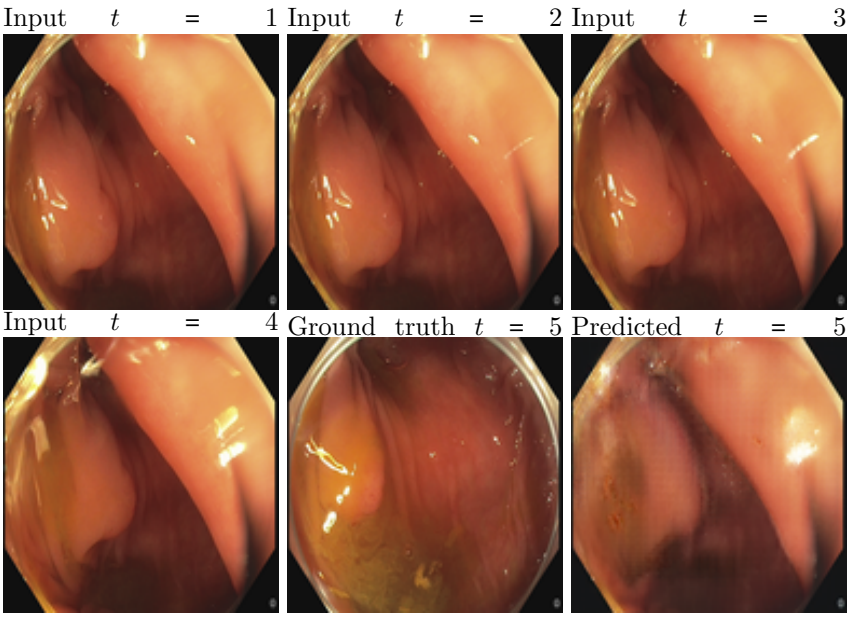


Figure 3.3: The figure shows a sample of four original input frames, from the OUS-Study-2019 dataset. The Figure further shows the the ground truth and the predicted output. The output is predicted using Vid2Pix introduced in Section 4.3.

In this phase, all videos in each preprocessed dataset are different due to the preprocessing. Based in this, we are not able to compare each experiment through similar videos. Instead we compare them by random sampled videos. We additionally set as equal terms as possible. We do this by reducing the number of video samples from the original dataset to the number of video samples we get from optical flow. In order to approach a more representative distribution of each dataset, we further shuffle the videos in the original and the preprocessed datasets. All experiments start out with a dataset consisting of a total of 28932 colonoscopy images from the *GEN-DAT* dataset described in Section 3.1.

Method	Number of Videos	Number of Frames	Videos for Experiment
Original	3553	28932	348
Skip 8	248	1984	248
Abs sum of diff	260	2080	260
Optical flow	348	2784	348

Table 3.1: The table shows an overview of the resulting number of videos and frames for each method after preprocessing of the *GEN-DAT* dataset. The right column shows an overview of the number of videos we used in the experiment from each preprocessed dataset.

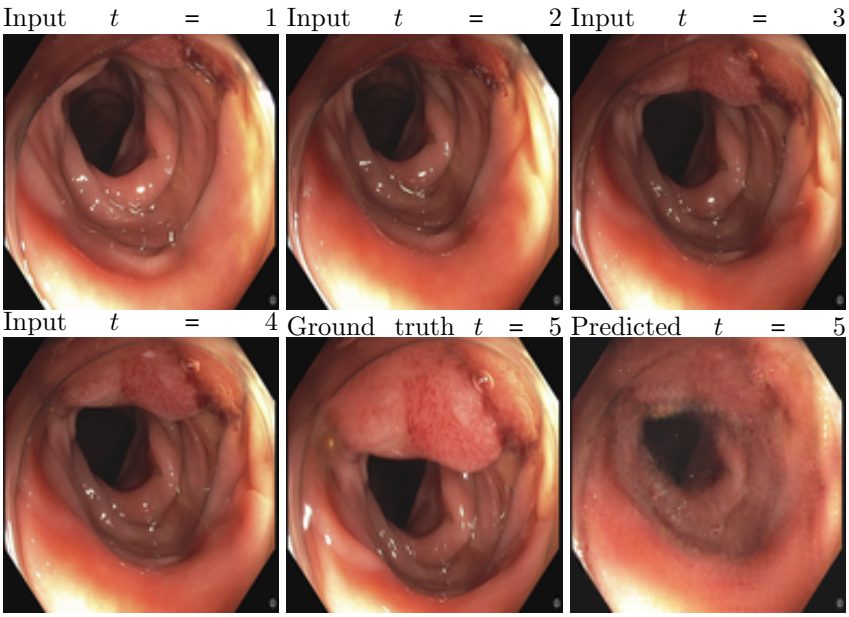


Figure 3.4: The figure shows a sample of the four preprocessed input frames, using the "Skip Frames Based on Quantity" method on the OUS-Study-2019 dataset. The Figure further shows the the ground truth and the predicted output. The output is predicted using Vid2Pix introduced in Section 4.3.

3.3.1 Skip Frames Based on Quantity

Before moving on to more advanced methods, we decided to select frames based on a set interval. The method involves skipping N frames in between each kept frame I in a sequence. The goal is to remove duplicates from all videos in a dataset. The experiment involves skipping the following number of frames in between each kept frame: $N = 5$, $N = 8$, $N = 10$ and $N = 20$. We choose to present the top result from this experiment, which is when $N = 8$.

In addition to skipping a fixed number of frames, we create the videos with a fixed length of 8 frames. Which is the required input for the generative model experiments. During preprocessing we choose to create a new video if the difference in frame number is larger than 10 frames. A difference of 10 frames was found to often cause large jumps in the video.

We test the effect of using the "skip frames based on quantity" method on the dataset. We do this by training our generator model that is introduced in Section 4.3 on the preprocessed dataset. Then we compare the effect by training the same model on the dataset that has not been preprocessed.

3.3.2 Skip Frames Based on Absolute Sum of Difference

The second approach involves calculating the sum of pixel values in two consecutive frames. Then we calculate the difference between the sum of the two frames, lastly we take the absolute value on the difference between the sums. When the calculated value is above a certain threshold, we keep the next frame.

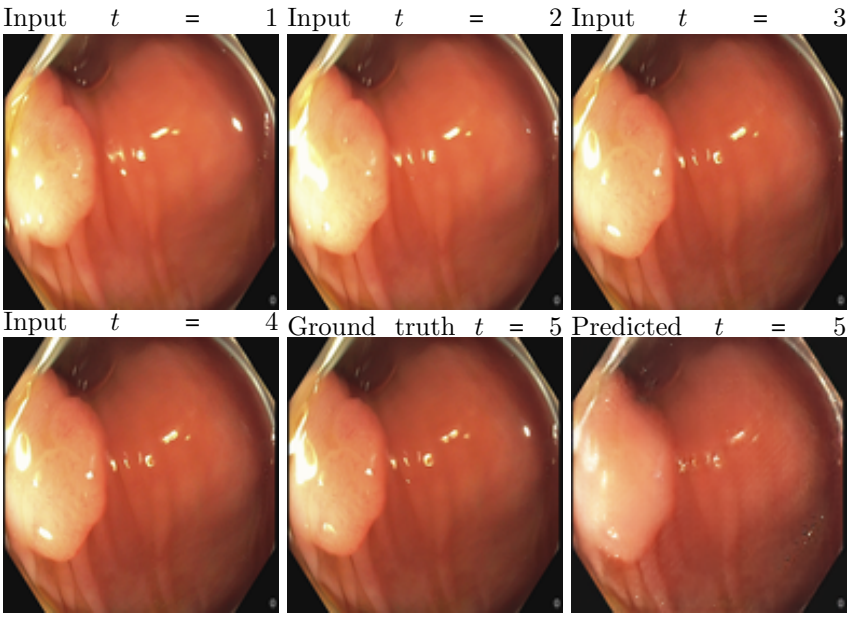


Figure 3.5: The figure shows a sample of the four preprocessed input frames, using the "Skip Frames Based on Absolute Sum of Difference" method on the OUS-Study-2019 dataset. The Figure further shows the ground truth and the predicted output. The output is predicted using Vid2Pix introduced in Section 4.3.

When the difference is lower or equal to the threshold, we skip the next frame. The computation explained above is done for each finding in a video.

In addition to computing the absolute sum of difference between frames, we create videos with a fixed length of 8 frames. Furthermore, we create a new video if the difference in frame number is larger than 10 frames.

To test the effect of preprocessing the dataset using the absolute sum of of difference, we train our generator model that is introduced in Section 4.3 on the preprocessed dataset. Then we compare the effect by training the same model on the dataset that has not been preprocessed.

3.3.3 Skip Frames Based on Dense Optical Flow

The last method we used for skipping frames is based on optical flow. Optical flow is a method for estimating the motion of an object, surface, or edge in a scene. There are two main types of optical flow estimation: sparse optical flow [3] and dense optical flow. Sparse optical flow selects a sparse set of pixels to process, such as corners or edges of an object. The benefit of using sparse optical flow is that it is computationally faster, since it processes fewer pixels in the image. The drawback is that it is less accurate on a pixel-level. We have chosen to use dense optical flow due to its high accuracy. An example of a visualized result of dense optical flow computation is shown in Figure 3.6. Dense optical flow processes all pixels in an image, and outputs flow vectors.

The flow vectors give an estimation of the direction and the intensity of motions in a scene. Flow is computed between two contiguous frames [64].

Figure 3.6: The top image shows the scene that is computed. The bottom image of the figure shows the results of computing dense optical flow between two images [64].

We use optical flow to capture movements from an object or its surroundings. In our experiment, we only consider the magnitude of motion to decide whether to keep or to skip a frame based on a set magnitude threshold. We tested a few threshold values, like the average magnitude between each continuous frame through the whole dataset. We decided to use a threshold of 20% above the average magnitude between each continuous frame in a video of the total dataset. We do this because we are only interested in how big the movements are in the video, the direction of the movement is not critical to solve our problem. As with the other two preprocessing methods, we create each video with a fixed length of 8 frames. Moreover, we create a new video if the difference in frame number is larger than 10 frames to avoid large jumps in the videos.

To measure the effect of preprocessing the dataset using optical flow, we train our generator model that is introduced in Section 4.3 on the preprocessed dataset. Then we compare the effect by training the same model on the dataset that has not been preprocessed.

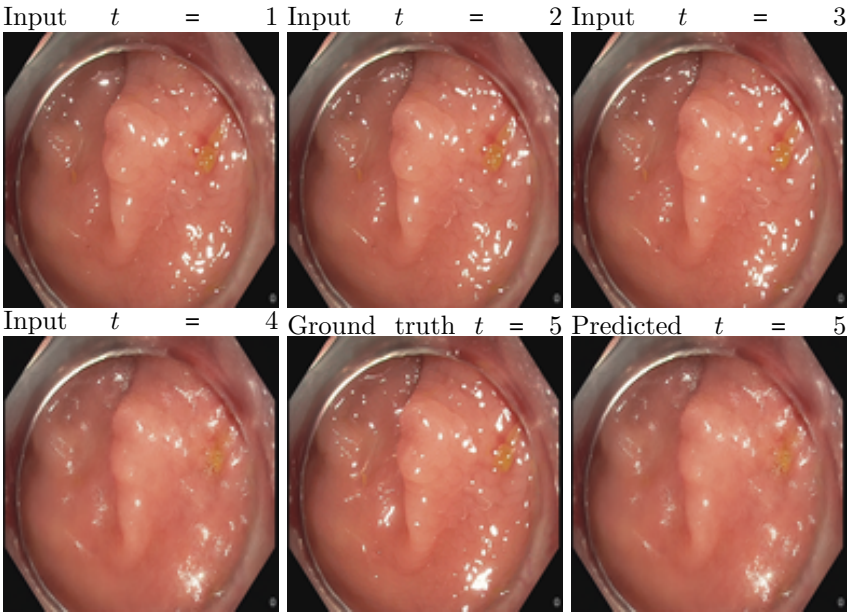


Figure 3.7: The figure shows a sample of the four preprocessed input frames, using the dense optical flow method on the OUS-Study-2019 dataset. The Figure further shows the ground truth and the predicted output. The output is predicted using Vid2Pix introduced in Section 4.3

3.4 Discussion

The "skip frames based on quantity" method in Section 3.3.1, was meant to be a preliminary approach to have a reference point for future experiments. From the results, we see that the camera movement in the colonoscopy videos are rather inconsistent concerning that it is unpredictably standing still or moving fast. The method, on the other hand, is consistent concerning that it is skipping a constant N number of frames throughout the entire video. From Figure 3.4, we observe that the sequence contain large jumps between some images, while some images show great similarity. Combining a consistent method on an inconsistent dataset, did not work well with the purpose of optimizing the original dataset. The method resulted in a random set of desired predicted videos and another random set of undesired predicted videos in terms of getting blurry, or changing color or brightness. Compared to not skipping frames, the method improves prediction.

A major drawback for the method based on the absolute sum of difference in Section 3.3.2, is that it does not differentiate between changes in brightness, noise or motions. This means that it can return a large absolute sum

of difference between two frames, regardless of whether the changes are due to brightness, noise or a motion. We aim to skip duplicate frames that are caused by the absence of motions. Like the method in Section 3.3.1, this method results in a random set of well predicted videos, while another random set of predicted videos are undesired. By undesired videos, we mean that we get large and sudden jumps between two frames, or large changes in color or brightness from one frame to another.Figure 3.5 shows an example of input frames after preprocessing and predicted output using this method. The figure shows one of the greater examples where input is consistent.

The absolute sum of difference did not solve our challenge of capturing motions, exclusively. Instead, we attempt to use a method that can capture motions. We refer to the experiment and result presented in Section 3.3.3 from using the dense optical flow method. By using this method, we were able to reach our goal. Figure3.7 shows an example of our preprocessed input by optical flow, and predicted output. The goal was to capture motions, and skip frames with absence of motions. Based on visual inspection, the generated videos did not show any signs of large and sudden jumps between frames nor any large changes in color or brightness. Overall, the motions in the videos show signs of being consistent.

3.5 Summary

In this chapter, we introduced the OUS-Study-2019 dataset and described how we divided it into separate parts. We further introduced how we examined *GENDAT* dataset and discovered some issues when using raw video sequences. We discovered that multiple sequences in the dataset consisted of large jumps between frames. Moreover, we discovered long sequences with absence of motions. This led us to a new challenge in training a CGAN for generating sequences. When the input sequences had large jumps, the generated output would turn bad in terms of change in color, brightness or getting blurry. To solve this, we first experimented with a quick approach which was to skip frames based on quantity. Another approach was to compute the absolute sum of difference between each frame in a video. Both approaches did not work well to capture inconsistent motions in a video, which resulted in using the dense optical flow method. With this method we managed to optimize the dataset by removing duplicate frames and large jumps between frames. This resulted in a data preprocessing framework that we use to preprocess data that will be used to generate video sequences with a CGAN. In the next chapter, we will introduce the methods that our sequence generator is built upon. Moreover, we describe how we generate future sequences based on past sequences with our method. Finally, we will present and discuss the results of our experiments.

Chapter 4

Future Sequence Generation

In the previous chapters, we have described the challenges concerning access to and the lack of large scale endoscopy data. Furthermore, we have described the need for medical expertise in order to annotate a dataset. We have also shown the importance of having a rich and balanced dataset, where the data covers many different scenarios.

Tackling these challenges is the motivation behind this thesis. Our main goal is to generate artificial polyp videos which look like the real thing. We have developed a generative model, Vid2Pix, which is able to generate videos that can go into a colonoscopy dataset.

In this chapter we aim to describe the details of the U-Net model in Section 4.2.1, and their use of skip connections. We then introduce the Pix2Pix model in Section 4.2.2, which together with U-Net are the building blocks of our generative model. Finally, we explain how our generative model is built.

4.1 Dataset Description

In chapter 3, we introduced the OUS-Study-2019 dataset in Section 3.1. We discovered that the dataset contained many duplicates, which caused our prototype generator network to perform poorly. Since there could be very little movement from frame to frame, our generator simply learned that the best approach was to generate a copy of the input. We applied dense optical flow in Section 3.3.3 on the whole training dataset in order to filter out duplicate frames.

In this section we describe how we use the *GEN-DAT* dataset, that consists of 28932 annotated polyp images. This is reduced to 2784 images with 348 videos after the filtering. We then split the dataset randomly into training, validation and test sets. We distribute the data in a way that gives us as much training data as possible. These subsets were used during the development phase and to train the generator model.

The train, validation and test set were distributed as follows:

- 261 polyp training videos with 8 frames in each video.
- 58 polyp validation videos with 8 frames in each video.

- 29 polyp test videos with 8 frames in each video.

Every video consists of 8 frames, where 4 frames are used as input to a generator and 4 frames are used as ground truth. In one of the experiments, our ground truth is an image instead of a sequence of images. In this matters, we use the first out of the four frames as ground truth. In this particular experiment, we choose to keep the video size in order to compare all experiments.

4.2 Methods

Our main goal is to generate real looking artificial polyp videos. With this in mind, we developed a generative model: Vid2Pix, which is able to generate videos that increase the size of the polyp class of a colonoscopy dataset. In this section we aim to describe the details of the U-Net model and their use of skip connections. Further we introduce the Pix2Pix model that together with U-Net build the foundation of our final generative model. Finally, we introduce our own contribution where we explain how U-Net and Pix2Pix contributes to it.

4.2.1 U-Net

The U-Net model architecture was developed by Ronneberger, Fischer, and Brox in 2015 to process biomedical images [75]. The architecture has a U-shape and is symmetric which can be seen in Figure 4.1. A standard U-Net architecture consists of two parts: an encoder, referred to as the contracting path located on the left side, and a decoder which is referred to as the expansive path on the right side. The contracting path includes a stack of convolutional, ReLU and max pooling layers which are used for feature extraction and also downsample the input data. The expansive path includes a stack of deconvolutional layers which is used to upsample the data and increase the resolution of the output. A full overview of the original U-Net architecture is shown in Figure 4.1.

Skip connections

A skip connection is a connection between two layers, where intermediate layers are skipped. In U-Net, the contracting path and expansive path are connected in two ways. First, in the transition between a series of downsampling layers and upsampling layers. Second, as skip connections directly in between each upsampling and downsampling layer. This means that the architecture share features with skip connections between each layer i and $n - i$ where n is the total number of layers [44]. For each skip connection, all channels at layer i are concatenated with layer $n - i$ [44, p. 3]. A simple example of a general encoder-decoder versus the U-Net architecture with skip connections is shown in Figure 4.2. The main objective of skip connections is to learn to assemble a more precise output based on combining high resolution features from the contracting path with the expansive path [75].

4.2.2 Pix2Pix

The Pix2pix model architecture is inspired by U-net. Pix2Pix is a generative model architecture developed to perform image-to-image translation tasks.

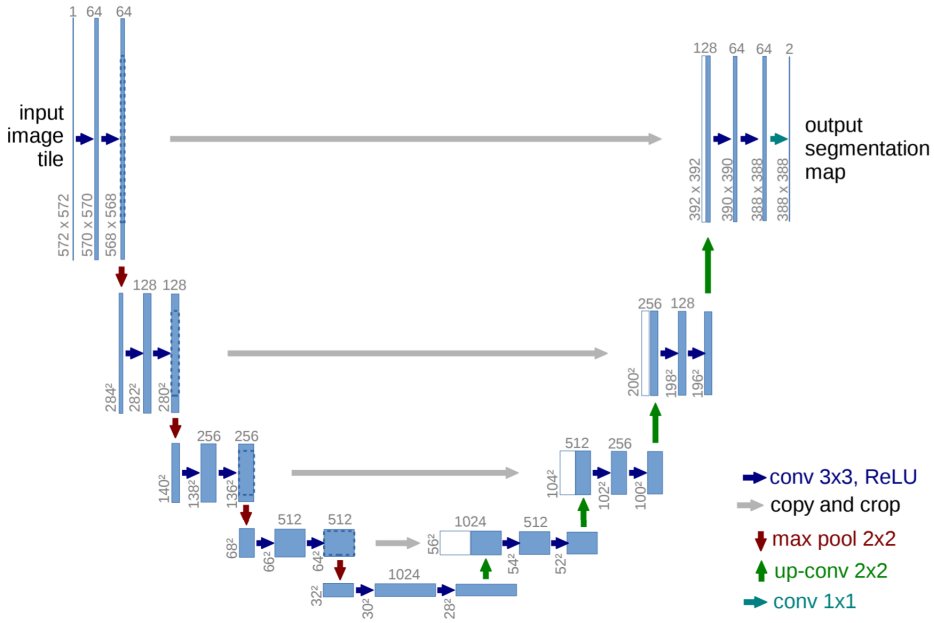


Figure 4.1: The U-Net architecture developed by Ronneberger, Fischer, and Brox [75].

In the paper "Image-to-Image Translation with Conditional Adversarial Networks", Isola et al. define image-to-image translation as "the task of translating one possible representation of a scene into another, given sufficient training data" [44]. Isola et al. introduces a variety of tasks for image-to-image translation. For instance, Pix2Pix can map edges to a photo or a semantic label to a photo. In our case, we want to predict a future image in a sequence based on past a sequence.

Pix2pix uses a conditional GAN framework. The Pix2Pix conditional GAN learns to map an image y and a random noise vector z, to x. An example of how edges to photo translation is done using a Pix2Pix is shown in Figure 4.3. In Pix2Pix, the discriminator and the generator both contain modules of layers in the following order: convolutional, batch normalization and ReLU or LeakyReLU.

Generator Similar to U-Net, Pix2pix utilizes skip connections. The skip connections are only present in the generator. The skip connections are connected between each corresponding upsamling and downsampling layer. The upsampling and downsampling layers are also connected in the transition between their respective series of layers as shown in Figure 4.2 on the right.

Discriminator The discriminator computes a feature map from two concatenated images. The computation is done twice. First from a concatenation of the input image and the generated image, and second from a concatenation of the input image and the ground truth. The discriminator attempts to guess which of the two feature mappings are generated or real. Isola et al. refers to

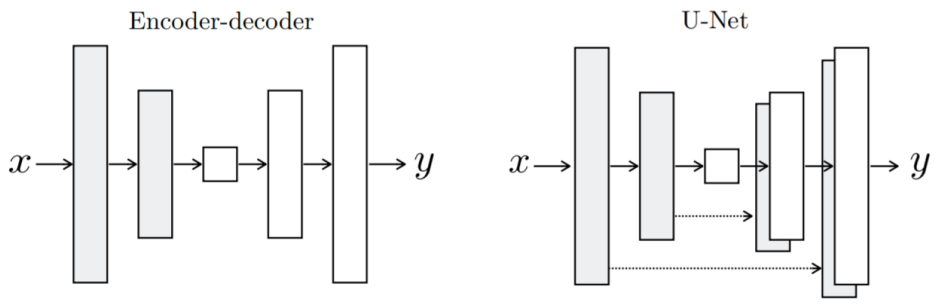


Figure 4.2: Two generator architectures. To the left: a general encoder-decoder architecture. To the right: U-Net architecture with skip connections [44]

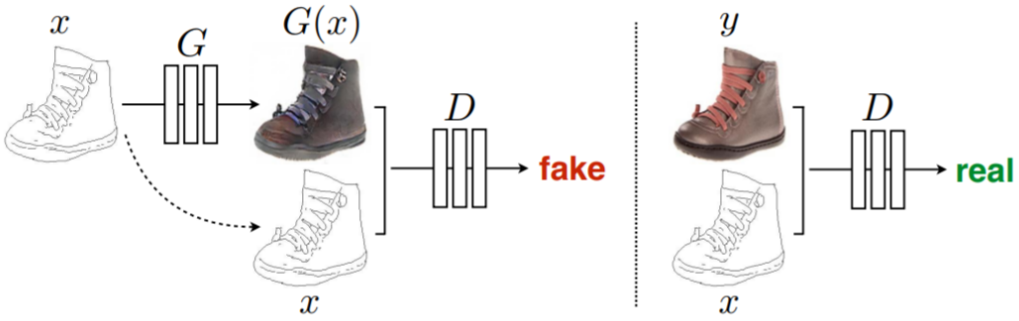


Figure 4.3: The figure shows an example of the training process of the Pix2Pix model. On the left: the generator tries to map the conditional input x, to a real image of a shoe. On the right: The ground truth and the conditional input x is is input to the discriminator. G tries to map edges to a photo, and G learns to fool the discriminator, D [44, p. 2].

the discriminator architecture as a PatchGAN, which is a convolution network that maps the two input images to a $N \times N$ output matrix rather than a one dimensional output vector. Each element in the $N \times N$ matrix is mapped to local patches in the input images. By putting attention to local image patches, PatchGAN is able to capture local texture details in the input images.

4.3 Future Sequence Generation with Vid2Pix

Pix2Pix was developed to translate an *image* in one domain to an image in another domain. We are trying to learn past *image sequence* (videos) in one domain in order to generate new sequences in the same domain. While the Pix2Pix architecture does not directly fit our problem, we wanted to extend Pix2Pix so that it could work with sequences.

Vid2Pix is a generative model that predicts a future frame conditioned on the past frames in a sequence. In contrast to Pix2Pix, the purpose of developing

Vid2Pix is to produce artificial videos, rather than images. Similar to Pix2Pix, Vid2Pix is a conditional GAN and consists of a generator with a U-Net architecture and a discriminator with a PatchGAN architecture. The table below shows the main modifications to the existing Pix2Pix.

- Add an additional dimension for input sequence support.
- Replace 2D with 3D convolutions for all downsampling layers.
- Replace 2D with 3D deconvolutions for all upsampling layers.
- Many-to-one input-output

4.3.1 Generator

The generator architecture is structured with a series of convolutional downsampling layers, a series of deconvolutional upsampling layers and skip connections. Instead of using 2D convolutions as in Pix2Pix for downsampling, we use 3D convolutions. We use 3D deconvolutional layers instead of 2D deconvolutional layers for upsampling. An overview of the full generator model is shown in Figure 4.4. We add an additional dimension so that we end up with three dimensions, two spatial and one temporal dimension. We do this to extract features from the temporal dimension. The temporal dimension provides information on the temporal shape and movement of an object or its surroundings as explained in Section 2.4.2. A full overview of the model architecture is shown in FigureB.1

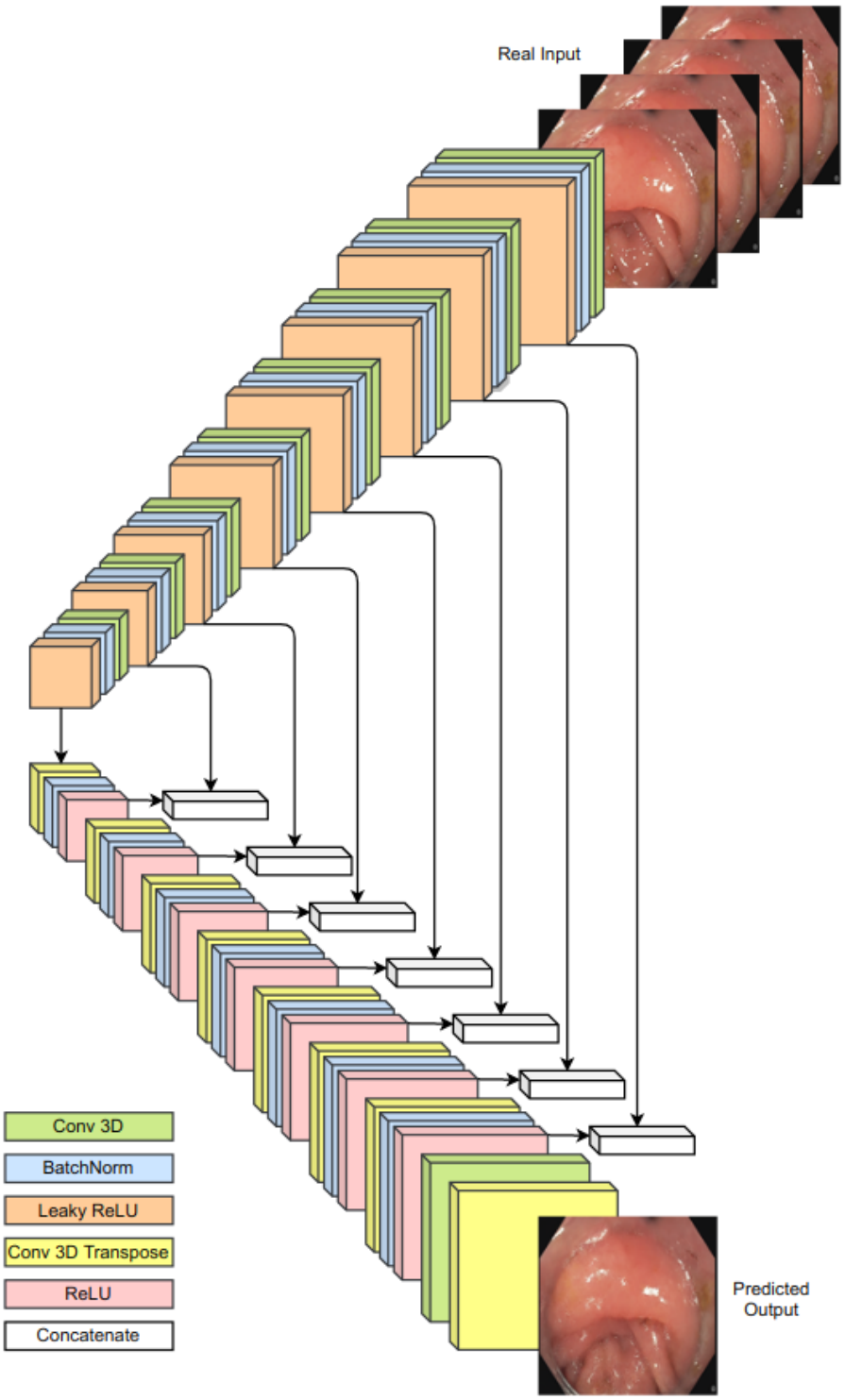


Figure 4.4: The Vid2Pix generator model architecture.

4.3.2 Discriminator

The discriminator outputs a downsampled feature map from either a concatenation of the input sequence and a generated image shown in Figure 4.5, or from a concatenation of the input sequence and the ground truth. The similar model shown in Figure 4.5 is used for the ground truth, besides switching the generated input with ground truth. The discriminator consists of a PatchGAN, which is explained in Paragraph 4.2.2. We use a patchGAN so that we can capture local texture details in the images. Further, we use 3D convolutions instead of 2D convolutions in a series of downsampling layers to ensure temporal feature extraction. A full overview of the model is shown in Figure B.2.

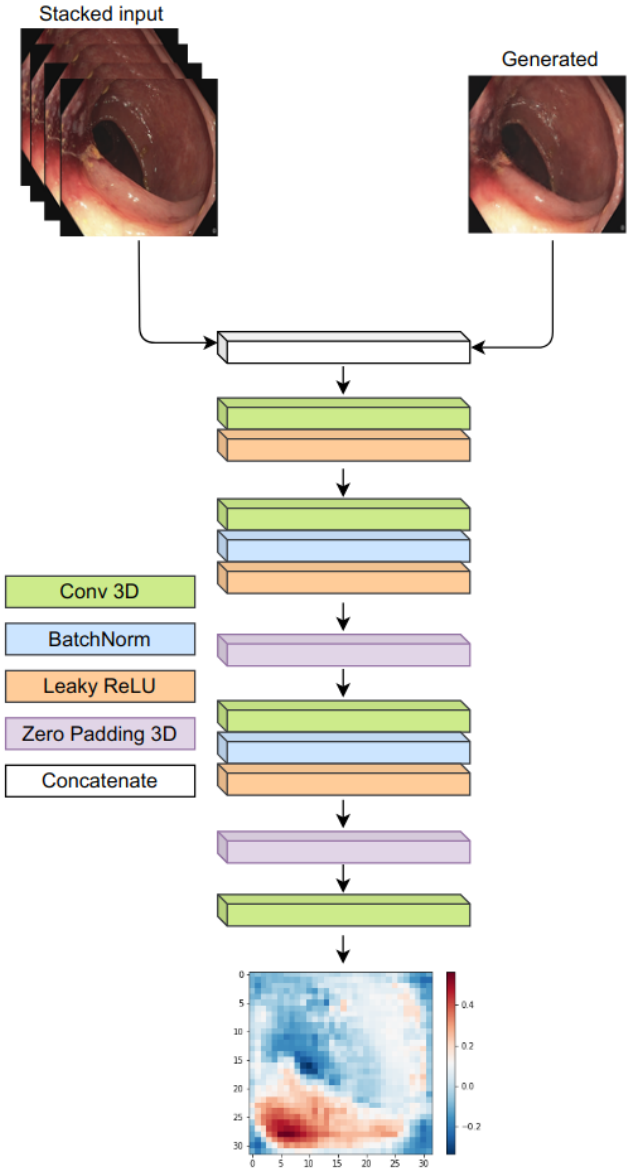


Figure 4.5: The figure shows Vid2Pix discriminator model with a real input sequence and the generated output. The output is compared to the output from Figure 4.5

4.4 Model Experiments and Results

In this section, we present the experiments performed to arrive at our final generative model. All decisions made during this phase was based on the goal of generating realistic looking motions and high image quality in each video.

In our first experiment, we use the original Pix2Pix model with four stacked images as input. Further, we modify the Pix2Pix model to learn spatiotemporal

features by replacing 2D convolutional and deconvolutional layers with 3D convolutional and deconvolutional layers. In further experiments, we add or remove parameters step by step until we finally end up with our model architecture: Vid2Pix.

Each generated result is presented in a figure together with the last frame of the four conditional input frames, and the ground truth. We choose three identical input sequences to generate one image for all experiments. Identical input sequences are chosen in order to compare the results of the experiments. The figures are used to compare the similarities and the differences between the images.

In this section, we will describe each experiment step by step, which will lead us to our final contribution. Each of the seven experiments represents a single change in the model architecture. It is not guaranteed that each step alone improves the performance of the model, but if we implement all the changes at once, we achieve the best performance. Thus, keeping changes done to the model architecture is not necessarily dependent on the previous step, but by the final result.

4.4.1 Experiment Setup

We have used the high-level programming language Python [27] to implement our model. Python is known for being easy to use and for having a large number of machine learning libraries and tools. Python has become popular for doing machine learning tasks [88]. The code written for this thesis is exclusively written in Python, and most of the experiments have been developed in the interactive development environment Jupyter.

Tensorflow was originally developed by researchers on the Google Brain Team. It is a popular end-to-end platform for building, deploying, and experimenting with machine learning models [109]. In 2019, Tensorflow released a 2.0 version of the library, which integrated the Keras API directly into the library itself. Keras is accessible through the *tensorflow.keras* module, where one has access to all features available through the standalone API. For this thesis, we use Tensorflow 2.0 and *tensorflow.keras* to experiment and build our machine learning models.

To train and test our experiments, we got access to the "Experimental Infrastructure for Exploration of Exascale Computing" server (Ex3) provided by Simula and funded by the Research Council of Norway. The server holds 16 NVIDIA V100 GPUs with a total of 512GB GPU memory.

4.4.2 Original Pix2Pix with Stacked Input

Design

In this experiment, we use the original Pix2Pix model architecture described in Section 4.2.2 without any modifications. Originally, Pix2Pix translates one input image to another output image. In this experiment we input a stacked sequence of images to predict and generate the next image in the sequence.

The original architecture does not allow to add another dimension to the input of the model without being modified. In Pix2Pix, 3 channels are used as the input. This means that the input is a color image where the 3 channels

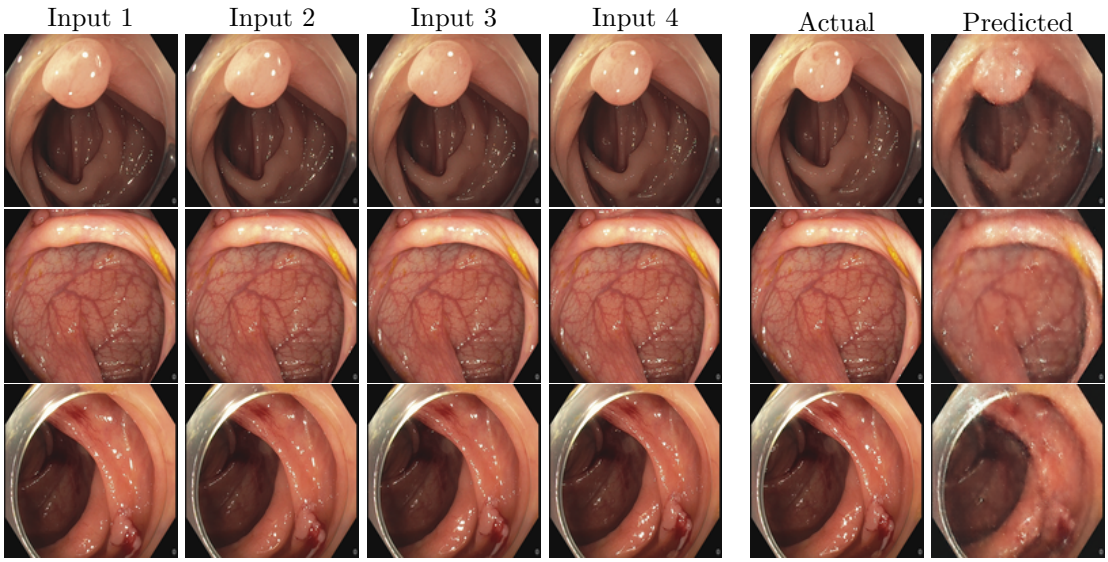


Figure 4.6: The figure shows the four frames that represent the stacked input to the model, the ground truth and the predicted output for the Original Pix2Pix experiment 2.0. To be able to fit the images in one line and based on the low image resolution, we choose a rather small figure size.

represent the colors red, green, and blue. The original architecture accepts one frame as input, but you can adjust the number of input channels. Therefore, we choose to stack our images in the channel dimension.

Instead of using a single frame as input to the model, we input four stacked frames. When we stack four images we get $4x3$ channels, which means that we have stacked four color images and end up with an input size of $128 \times 128 \times 12$. When our stacked input consists of four images from time $t = 1$ to $t = 4$, we try to predict the image at $t = 5$. In order to predict an image at $t = 5$, our ground truth has to be the next image in the sequence at $t = 5$.

Results

The upper row of Figure 4.6, shows four input frames from $t = 1$ to $t = 4$, followed by ground truth at $t = 5$ and the predicted output at $t = 5$ for input video 1. From the figure, we observe that the element sizes and the large structures in the predicted output resemble the ground truth. The placement of elements are also similar to the ground truth. Moreover, we see a rough pattern in some bounded areas of the predicted image.

The middle row of Figure 4.6, shows four input frames from $t = 1$ to $t = 4$, followed by ground truth at $t = 5$ and the predicted output at $t = 5$ for input video 2. In the upper row, the larger structures appear similar to the ground truth. On a detailed level, we observe that the thinnest blood vessels are blurred and hardly visible.

The lower row in Figure 4.6, shows four input frames from $t = 1$ to $t = 4$, followed by ground truth at $t = 5$ and the predicted output at $t = 5$ for input video 3. The larger features in the predicted image look similar to the features

in the ground truth image. Important areas that consist of an outgrowth with hemorrhages is less visible due to blurriness. Overall, the image appear close to ground truth on large structures, while the smaller elements of the image appear blurred.

An overall observation for all input videos and their predicted output in Figure 4.6 is that larger structures in the generated images appear similar to their respective ground truth, while the smaller features in the images appear blurred.Training loss is shown in Figure A.1.

4.4.3 Replace with 3D Layers

Design

3D convolutions have been proven sucessful for temporal feature extraction which we introduced in Section 2.4.2. According to Ji et al., 3D CNNs outperform 2D CNNs on several spatiotemporal feature extraction tasks [46]. In this experiment we aim to improve the performance of the model by replacing the 2D layers with 3D layers in the Pix2Pix.

We add one extra dimension to the input and output of the model, in order to be able to use 3D layers. We choose to use an input size of $4\times128\times128\times3$ and an output size of $4\times128\times128\times3$ where 4 represents the number of frames, and 128×128 represents the image resolution. 3 represents the number of channels.

The following changes were done to the layers in both the discriminator model and the generator model:

- Replace 2D convolutional layers with 3D convolutional layers.
- Replace 2D deconvolutional layers with 3D deconvolutional layers.
- Replace 2D zero padding layers with 3D zero padding layers in the discriminator.

This experiment is using the original Pix2Pix architecture except for replacing 2D convolutions and deconvolutional layers with 3D layers.

Results

The upper row of Figure 4.7, shows four input frames from t = 1 to t = 4, followed by ground truth at t = 5 and the predicted output at t = 5 for input video 1. We see that the large structures like the polyp and the intestinal folds are located in the same areas as in the ground truth. The smaller structures are difficult to see clearly. There is a grainy texture on a detailed level.

The middle row of Figure 4.7, shows four input frames from t = 1 to t = 4, followed by ground truth at t = 5 and the predicted output at t = 5 for input video 2. We observe that the larger and folded structure starting from the middle of the image partially blends in with the rest of the tissue. The thin blood vessels are missing, and the thicker blood vessels are difficult to see. The texture is grainy.

The lower row of Figure 4.7, shows four input frames from t = 1 to t = 4, followed by ground truth at t = 5 and the predicted output at t = 5 for input video 3. Like in the predicted images from video 1 and 2, the larger features are similar to the ground truth. For the local areas, we observe that the outgrowth

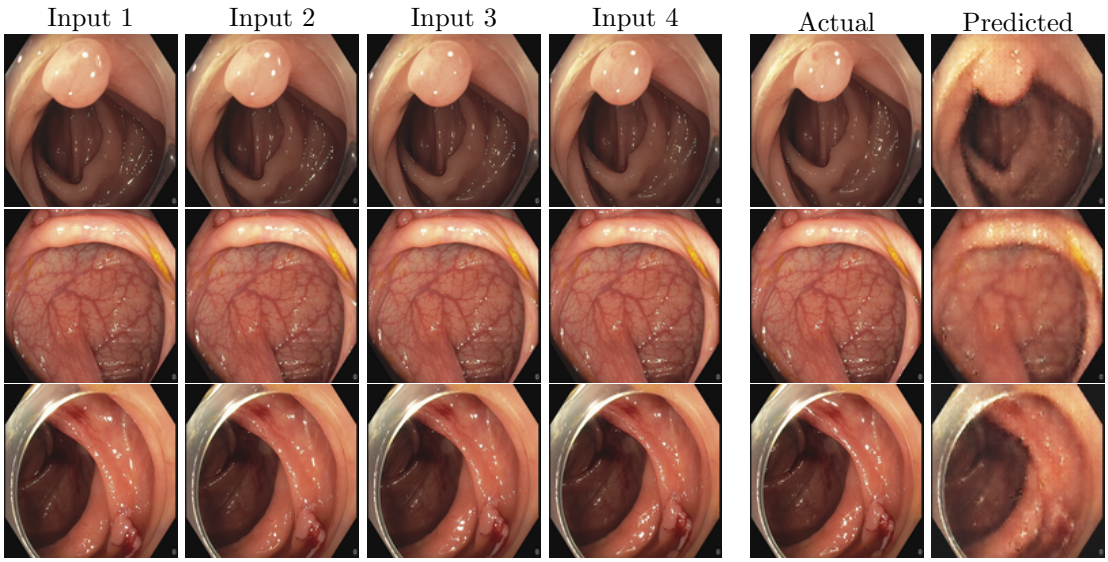


Figure 4.7: The figure shows the four frames that represent the stacked input to the model, the ground truth and the predicted output for the Original Pix2Pix experiment 2.1.

with hemorrhages differs from the ground truth in blending in with other tissue. The smaller features appear rather grainy here as well.

Overall, some of the generated images consist of larger structures that partially differ from their respective ground truth. The smaller structures consists of greater errors like missing blood vessels. Additionally, all of the generated images appear grainy. Training loss is shown in Figure A.2.

4.4.4 Change Filter Size

Design

This experiment is built upon the "Replace With 3D layers" experiment from Section 4.4.3. We keep the earlier modification, and further develop the model by changing the filter size. The filter size is changed from 4 to filter size 3 and 5 in both the generator and discriminator layers. We apply two filter sizes instead of one in order to test if the filters can better learn higher level features from filter size 3 and lower level features from filter size 5. Tran et al. suggests the filter size of 3 to be the best working filter size for 3D convolutions[95].

In the generator, we use a filter size of 3 in the first four downsampling layers and we use a filter size of 5 for the remaining four downsampling layers. Each upsampling layer uses the filter size that corresponds to the downsampling layer it is skip connected with. A full overview of the changes that have been made together with the corresponding skip connections is shown in Table 4.1. In the discriminator, we use filter size 3 for the first two downsampling layers. Next, we use filter size 5 on a new downsampling layers, and then filter size 3 on the last layer. A full overview of the filter size changes we have made to the discriminator is shown in Table 4.2.

Layer	Filter Size Pix2Pix	Filter Size Experiment 2.1	Skip Connection
Conv3D 1	4	3	DeConv3D 8
Conv3D 2	4	3	DeConv3D 7
Conv3D 3	4	3	DeConv3D 6
Conv3D 4	4	3	DeConv3D 5
Conv3D 5	4	5	DeConv3D 4
Conv3D 6	4	5	DeConv3D 3
Conv3D 7	4	5	DeConv3D 2
Conv3D 8	4	5	no skip
DeConv3D 1	4	5	no skip
DeConv3D 2	4	5	Conv3D 7
DeConv3D 3	4	5	Conv3D 6
DeConv3D 4	4	5	Conv3D 5
DeConv3D 5	4	3	Conv3D 4
DeConv3D 6	4	3	Conv3D 3
DeConv3D 7	4	3	Conv3D 2
DeConv3D 8	4	3	Conv3D 2

Table 4.1: Overview of changes in filter size for all convolutional layers and deconvolutional layers in the generator

Layer	Filter Size Pix2Pix	Filter Size Experiment 2.2
Conv3D 1	4	3
Conv3D 2	4	3
Conv3D 3	4	5
Conv3D 4	4	3
Conv3D 5	4	3

Table 4.2: Overview of changes in filter size for all convolutional layers in the discriminator

Results

The upper row of Figure 4.8, shows four input frames from t = 1 to t = 4, followed by ground truth at t = 5 and the predicted output at t = 5 for input video 1. W see that larger features like the polyp and the intestinal folds are located in the same areas as they are located in the ground truth. The smaller features are difficult to see clearly. Both the large and the small structures in the image appear pixelated with horizontal lines across the whole image. We see that the colors turn brighter in the predicted image compared to the ground truth.

The middle row of Figure 4.8, shows four input frames from t = 1 to t = 4, followed by ground truth at t = 5 and the predicted output at t = 5 for input video 2. We observe that the small polyp located in the upper left area of the image is hardly visible. In addition, the blood vessel are difficult to see clearly. Like in predicted image from video 1, the whole image appear pixelated with

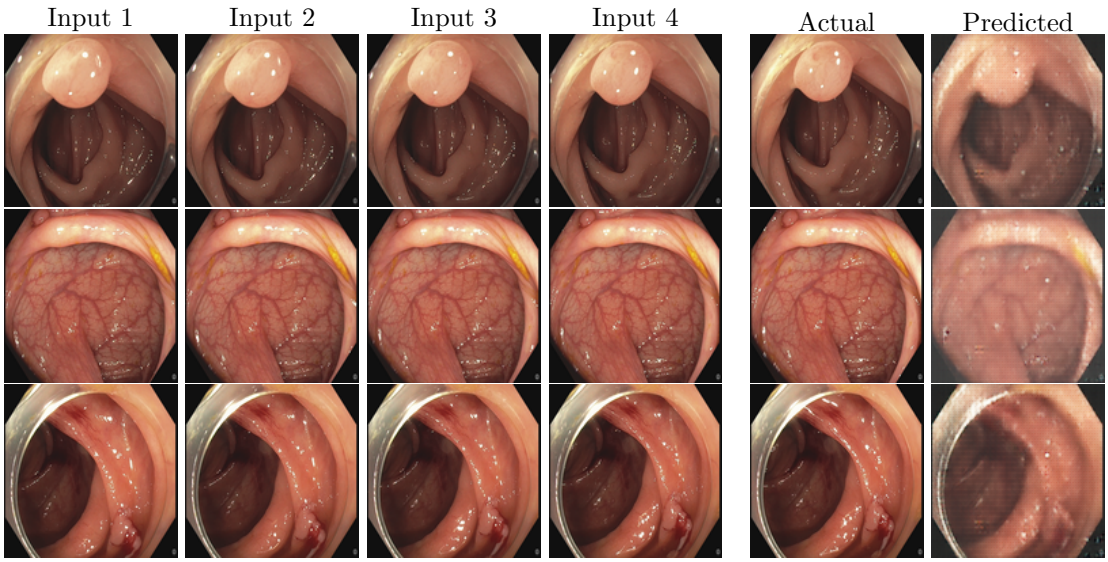


Figure 4.8: The figure shows the four frames that represent the stacked input to the model, the ground truth and the predicted output for the Original Pix2Pix experiment 2.2.

horizontal lines across it. Colors additionally turn brighter in the generated image compared to the ground truth.

The lower row of Figure 4.8, shows four input frames from $t = 1$ to $t = 4$, followed by ground truth at $t = 5$ and the predicted output at $t = 5$ for input video 3. From the figure, we observe that hemorrhages on the outgrowth is hardly visible. As in the previous predicted images, the large and the small features appear pixelated and colors are brighter in the generated image compared to the ground truth. The horizontal lines are present across the whole image.

Overall, we observe that the generated images in this experiment tend to have brighter colors compared to their respective ground truth. Moreover, we observe that the generated images tend to be pixelated.Training loss is shown in Figure A.3.

4.4.5 Offset Downsampling

Design

This experiment is built upon the "Change filter size" experiment in Section 4.4.4. To generate a more precise output, we aim to learn higher level features from a higher a input image resolution.

In the previous model, we downsample our input image resolution in the first convolutional layer, *before* the skip connection which is explained in Section 4.2.1. By doing this, the last layer of the generator will get concatenated with high level features from a 64×64 image resolution.

Instead, we offset the downsampling of the image resolution. This means that we downsample from 128×128 to 64×64 *after* the first skip connection. By doing this the last layer of the generator will get concatenated with high

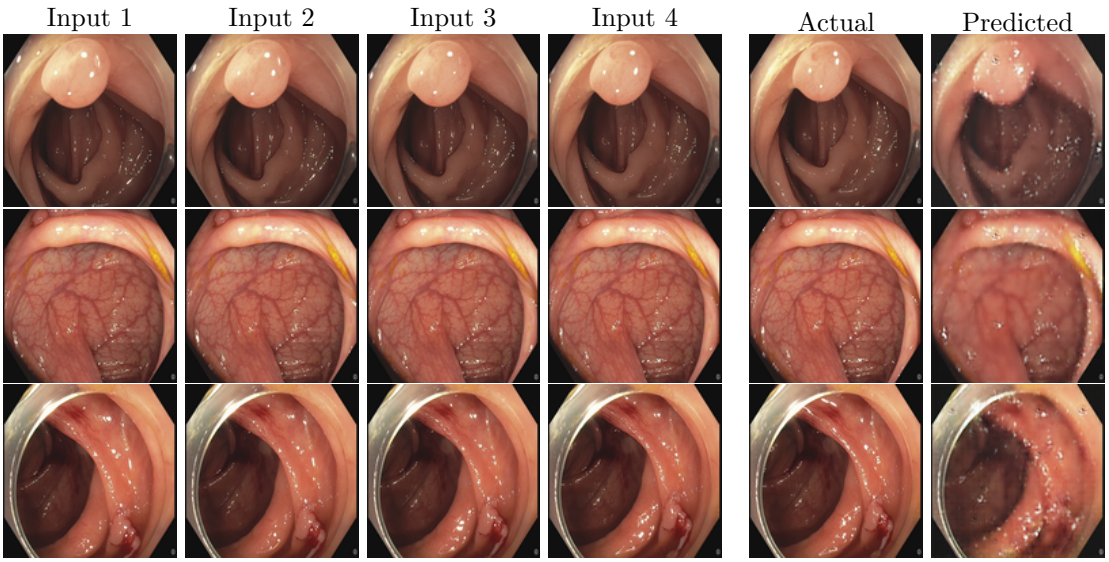


Figure 4.9: The figure shows the four frames that represent the stacked input to the model, the ground truth and the predicted output for the Original Pix2Pix experiment 2.3.

level features from a 128×128 image resolution. Our goal is to generate a more precise output through learning of higher level features.

Results

In the upper row of Figure 4.9, we see four input frames from $t = 1$ to $t = 4$, followed by ground truth at $t = 5$ and the predicted output at $t = 5$ for input video 1. From the figure, we observe that the the overall structure of the image is similar to ground truth. Moreover, we observe that the colors in the generated image and the ground truth appear uniform. We find some local areas of the predicted image to be rough, especially in areas where we find reflections in the ground truth image.

The middle row of Figure 4.9, shows four input frames from $t = 1$ to $t = 4$, followed by ground truth at $t = 5$ and the predicted output at $t = 5$ for input video 2. We observe that the lighter and darker areas in the ground truth and the predicted image to be similar. The folded structure located in the middle of the image and the small polyp on in the upper left corner are both easily visible. The thin blood vessels are not visible due to blur, while the thick blood vessels are somewhat visible.

The lower row of Figure 4.9, shows four input frames from $t = 1$ to $t = 4$, followed by ground truth at $t = 5$ and the predicted output at $t = 5$ for input video 3. From the figure, we see that large features in the ground truth image and the generated image to be similar. Moreover, colors seems to be uniform between the predicted and ground truth. The outgrowth with hemorrhages is somewhat visible, but not in detail. While the texture of the image is is somewhat rough.

Overall, we observe that the predicted images tend to have a rough pattern, especially in the same areas as we find light reflections in the ground truth

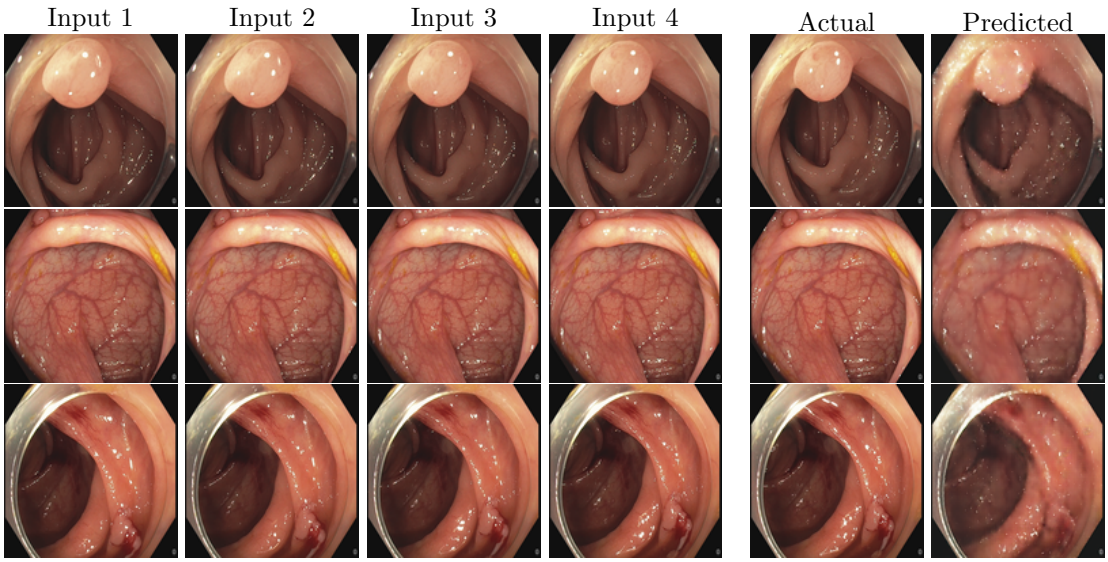


Figure 4.10: The figure shows the four frames that represent the stacked input to the model, the ground truth and the predicted output for experiment 2.4.

images. We observe that colors are uniform and that some areas in the generated images look more similar to ground truth. Training loss is shown in Figure A.4.

4.4.6 Reduce Discriminator Complexity

Design

We base this experiment on the "Offset downsampling" experiment from Section 4.4.5. In order to improve the performance of the generator from the previous experiment, we remove one layer from the discriminator. We attempt to make the discriminator less powerful by reducing its complexity, hence we make the generator more powerful by gaining greater complexity than the discriminator.

Results

The upper row of Figure 4.10, shows four input frames from $t = 1$ to $t = 4$, followed by ground truth at $t = 5$ and the predicted output at $t = 5$ for input video 1. The figure shows clear similarities between the predicted output and the ground truth on an overall level. The placement of structures are close to the ground truth. Moreover, we observe some small grainy areas on a detailed level.

The middle row of Figure 4.10, shows four input frames from $t = 1$ to $t = 4$, followed by ground truth at $t = 5$ and the predicted output at $t = 5$ for input video 2. We observe that the predicted output and the ground truth are similar on an overall level. Moreover, be observe small grainy areas on the generated image in the same areas that we observe light reflections in the ground truth. Thick blood vessels are easily visible, while thin blood vessels are not visible.

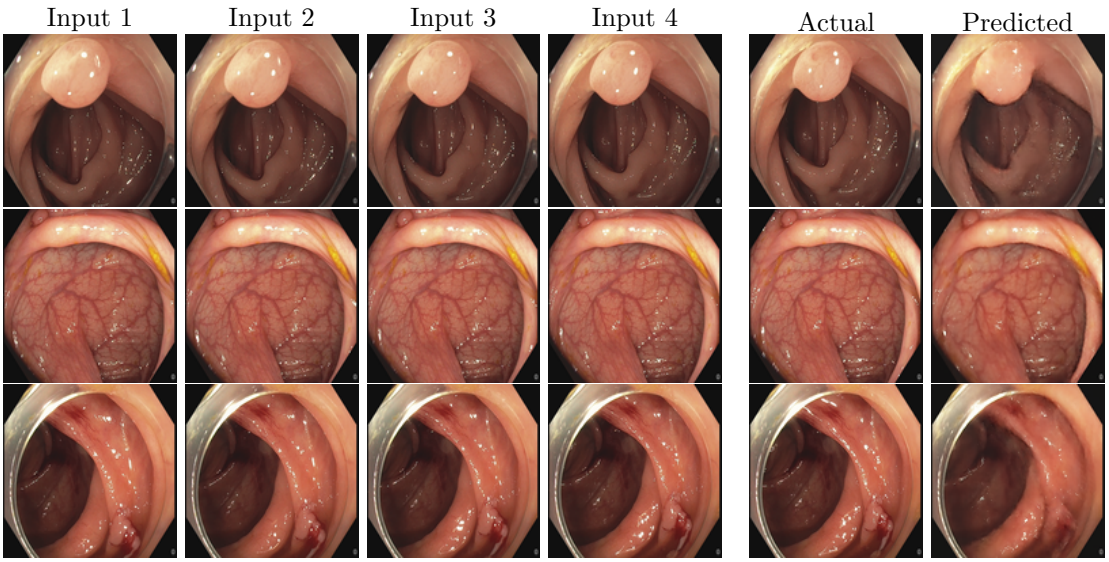


Figure 4.11: The figure shows the four frames that represent the stacked input to the model, the ground truth and the predicted output for experiment 2.5.

The folded structure in the middle of the image and the small polyp in upper left corner are both easily visible.

The lower row of Figure 4.10, shows four input frames from $t = 1$ to $t = 4$, followed by ground truth at $t = 5$ and the predicted output at $t = 5$ for input video 3. We observe similarities between ground truth and the generated image on an overall level. The outgrowth with hemorrhages is visible, but with some blur. We observe some rough and grainy details in the same areas where we find light reflections in the ground truth. Moreover, we can see more clear edges further into the colon.

Overall, we observe some areas in the images as very clear and similar to their respective ground truth. We observe small grainy details in in bounded and detailed areas of the three images. Moreover, the larger features like the folded structures, thick blood vessels and polyps are easily visible. Training loss is shown in Figure A.5.

4.4.7 Keep more features

Design.

Based on the previous experiment "Reduce descriminator complexity" in Section 4.4.6, we attempt to further develop the model in experiment by stopping the downsampling when the image resolution is downsampled twice. This means that we stop downsampling the image when the image resolution reaches 32×32. When we downsample the image resolution, we set strides to 2 for the dimension that is intended to be downsampled through a 3D convolutional layer. When we stop the downsampling, we set strides to 1 for the dimensions that are not intended to be downsampled through a 3D convolutional layer.

Results. The upper row of Figure 4.11, shows four input frames from $t = 1$ to $t = 4$, followed by ground truth at $t = 5$ and the predicted output at $t = 5$ for input video 1. We observe great similarities between ground truth and the generated image on an overall level. The polyp hardly shows any artifacts and there are few bounded areas consisting of small artifacts. The large structures are highly similar to the ground truth.

The middle row of Figure 4.11, shows four input frames from $t = 1$ to $t = 4$, followed by ground truth at $t = 5$ and the predicted output at $t = 5$ for input video 2.The ground truth and the generated image are clearly similar. Both the thick and the thing blood vessels are clearly visible, but a few of the thinnest vessels can be unclear. The large features are similar to the ground truth.

The lower row of Figure 4.11, shows four input frames from $t = 1$ to $t = 4$, followed by ground truth at $t = 5$ and the predicted output at $t = 5$ for input video 3. The ground truth and the generated image shows an overall similarity. The outgrowth with hemorrhages is visible, but with somewhat blurriness. The large structures are clearly similar to the ground truth and with few artifacts.

Overall, we observe that the generated images in this experiment show great similarity to their respective ground truth. We observe few artifacts in local areas of the images. Moreover, we observe a few local areas with blur. Training loss is shown in Figure 4.15.

4.4.8 Add Noise

Design

In order to prevent the generator model from producing deterministic outputs, we add Gaussian noise together with the conditional input. This is a method proposed in the paper "Generative Image Modeling Using Style and Structure Adversarial Networks" [103]. We further base the experiment on the previous experiment: "Keep more features" from Section 4.4.7. Here we aim to improve the performance of the previous model by adding Gaussian noise to the input. We do this in order to make the model more robust to small changes in the input and to prevent the model from predicting a copy of the input.

Results

The upper row of Figure 4.12, shows four input frames from $t = 1$ to $t = 4$, followed by ground truth at $t = 5$ and the predicted output at $t = 5$ for input video 1. Overall, we observe large similarities between ground truth and the generated image. The colors are brighter in the predicted image, in addition it has lower color contrast. The polyp hardly shows any artifacts. We observe a few small artifacts on the left intestinal fold. Moreover, the large structures are greatly similar to the ground truth.

The middle row of Figure 4.12, shows four input frames from $t = 1$ to $t = 4$, followed by ground truth at $t = 5$ and the predicted output at $t = 5$ for input video 2. We observe great similarity between the generated image and the ground truth. We clearly observe the polyp in the upper left corner and tissue shows similar light reflections as the ground truth. The predicted image is overall brighter than the ground truth, and with lower color contrast. Large and small artifacts are not observed. Large and small blood vessels are visible, besides the very thinnest.

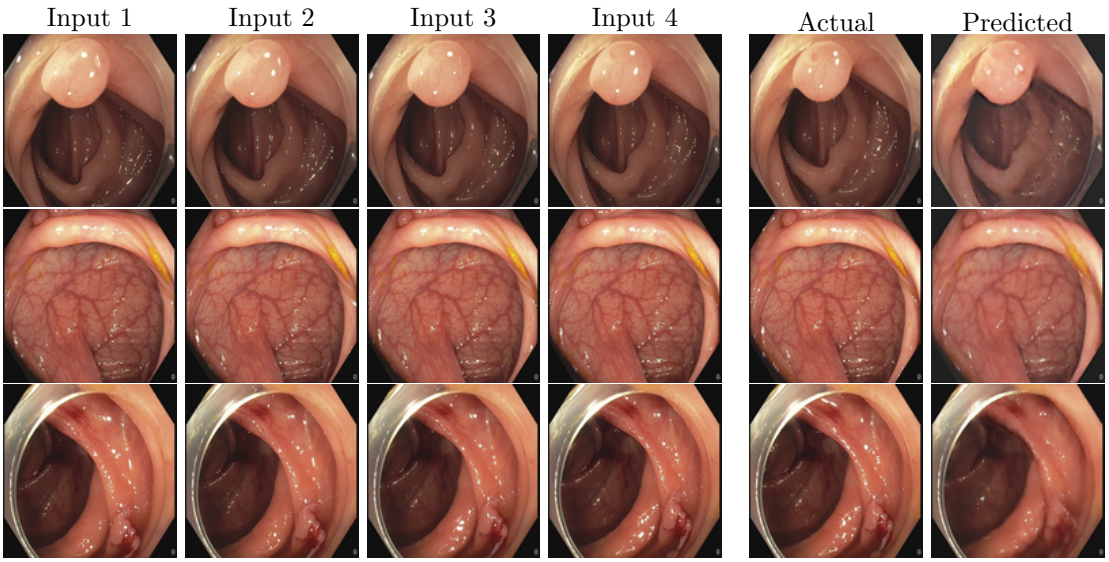


Figure 4.12: The figure shows the four frames that represent the stacked input to the model, the ground truth and the predicted output for experiment 2.6.

The lower row of Figure 4.12, shows four input frames from $t = 1$ to $t = 4$, followed by ground truth at $t = 5$ and the predicted output at $t = 5$ for input video 3. The generated image and the ground truth shows a great similarity. The predicted image tend to be brighter compared to ground truth and with lower color contrast. The outgrowth with hemorrhages is visible but with some artifacts. We observe a few local areas with small artifacts, while large structures are greatly similar to the ground truth.

Overall, the three images that are generated in this experiment shows great similarity to their respective ground truth. Some local areas consists of small artifacts in two of the images, while in one of the images, we observes no large or small artifacts. Training loss is shown in Figure A.7.

4.4.9 2D Output

Design

This experiment is built upon the "Add noise" experiment from Section 4.4.8. Here we aim to improve the performance of the generator by outputting one image instead of four. We do this by adding an extra convolutional layer as the second last layer of our generator. In the previous model, we got $4\times128\times128\times64$ as input to the last *deconvolutional* layer of the generator model. Now, we get the same size as input to a *convolutional layer*, that will output $1\times128\times128\times64$ and pass it further as an input to the last deconvolutional layer.

Results

The upper row in Figure 4.13, shows four input frames from $t = 1$ to $t = 4$, followed by ground truth at $t = 5$ and the predicted output at $t = 5$ for input video 1. We observe great similarities between ground truth and the generated

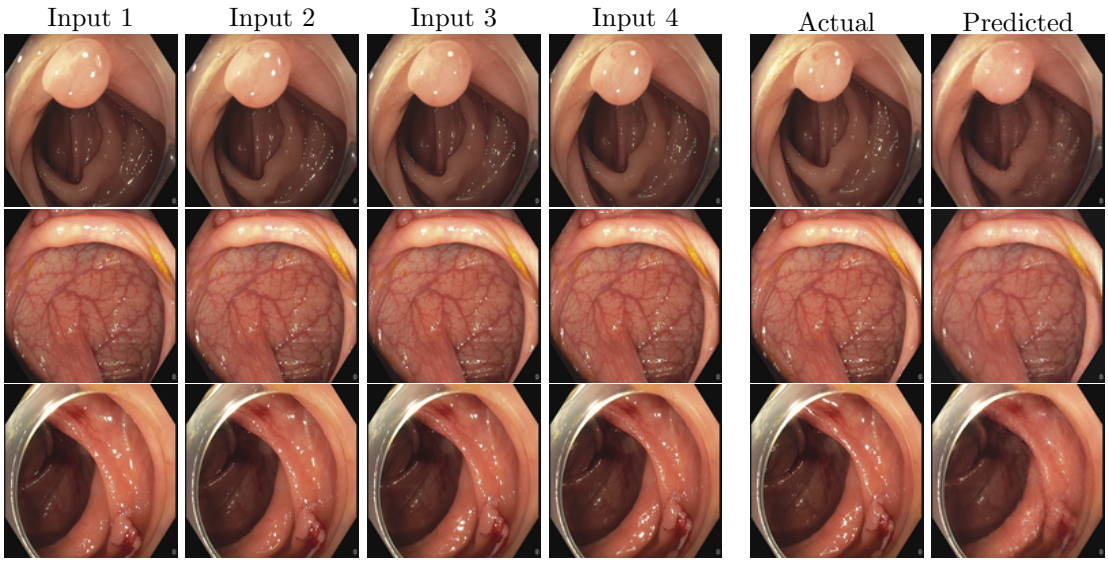


Figure 4.13: The figure shows the four frames that represent the stacked input to the model, the ground truth and the predicted output for the "2D output" experiment using early stopping.

image on an overall level. We do not observe any artifacts on the polyp or any other tissue. We observe both the large and the small features as greatly similar to the ground truth.

The middle row in Figure 4.13, shows four input frames from $t = 1$ to $t = 4$, followed by ground truth at $t = 5$ and the predicted output at $t = 5$ for input video 2. Overall, we observe the ground truth and generated image as greatly similar. We further observe thin and thick blood vessels clearly visible as well as the very thin vessels. We do not observe any artifacts on the polyp or other tissue. We observe both the large and the small features to be very sharp.

The lower row in Figure 4.13,shows four input frames from $t = 1$ to $t = 4$, followed by ground truth at $t = 5$ and the predicted output at $t = 5$ for input video 3. We observe the overall structures of the generated image to be greatly similar to the ground truth. We further observe the outgrowth with hemorrhages to be clearly visible with no artifacts but slightly blur. We do not observe any artifacts on tissue. We further find both the large features and the smaller features of the generated image as sharp.

Overall, we observe the three generated images in this experiment as greatly sharp. The images further show clear similarities with their corresponding ground truth. We observe no large or small artifacts. Furthermore, we observe a few local areas in one of the images as slightly blur.

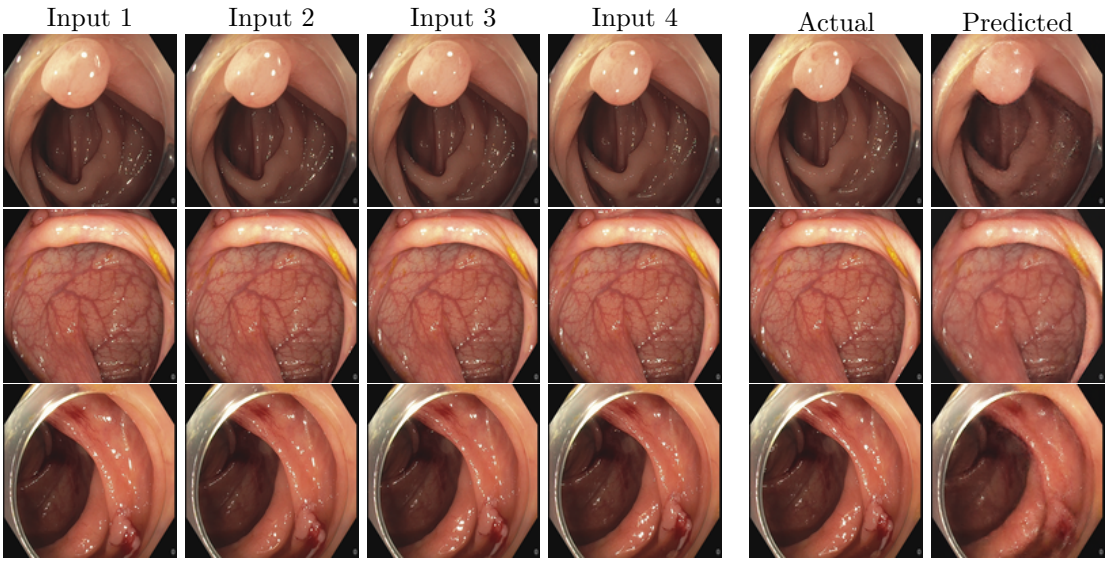
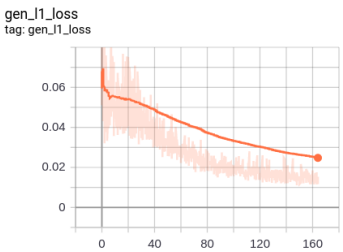


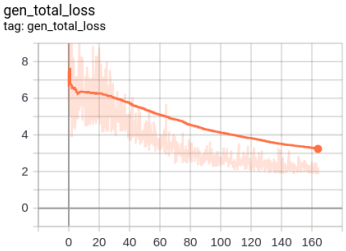
Figure 4.14: The figure shows the four frames that represent the stacked input to the model, the ground truth and the predicted output for "2D output" experiment with 1000 epochs.

(a) Discriminator Loss

(b) Generator Loss



(c) L1 Loss



(d) Total Generator Loss

Figure 4.15: The figure shows loss graphs for the "2D output" experiment.

4.4.10 Discussion

Throughout our experiments, we observe a progressive improvement on the majority of our generated output. The improvements are shown through image quality and realistic looking motions in the sequences. We observe that light

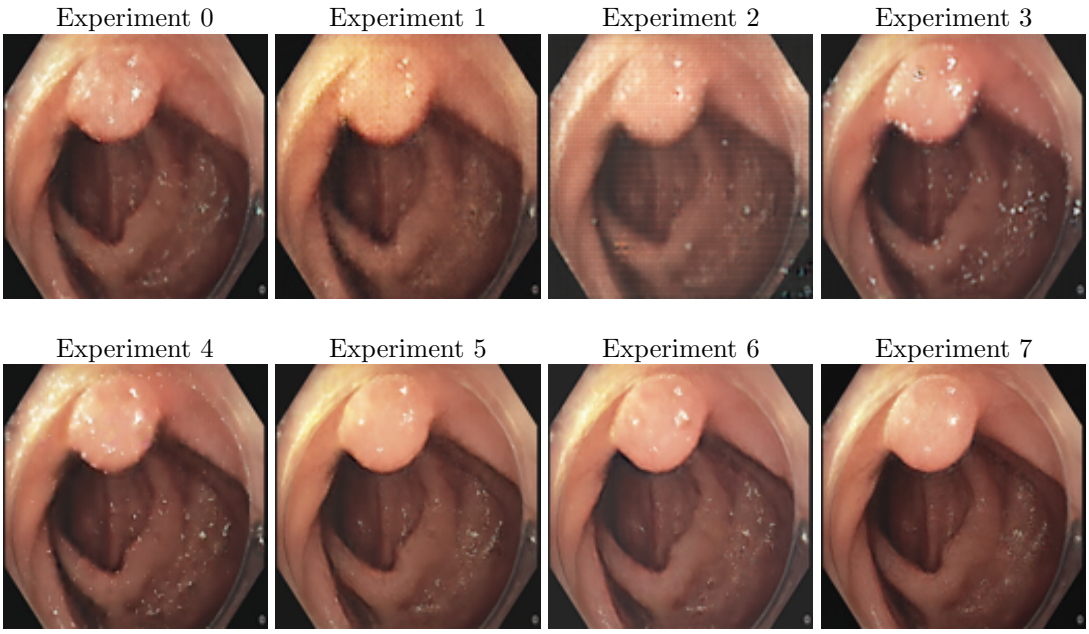


Figure 4.16: The figure shows the predicted output for all experiments form input video 1.

reflections on the input sequence cause noise or blur in the generated image. This issue gradually improves, and in our last experiment the artifacts have disappeared completely.

We observe a similar tendency on the blood vessels, where the thinner vessels are hardly visible in our first experiments due to blur. In the final experiment, we hardly see any difference between the blood vessels in the generated image and the ground truth. The image consisting of an outgrowth with hemorrhages was the most difficult one to improve. In early experiments, the generated images consisted of large and clearly visible artifacts. While in the last experiment, we were not able to observe any artifacts. Overall, these findings show a great improvement of our generator model.

In our second experiment: "change filter size" in Section 4.4.3, we observe a setback from changing the filter size. We assumed that the combination of parameters was not optimal at that particular stage, but would be beneficial on the final stage. We choose to conduct an additional experiment to confirm whether the filter change is beneficial to our final solution or not. The experiment confirmed that the filter change was beneficial at the final stage. We furter conducted an additonal experiment to test if our model would improve by training on more epoch. Therefor we trained our model on 1000 epochs, instead of using early stopping as we did on all the other experiments. The results from training on 1000 epochs are Shown in Figure 4.14. We found by visual inspections from generated videos that the model did not improve from this.

When we reduced the discriminator complexity, we achieved a great improvement on our sequences in terms of image quality and realistic looking motions. This was further corroborated by the generated images and the loss graph. In

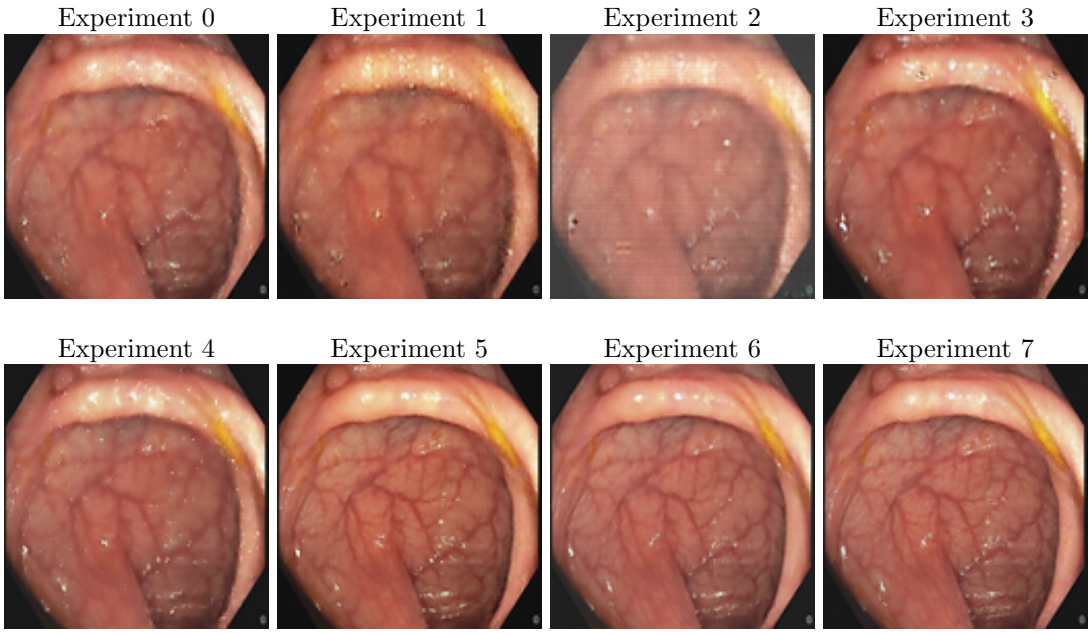


Figure 4.17: The figure shows the predicted output for all experiments form input video 2.

the next section we will describe how we aim to create a future video sequence based on our past input sequence.

4.5 Creating Videos from Vid2Pix

Setup We create a video from the model as an iterative process shown in figure 4.19. The iterative process is performed after we have trained the model. We input a sequence of four real images to the model, and then the model predicts the next image in the sequence. Then, we store the predicted image to a video and we create a new input sequence. The new input sequence is a shifted version of the previous sequence. We shift by removing the first image in the sequence, and adding the predicted image at the end of the sequence. Then, we continue until we get the desired video length.

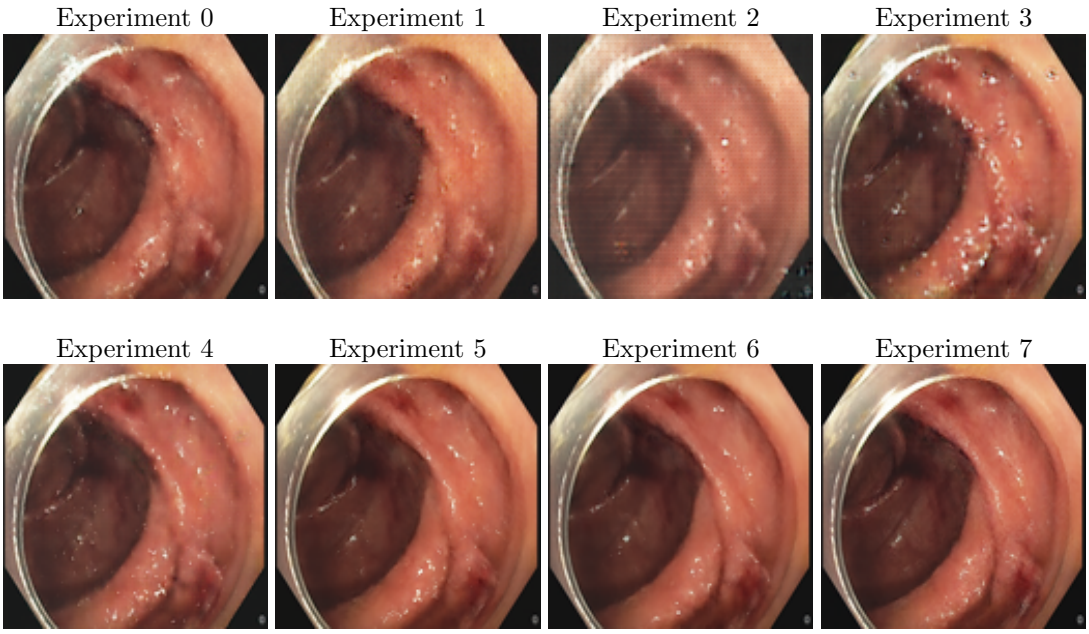


Figure 4.18: The figure shows the predicted output for all experiments form input video 3.

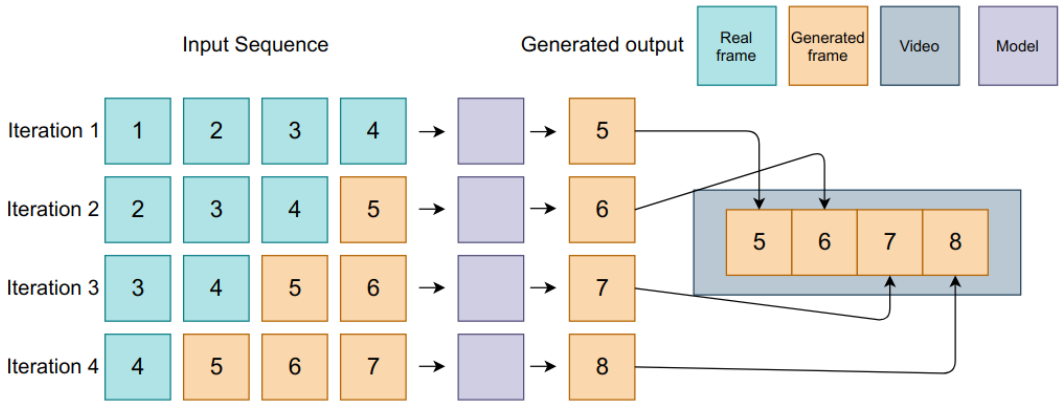


Figure 4.19: The figure shows how we create a 4 frame long video through 4 iterations

4.5.1 Create Videos from Generated 3D Output

The first six out of seven future sequence experiments take four frames as input and predict four new frames as output. We tested two different approaches for the six experiments. The first approach is using all the predicted frames to create a video.

1. Add 4 input frames to model
2. Predict 4 output frames
3. Add 4 predicted output frames to the video

4. Shift input sequence by removing frame number 1 and adding first predicted frame as frame number 4

5. Repeat until desired video length is reached

For the second approach, we used only the first out of 4 predicted frames for each iteration. The steps we follow to create a video is shown in the list below.

1. Add four input frames to the model.

2. Predict four output frames.

3. Add add the first out of the four predicted output frames to the video.

4. Shift the input sequence by removing frame number 1 and adding the first predicted frame as frame number 4.

5. Repeat until the desired video length is reached.

4.5.2 Create Videos from Generated 2D Output

The method we use to create videos from a 2D output is similar to the second approach we described in Section 4.5.1. The only difference is that we are only predicting one image and not four. The steps are described in the list below and are visualized in Figure 4.19.

1. Add four input frames to the model.

2. Predict one output frame.

3. Add the predicted output frame to the video.

4. Shift the input sequence by removing frame number 1 and adding the predicted frame as frame number 4.

5. Repeat until the desired video length is reached.

Results Video motions are difficult to see through a figure consisting of consecutive images. Therefor we choose to create and publish short video samples from all our experiments. The samples have been made available from the following GitHub repository: generated-videos [1]. To generate the videos from experiment 1 to 6, we used the second approach from Section 4.5.1. To create the videos from experiment 7, we used the approach from Section 4.5.2.

4.5.3 Discussion

Based on visual inspections of the videos we created with the two approaches in Section 4.5.1, we decided to use the second approach. We found that the second approach performed better on both image quality and created more realistic looking motion.

[1] `https://github.com/odaned/generated-videos`

4.6 Summary

In this chapter, we have looked into how we can predict future frames from a video. We further discussed how limited access to medical datasets led us to try to generate realistic-looking video sequences from an existing dataset. Next, we described the details of the dataset we used to conduct our sequence generation experiments. We have introduced the two deep learning model architectures, U-Net and Pix2Pix, which inspired of our work. This led us to an overview and a description of our main contribution to the thesis: Vid2Pix. We then went through our experimental process and the results that led us to the final model Vid2Pix. In the last part of the chapter, we describe how we conducted our video creation experiments and the associated results.

The next chapter, we do a thorough evaluation of the approach using standard quality metrics, an assessment done by trained medical doctors, and a case study on how this system may extend the size of existing datasets.

Chapter 5

Sequence Evaluation

In this chapter, we first look into the datasets that we use to conduct our experiments. We will further describe the different metrics we use to evaluate our results. Then, we introduce three different methods for evaluating the quality of our generated sequences. In the first method, we look at how the generated sequences have been inspected and assigned a score by two experienced medical doctors. The second method is a similarity measure, it measures how similar the generated frame and the ground truth is to one another. Finally we look at the results of a polyp classifier which has been trained both with and without artificial data.

5.1 Quality Assessment of Generated Videos

5.1.1 Dataset Description

In Section 3.1 we described how we divide the OUS-Study-2019 dataset into two separate datasets. In this chapter, we will be using the *CLSF-DAT* dataset. In this section, we will address how we aim to increase the size of the dataset by generating sequences. We will also introduce the ImageNet dataset in Section 5.2.1 that we use in our transfer learning experiments which are introduced in Section 2.4.1.

Even when the generated sequences contributes to an enlarged dataset, we still remain with an insufficient size of training data. With this concern, we choose to use pre-trained weights from the ImageNet dataset as input to our polyp classifier that will be introduced in Section 5.2. In the sections below, we will further describe the details of each dataset.

Combining OUS-Study-2019 with generated data

In Chapter 3, we discovered that the OUS-Study-2019 dataset contained duplicate frames with little movement. We applied dense optical flow explained in Section 3.3.3 on the dataset in order to filter away duplicate frames. We decided to use the same algorithm to optimize the *CLSF-DAT* training and validation dataset we use for polyp classification as well.

We start out with 23529 annotated polyp training images and 23529 pseudo-normal mucosa images. Moreover, we have 15520 annotated polyp validation

images and 15520 pseudo normal mucosa images. After filtering the data, we end up with a training dataset consisting of 626 polyp videos with a total of 5008 images and 642 normal mucosa videos with 5136. By applying dense optical flow on the validation dataset, we get a validation dataset consisting of 328 polyp videos with a total of 2624 images and 356 normal mucosa videos with a total of 2848 images. We do not apply dense optical flow on the test datatset. The test data consists of 15107 polyp images and 15107 normal mucosa videos.

Dataset	Train	Train Filtered	Val	Val Filtered	Test
Original	47 058	10 144	31 040	5 248	30 208
Original + Generated	47 058	10 144 + 10 144	31 040	5 248	30 208

Table 5.1: The table shows an overview of the resulting number of images that is used for training, validation and testing the polyp classifier. Train Filtered and Val Filtered stands for trainig data and validation data after optical flow.

We split the training dataset into three sets consisting of train, validation and test. For train, validation and test set were distributed as follows:

- 626 polyp training videos with 8 frames in each video
- 642 pseudo normal mucosa training videos with 8 frames in each video
- 328 polyp validation videos with 8 frames in each video
- 356 pseudo normal mucosa validation videos with 8 frames in each video

5.1.2 Evaluation Metrics

Evaluating whether a generated video looks realistic or not is a challenging task. Our goal is to generate videos which look realistic enough to be used as training data for a neural network. If we can achieve this, we can improve one of the biggest problems which exists in the field today, which is the lack of annotated training data. To evaluate this, we use both objective and subjective assessment methods.

We want an objective evaluation of how good a generated video is, ideally a number which quantifies how realistic a video looks. Note that a number which measures the similarity between the ground truth and all the frames of a generated video, might not represent what we are trying to measure. After all, we are interested in generating new frames which look different from the ground truth.

In order to evaluate the first generated frame from our model, we chose to use three popular similarity measurement methods [114]: mean square error (MSE), peak signal to noise ratio (PSNR), and structural similarity (SSIM) index.

Mean Square Error

When we generate an image from our generator model, we first downsample our images to a lower resolution to extract lower dimension features. Then we upsample or reconstruct our image back to its original resolution. In this process

information will be lost, and to quantify the loss we use the MSE metric. With MSE, we can quantify the image reconstruction quality. The MSE computes the average square difference between the estimated value and the actual value. In our case, we estimate the average square between the first (out of four) generated frame and the ground truth.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

A low MSE means that the two images as very similar. Thus we want the MSE to get as close to zero as possible.

Peak Signal to Noise Ratio

Peak Signal to Noise Ratio (PSNR) is an approximation to human perception of reconstruction quality. Similar to MSE, PSNR measures the image reconstruction quality and is a logarithmic representation of MSE. PSNR measures the peak error between the images. Due to the range of validity of the metric, researchers do not recommend using the metric to compare results across domains [40]. In this thesis we use the metric to compare results from experiments in the same domain. The PSNR value is dependent on the properties of the data and it can theoretically go up to infinity. A higher PSNR value indicates a better result.

$$PSNR = 10\log_{10}\left(\frac{R^2}{MSE}\right)$$

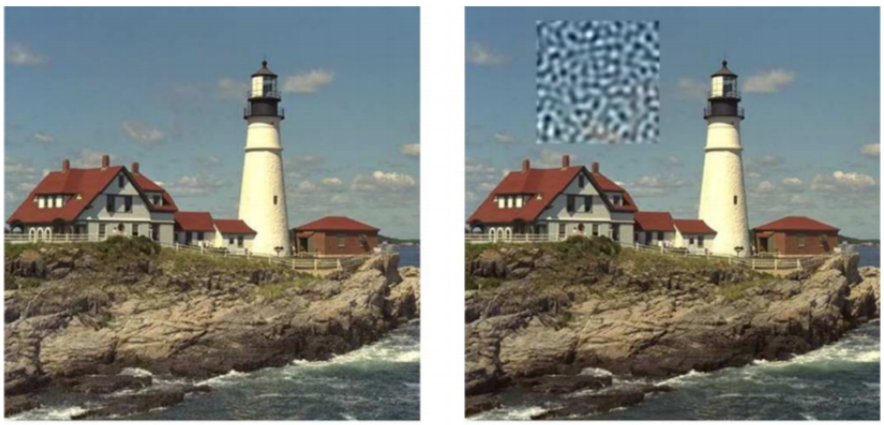


Figure 5.1: Two images with different perceived image quality but with identical PSNR value.

Structural Similarity Index

Compares local patterns of pixel intensities [106]. It proves to provide a better approximation to perceived image deformation compared to PSNR and MSE [37]. A higher SSMI value indicates a higher similarity between the images.

Confusion Matrix

A popular way to visualize how well a model performs is through a confusion a matrix. A confusion matrix visualizes how many samples that actually are from class A, are classified as class A. It further visualizes how many classes that are wrongly classified. This applies for all classes that the model is trained to predict.

5.1.3 Subjective Assessment of Generated Videos

Generated videos are tricky to evaluate from metrics alone. Evaluating if a scene look realistic or not is especially tricky. We have chosen to also conduct a subjective, human assessment to evaluate the generated videos. We recruited two medical doctors with experience in working with endoscopy data to participate in the assessment. All quotes that are listed under the "comments" paragraphs are directly translated from Norwegian to English.

Participants

Reviewer 1 is a medical doctor with 2 years of experience on evaluating endoscopy data.

Reviewer 2 is a medical doctor and gastroenterologist. The reviewer is specialized in quality improvement and artificial intelligence in endoscopy. The reviewer has extensive experience in endoscopy examinations and data evaluation.

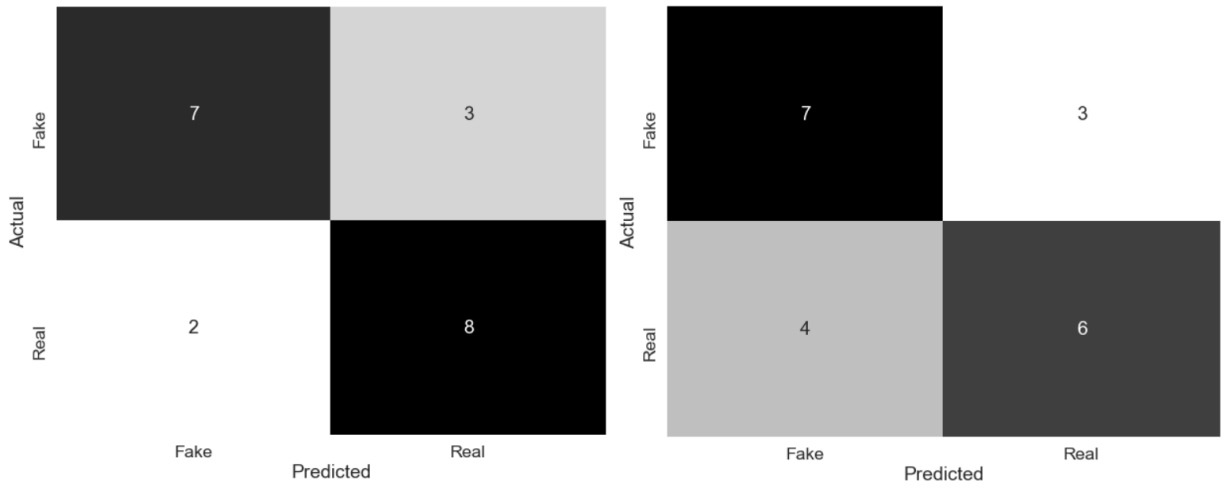
Assessment Setup and Results

The assessment is divided into two sessions. The first session involves classifying videos into two classes, either "real" or "fake". In the second session, it is revealed that all videos are fake. The subjects are then asked to rank the fake videos, by how realistic they look.

Assessment Session 1

Setup Each reviewer was given access to a folder containing 20 videos, out of which 10 were artificial and 10 were real. The reviewers are not given any information about how many videos are real or fake. They are informed that the folder consists of both real and fake colonoscopy videos. Then they are asked to view the videos, and classify each video as either real or fake. The following instructions were given to the reviewers in session 1:

- You will get access to a folder in the cloud consisting of 20 real and fake colonoscopy videos.



(a) (b)

Figure 5.2: The figure shows the resulting confusion matrices from the polyp classification assessment. (a) shows classification done by reviewer 1 part 1. (b) shows classification done by reviewer 2 part 1.

- Write down the video name together with your evaluation: fake or real.
- Return your evaluation together with comments, if any.

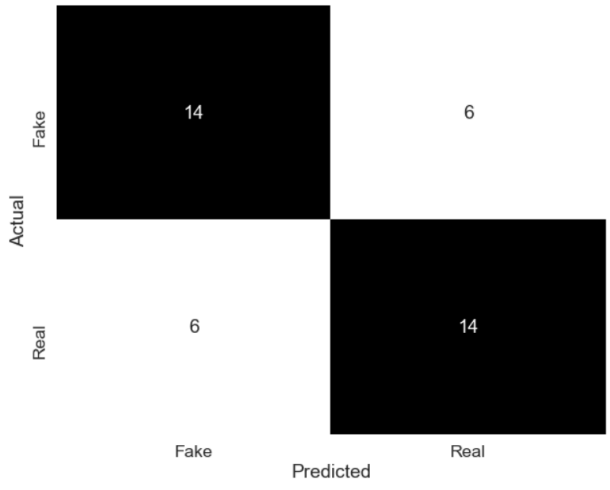


Figure 5.3: The figure shows a confusion matrix with the total classification results from both reviewers part 1

Results

Comments After the session, we were given feedback from the reviewers. All quotes are directly translated from Norwegian to English. Here are the comments from session 1:

> *I did not feel one hundred percent sure of my own evaluation*
>
> *There was only one video that I was sure was real*
>
> *Some of the videos had strange artifacts and lines that look unrealistic*
>
> *Some of the videos consists of a few strange motions*
>
> *Videos are difficult to evaluate because they are short and the resolution is low*

From the comments, we find that there is uncertainty related to own assessment. This may indicate that the generated videos look realistic, which makes it hard to distinguish them from those that are real. It may also indicate that both the real and the fake videos are of bad quality and therefor hard to distinguish. Another comment addresses that some of the videos consists of strange artifacts and motions, this indicates that there is still room for improvement on both the image quality and motion.

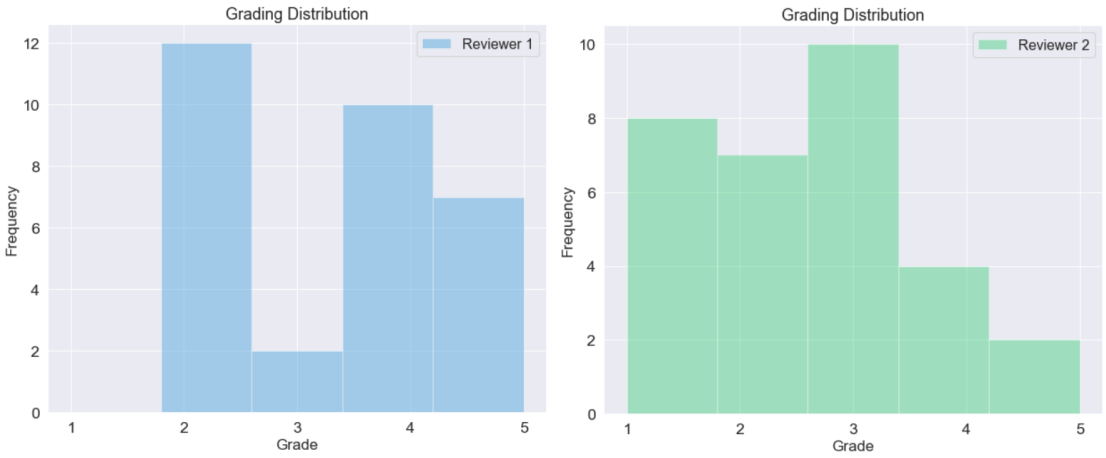
On the last comment, the reviewer addresses the difficulties related to short video lengths and low resolution. Since our goal was to generate more data to train a classifier, we did not seek to generate long sequences. Based on this, we could not provide long sequences for the reviewers. However, a higher image resolution would be preferable.

Assessment Session 2

Setup Each reviewer had access to a folder consisting of 31 fake videos. The reviewers are informed that all videos are fake. They are then asked to grade how the videos from 1 to 5, based on how real they think the videos look, where 1 is least real and 5 is most real. A full description of the scale is shown in the list below.

The table below shows the instructions that were given to the reviewers in session 2 of the experiment:

- You will get access to a folder consisting of only fake colonoscopy videos.
- Write down the video name together with your rating on a scale of 1 to 5 where:
    1. Obviously fake
    2. Likely fake
    3. Unsure
    4. Seems real, but with some artifacts
    5. Very real
- Return your evaluation together with comments, if any.



(a) (b)

Figure 5.4: The figure shows grading distributions from assessment 2. 31 generated videos were graded on a scale from 1 to 5. (a) shows the grading distribution done by reviewer 1. (b) shows the grading distribution done by reviewer 2.

Here we present the results from session 2 of the assessment. We choose to present the results as a histogram. The histogram shows the grading distribution after grading 31 generated videos. The grading is done by two reviewers. The grading scale goes from 1 to 5, where 1 is "Highly visible generated" and 5 is "Very real".

Results We can see from Figure 5.4a that reviewer 1 did not grade any of the the generated videos as 1: "Highly visible generated". Moreover, a majority of 12 videos were graded as 2: "Visible generated". A minority of 2 videos were graded as 3: "Unsure". Furthermore, 10 videos were graded as 4: "Seems real, but with some artifacts", and finally 7 videos were graded as 5: "Very real".

In Figure 5.4b. Reviewer 2 graded 8 of the generated videos as 1: "Highly visible generated". Furthermore, 7 videos were graded as 2: "Visible generated". A majority of 10 videos were graded as 3: "Unsure". Moreover, 4 videos were graded as 4: "Seems real, but with some artifacts", and finally a minority of 2 videos were graded as 5: "Very real".

In Figure 5.1.3, we compare the distributions. The histogram provides variable results from the two reviewers. We see that reviewer 2, grade the majority of the videos with grades below or equal to 3. While reviewer 1 grade the majority of the videos as above or equal 3.

Comments After the session, we were given verbal feedback from the reviewers. All quotes are directly translated from Norwegian to English. Here are the comments from session 2:

> *I graded videos that were characterized by strange movements with the grade 2*

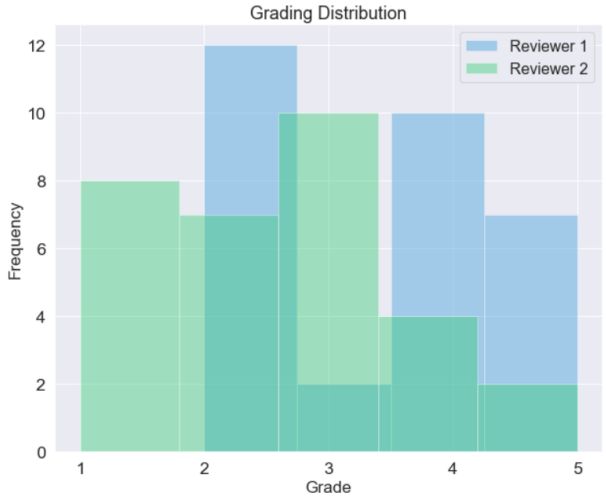


Figure 5.5: This figure shows a comparison of the grading distributions from both reviewers. 31 generated videos were graded on a scale from 1 to 5.

> *If the videos were presented as images instead of videos, I would grade all of them as very real*
>
> *There were no obvious artifacts that revealed that the videos were fake*
>
> *All the colors and shapes looked real, including the videos that were graded as visible generated*
>
> *It did not add much extra to present the material as short videos. You would have achieved the same by presenting them as a sequence of images*
>
> *With my experience in evaluating videos, the videos should be at least 10 seconds long*
>
> *I consider this method to be relevant for detecting other abnormalities like lymphoma or angiectasis*

One reviewer reveals that if the videos were presented as images, they would all be graded as real. This indicates that the videos provide some extra information to the reviewer that would be gone by presenting them as images. This is supported by another comment, where the reviewer observes strange movements. These movements would perhaps not be possible to observe through still images. Another comment reveals that the reviewer find shapes and colors in the videos to look real. This proves our method to work well in generating realistic looking videos. This was supported by another reviewer that stated that there were no obvious artifacts that revealed the videos to be fake. A rather helpful comment for future work, was that that the reviewer found the method relevant for detecting other abnormalities.

Discussion

From session one of the subjective assessment, we learned that 6 real videos were mistaken for being fake. Moreover, 6 videos were mistaken for being real when they were fake. This may indicate that low resolution images makes it difficult to decide whether the video is real or fake. It may also indicate that the videos look real, which makes it hard to distinguish real videos from fake videos. From a comment, we learned that evaluation of the sequences could have worked better by evaluating them side by side. Contrary to this, we found from another comment that the reviewer easily could find strange motions and tissue through looking at the videos. This comment gives an indication that looking at videos instead of image sequence can be helpful to detect artifacts.

From session two of the subjective assessment, we learned that there were great differences in grading from the two reviewers. The mean grades from reviewer 1 was 3.4, while the mean was 2.8 for reviewer 2. This indicates that the reviewers have different perceptions on video quality and calibration of the grading scale. It also shows a problem with subjective measures, they vary from person to person.

5.1.4 Evaluation by Similarity Measure

When we create a fake video which is visualized in Figure 4.19, we start with four real input frames to the generator. For each frame we generate, we replace one real frame by one fake frame to the input of the generator. For every new generated frame, the image quality is gradually weakened. Yet, we want an indication on how well the generator perform on generating realistic structures and motion based on a sequence of exclusively real input frames and a real ground truth.

First, we want to find out if the generated frame looks real i terms of structure and motion. Second, we want to find out if the first generated frame in a video is similar to its ground truth. Based on these considerations, we perform similarity measurements on the first generated frame and the corresponding ground truth on all generated videos for all experiments by using the *CSLF* test dataset. Here, we introduce our evaluation done by comparing the first generated frame to its corresponding ground truth. We do this for all generated experiments.

Results

Experiment	PSNR	MSE	SSIM
Original Pix2Pix	72.1301	0.0050	0.8011
Replace with 3D layers	71.5228	0.0057	0.7856
Change filter size	70.9309	0.0065	0.7713
Offset downsampling	70.9941	0.0063	0.7760
Reduce discriminator complexity	71.8187	0.0054	0.7944
Keep more features	72.6833	0.0047	0.8292
Add noise	72.4309	0.0050	0.8230
2D output	73.3718	0.0042	0.8409

Table 5.2: This table shows the average PSNR, MSE and SSIM measurements between one generated image and one the ground truth per video. The measurements are done on a total of 626 videos from the *CLSF-DAT* dataset.

Discussion

SSIM has been shown to give a better approximation to perceived image deformation [37]. We still prefer to use the PSNR and MSE matrics to substantiate the the SSIM metric. Table 5.2 shows the PSNR, MSE and SSIM similarity measurements. From the table we observe that when we modify the original Pix2Pix model by using 3D convolutional and deconvolutional layers, the SSIM and PSNR value first decreases, and MSE increases. SSIM then gradually increases until we meet a larger jump after reducing the discriminator complexity. Then, on the last experiment we meet another larger jump, where we change our model to predict an image instead of a sequence. This may indicate that reducing the discriminator complexity and changing the output dimension has a positive effect on the quality of generated output.

5.2 Case study on real world use case scenarios

In order to test if our additional generated polyp videos can help to improve the performance of a polyp classifier, we setup an image classification experiment. We use a state-of-the-art image classification model provided by the Keras API.

5.2.1 ImageNet

ImageNet is a large scale, publicly accessible image database provided by Princeton and Stanford University [42]. As of April 2010, the database contains more than 14 million images, where each image belongs to one of 27 high-level categories and tens of thousands of subcategories. The ImageNet categories vary widely, ranging from musical instruments to animal species [42]. A few example images from the ImageNet database is shown in Figure 5.6. ImageNet Large Scale Visual Recognition Challenge (ILSVRC) was a competition held yearly between 2010-2017. The objective of the challenge is to evaluate and compare the progress of algorithms for computer vision by using a large scale

image database. Another important goal of the challenge is to benchmark the state-of-the-art within computer vision.

The deep learning API Keras[1], provides a selection of deep learning models with optional pre-trained weights from the ImageNet database. For our image classification experiment, we will be using the ResNet50 deep learning model which has been pre-trained on ImageNet.

Figure 5.6: The figure shows a few examples of species of birds that are present in the ImageNet database [80]

5.2.2 Metrics

We want to train a model to be able to classify our images into one of two classes: polyp or normal. In order to evaluate the performance of our model, we can use multiple metrics.

Accuracy

The Accuracy metric used for evaluating how good a model is to predict the correct class. The metric is used for both evaluating multi-class classifiers and binary classifiers. Accuracy has the following definition:

$$Accuracy = \frac{N_{correct}}{N_{total}}$$

Accuracy is the percentage of the correctly classified images. Where $N_{correct}$ is the number of correctly classified images, and N_{total} is the total number of classified images. For binary classification problems, like ours, we can additionally define the accuracy in terms of positives and negatives:

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

Where:

- True Positive (TP): images classified as True are actually True
- True Negative (TN): images classified as False are actually False
- False Positive (FP): images classified as True are actually False
- False Negative (FN): images classified as False are actually True

[1] Keras `https://keras.io/`

Recall

Recall can also be referred to as sensitivity or true positive rate (TPR). Recall is the measure of completeness. It is not sensitive to data distributions [33] and it shows the models ability to correctly classify polyps out of all the images that consists of polyps.

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

Precision

Precision is also referred to as positive predictive value (PPV). Precision is a measure of exactness. It is sensitive to data distributions [33] and gives the percentage of the correctly classified polyps out of the total correctly classified samples.

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

Specificity

Specificity is also reffered to as the true negative rate (TNR). Specificity gives the models ability to correctly classify normal mucosa out of the images that consists of normal mucosa.

$$TNR = \frac{TN}{TN + FP}$$

F1 Score

The F1 score measures the effectiveness of classification. It is the weighted average of recall and precision. As Precision, F1 Score is additionally sensitive to data distributions [33]. If we have an uneven class distribution, F1 score can give a better measure of the performance of the model than Accuracy.

$$F1 = 2 \times \frac{PPV \times TPR}{PPV + TPR}$$

Matthews correlation coefficient (MCC)

Matthews correlation coefficient (MCC) takes all TP, TN, FP and FN into account. It produces a high score only if all of the mentioned values give a good score. This means that TP and TN should give as high score as possible and FP and FN should give as low score as possible. The metric outputs a score between -1 and 1, where 1 means that all samples are perfectly classified, while -1 means that the model did a bad classification.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

5.2.3 ResNet50

ResNet is a deep convolutional neural network. The ResNet model architecture was the winner of the ImageNet challenge in 2015 [34]. Its main contributions comes from allowing to train very deep neural networks without encountering

the vanishing gradient problem [67]. ResNet uses skip connections to pass information from earlier layers to later layers in the model. The skip connections are used to reduce the vanishing gradient problem. ResNet50 is one out of many proposed versions of ResNet that vary by depth [34]. ResNet 50 has a depth of 50 layers. We choose to use ResNet50 as our binary classifier due to its great proven performance [34] and fewer trainable parameters than other models that are available in the Keras API [93].

5.2.4 Transfer Learning with ImageNet

Before we start training the ResNet50 model with our *CLSF-DAT* dataset, we use transfer learning to re-train a trained network. The network has been trained using the ImageNet dataset 5.2.1. We extract pre-trained weights by using the transfer learning method called fine-tuning 2.4.1. By fine-tuning, we mean that we initialize our network with weights that are trained on a different dataset. Then we continue training the weights on a new dataset, without freezing any layers. We chose to use the ImageNet data because it was easily accessible by being available as a part of the ResNet50 classification model in the Keras API presented in Section 5.2.3.

5.2.5 Polyp Classifier Experiment

In order to evaluate the usefulness of the generated polyp videos, we will train a polyp classifier both with and without using the artificial data. If the classifier which has seen artificial data outperforms the other classifier, we can conclude that introducing artificial data can be useful.

The first classifier is trained on the *CLSF-DAT* dataset only, while the second is trained on the *CLSF-DAT* dataset combined with the artifical dataset that is generated from the *CLSF-DAT* dataset.

The goal of our classification experiment is to evaluate the effect our artificial data has when training a polyp classifier. We train two models, one with real data only, and one with real and fake data. We then compare the accuracy of both models, using a separate test dataset which only contains real data.

Experiment Setup

The steps of our classifications experiments go as follows:

1. Create a ResNet50 binary classifier with transfer learning from the imagenet database

2. Train the classifier on the original dataset containing the two classes: polyp and normal mucosa

3. Add the generated polyp videos to the original dataset and train the classifier again. This training dataset will be unbalanced.

4. Test the performance of the two trained models using the same test dataset.

Hyperparameters ResNet50	Values
Epochs	200
Batch Size	32
Optimizer	SGD
Learning Rate	0.001

Table 5.3: This table shows the hyperparameters used for training the ResNet50 model

5.2.6 Results

Here we present the loss graphs for training and validation of the identical models. Where one model where trained on the *CLSF-DAT* dataset, while the second model were trained on the *CLSF-DAT* and the generated dataset combined.

Training with original and generated data combined

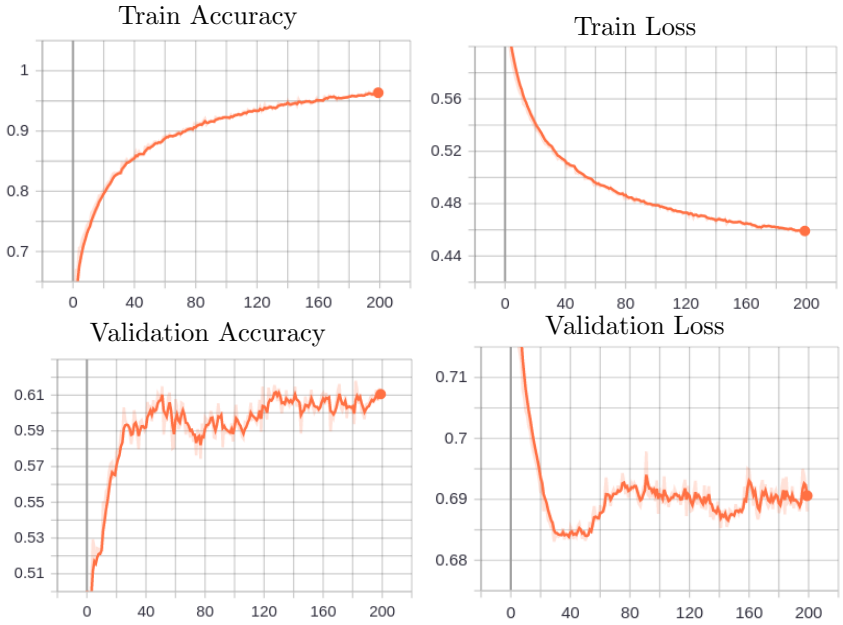


Figure 5.8: The figure shows train and validation accuracy and loss during training of ResNet50 pre-trained on ImageNet using the *CLSF-DAT* dataset combined with generated generated polyp videos. The dataset is unbalanced.

Training with original data

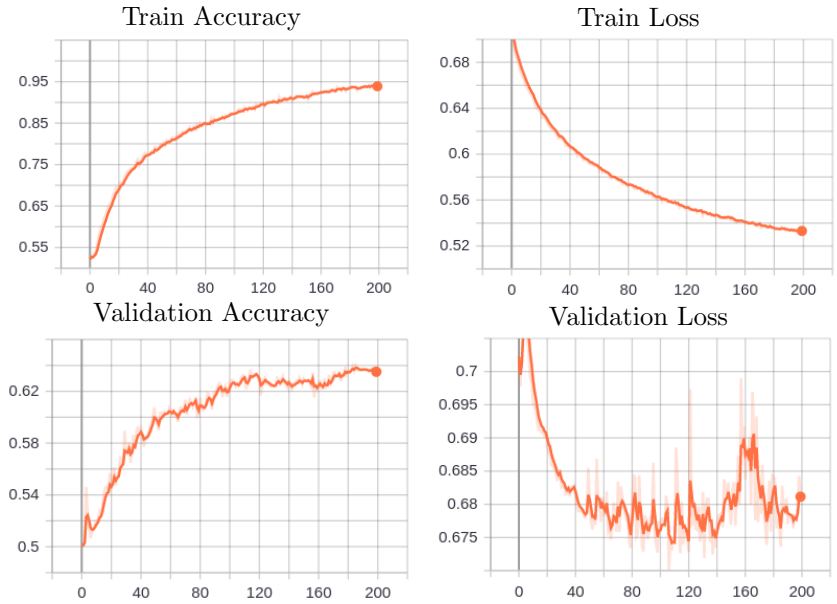


Figure 5.9: The figure shows train and validation accuracy and loss during training of ResNet50 pre-trained on ImageNet using the *CLSF-DAT* dataset.

Here we present the resulting metrics after testing the two identical models that were trained on different datasets. We have used the *CLSF-DAT* test dataset to test the models.

Dataset	ACC	REC	PREC	SPEC	F1	MCC
Original	0.678	0.620	0.702	0.737	0.658	0.359
Original and Generated combined	0.635	0.687	0.623	0.584	0.653	0.272

Table 5.4: This table shows metrics from prediction on two models: One trained on the *CLSF-DAT* dataset and one trained on *CLSF-DAT* dataset combined with generated polyp videos. Both models are tested on the *CLSF-DAT* test dataset

5.2.7 Discussion

Based on the MCC value, the polyp classification results did not give an overall good improvement from adding the fake polyp data to the original dataset. However, from Figure 5.10b and Figure 5.10a we can observe that the number of correctly classified polyps has increased from 9362 to 10375 when add the fake dataset to train the model. We further observe that the number of wrongly classified polyps have additionally decreased from 5739 to 4726 when we add the fake dataset to train the model. On the other hand, the number of correctly

classified normal mucosa has decreased and the number of miss-classified normal mucosa has increased. F1 Score, which is close to similar from both models, gives an indication that model performance is unchanged and that the unbalanced data distribution may have an effect on the overall performance of the model that has been trained with fake data.

By inspecting the data, we discovered errors that involved in multiple blurry frames which is shown in Figure 5.11. The blurry frames were present in both the polyp class and the normal mucosa class. Many of the blurry frames are covering crucial parts of the image where the polyp should be located. In the Section 4.4.10 we discussed how our generator were generating bad frames consisting of many artifacts if input was blurry. We believe that by removing blurry frames from the dataset, we can improve generated data from the sequence generator. Additionally, we believe that removal of blurry frames from the training data can have an effect on how the model perform both with and without the fake training data.

5.3 Summary

In this section, we introduced the second part of our dataset: *CLSF-DAT*. We explained how we generated artificial polyp videos by using this dataset as an input.

We further introduced metrics to compare and evaluate the quality of the generated sequences. From the measures, we found that our experiments progressively improved and that our final model achieved a better similarity measures than Pix2Pix.

Then we explained how we conducted the a subjective assessment for our generated videos. The assessment showed that quality and motions where great in some videos, and others had clearly visible artifacts. We also learned that the two reviewers did grade the videos very different and which makes subjective measures difficult to interpret, because they vary from person to person.

Finally, we evaluated the usefulness by training two identical polyp classifiers on *CLSF-DAT*, with and without adding the generated videos. F1 Score indicated that the performance of the model trained on the unbalanced dataset was unchanged. We further discovered blurry frames in the data, which may have had en effect on the results.

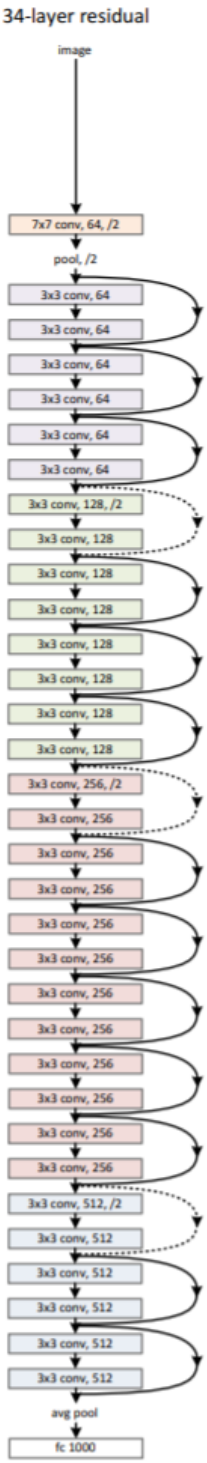
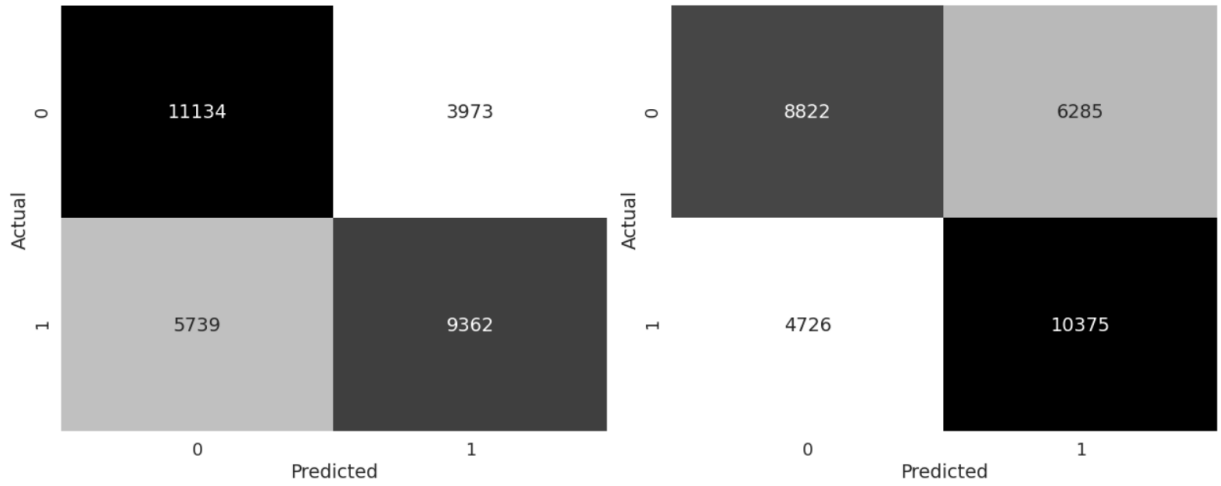


Figure 5.7: The figure shows the model architecture of ResNet34 which has a depth of 34 layers. ResNet50 is similar to ResNet34 besides the depth. ResNet50 consists of 50 layers. [34]



(a) (b)

Figure 5.10: The figure shows the resulting confusion matrices after classifying an unseen test dataset consisting of 30 208 images where 1 represents polyps and 0 represents normal mucosa. (a) shows the results from the model trained on the *CLSF-DAT* dataset alone and (b) shows the results from the model trained on *CLSF-DAT* dataset combined with the fake polyp videos.

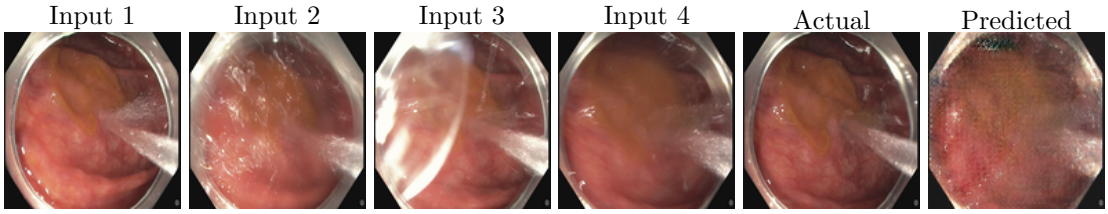


Figure 5.11: Blurry input

Chapter 6

Conclusion and Future work

6.1 Summary and Contributions

Colorectal cancer is a great health issue for men and women in the United States. We have learned that early detection of polyps is important for a positive outcome. A major challenge today is that medical doctors do not find every polyp during an examination. Computer aided disease detection systems have been developed to help physicians improve their detection rate. Such systems often consist of deep learning models with high complexity. Perhaps the most important piece of building a good system is the **quality of the dataset**. Unfortunately, there are no publicly available large scale endoscopy datasets. In order to address the problem of not having a large, high quality dataset, we proposed in section 1.2 a system that aims to answer the following research question:

> *Can generative models be used to generate realistic-looking videos of colon polyps?*

To answer this question, we have defined three objectives:

Objective 1 *Prepare the training data in a way that optimizes network leaning and avoids overfitting to specific video frames.*
This objective stems from the common need for quality training data when developing deep neural networks. We developed a preprocessing framework, based on optical flow, which was able to filter away stationary frames of a video. We compared computational cost of optical flow to the benefits of an higher quality dataset. We find that the preprocessing greatly improves the quality of the generated videos.

Objective 2 *Generate artificial videos of colon polyps using generative adversarial networks (GANs).*

This objective is the main contribution of this thesis and comes from the requirement of needing more labeled data in the medical sector. We developed a CGAN to generate future frames given an existing video. The key parts of the model were the 3D convolutional and deconvolutional layers. These layers made it possible to create realistic looking spatiotemporal features. We found

that the model was able to successfully generate videos of colon abnormalities. This achievement could reduce the need for collecting labeled datasets.

Objective 3 *Perform a thorough evaluation of the generated videos using a quantitative and qualitative approach, in addition to evaluating the fake videos on a real-world use-case.*

This objective aims to answer if the generated videos are of sufficient quality to be useful. We conducted a thorough evaluation of the generated videos, using both objective and subjective measures. Using a similarity measure as a metric, we could see gradual improvements throughout our experiments. The final model achieved an SSIM score of 0.84. In comparison, the original Pix2Pix model got a score of 0.80. Further we achieved a PSNR score of 73.37 and MSE score of 0.0042 compared to 72.13 and 0.0050 respectively with Pix2Pix from 5.2. These results show that our Vid2Pix model outperforms the Pix2Pix model for artifical video generation.

Two physician also evaluated the quality of the generated videos. The reviewers classified 3 out of 31 videos each to be real that were actually fake. Based on these results, we are not able to make an objective conclusion on the quality of the videos. The assessment mainly taught us new perspectives on real-world use-cases and how we can improve a subjective assessment for human spatiotemporal perception.

We also trained an image classifier on a dataset with and without adding the artificial videos. We found that using artificial videos increased the number of correctly classified polyps, but simultaneously the number of miss-classified normal images increased. Thus the our overall model classification results did not improve. We believe that multiple blurry frames caused some of these problems. Our findings showed that blurry frames caused output consisting of artifacts from our generator. In addition, we discovered that the blur caused some polyps to be less visible. More work would be needed in order to find why it did not improve.

Regarding the overall research question, our experimental results show that our proposed generative models can generate realistic polyp frames. This is supported by the fact that trained and experienced gastroenterologists struggle to differentiate between real and fake videos.

6.2 Future Work

The OUS datasets contains 80000 images, where some are of low quality. We believe that spending more time on cleaning this dataset would improve both the video generator and the polyp classifier. For example, we believe that by removing blurry frames would be beneficial. One physician commented that the low resolution of the images, 128×128, made it difficult to assess the image quality. We believe that increasing the size, for example to 256×256 or larger, would result in higher quality videos. Note that this would greatly increase the time it takes to train a model.

We would additionally test how many generated future frames we can add to a training dataset in order to improve a polyp classifier. This could be done progressively, starting with one predicted frame per video.

We would further try to train a model on a greater variety of colonoscopy recordings. We could do this by training an image classifier from the Kvasir dataset [69] and further retrain the model on the OUS-2019 dataset. We could additionally test the same approach by using a sequence classifier that is pre-trained on videos from Hyperkvasir [7]. We believe that this approach could help generalize the model better to data in the same domain.

As suggested by one medical doctor from the subjective assessment, training a generator on other abnormalities like lymphomas could be helpful in detecting other abnormalities.

Capsule endoscopy is a promising advance in screening. Our final suggestion would be to test our system by using data from our recent publication on a video capsule endoscopy dataset, where we contributed in annotating the dataset: *Kvasir-Capsule, a video capsule endoscopy dataset* [86].

Bibliography

[1] Martin Arjovsky and Léon Bottou. "Towards Principled Methods for Training Generative Adversarial Networks". In: *arXiv:1701.04862 [cs, stat]* (Jan. 2017).

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein GAN". In: *arXiv:1701.07875 [cs, stat]* (Dec. 2017).

[3] Simon Baker and Iain Matthews. "Lucas-Kanade 20 Years On: A Unifying Framework". en. In: *International Journal of Computer Vision* 56.3 (Feb. 2004), pp. 221–255. ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000011205.11775.fd.

[4] Jorge Bernal and Histace Aymeric. *Gastrointestinal Image ANAlysis (GIANA) Angiodysplasia D&L challenge*. https://endovissub2017-giana.grand-challenge.org/home/. Accessed: 2017-11-20. 2017.

[5] Jorge Bernal and Histace Aymeric. *MICCAI Endoscopic Vision Challenge Polyp detection and segmentation*. https://endovissub2017-giana.grand-challenge.org/home/. Accessed: 2017-12-11. 2017.

[6] Jorge Bernal et al. "WM-DOVA maps for accurate polyp highlighting in colonoscopy: Validation vs. saliency maps from physicians". In: *Computerized Medical Imaging and Graphics* 43 (2015), pp. 99–111. DOI: 10.1016/j.compmedimag.2015.02.007.

[7] Hanna Borgli et al. *HyperKvasir, A Comprehensive Multi-Class Image and Video Dataset for Gastrointestinal Endoscopy*. Tech. rep. Open Science Framework, Dec. 2019. DOI: 10.31219/osf.io/mkzcq.

[8] Dongdong Chen et al. "Coherent Online Video Style Transfer". In: *arXiv:1703.09211 [cs]* (Mar. 2017).

[9] François Chollet. "Xception: Deep Learning with Depthwise Separable Convolutions". In: *arXiv:1610.02357 [cs]* (Apr. 2017).

[10] Timothy Cogan, Maribeth Cogan, and Lakshman Tamil. "MAPGI: Accurate Identification of Anatomical Landmarks and Diseased Tissue in Gastrointestinal Tract Using Deep Learning". In: *Computers in Biology and Medicine* 111 (Aug. 2019), p. 103351. DOI: 10.1016/j.compbiomed.2019.103351.

[11] *Colon Polyps - Symptoms and Causes*. https://www.mayoclinic.org/diseases-conditions/colon-polyps/symptoms-causes/syc-20352875.

[12] *Colonoscopy - Mayo Clinic*. https://www.mayoclinic.org/tests-procedures/colonoscopy/about/pac-20393569.

[13] *Colonoscopy — NIDDK*. https://www.niddk.nih.gov/health-information/diagnostic-tests/colonoscopy.

[14] *CS 230 - Recurrent Neural Networks Cheatsheet*. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks.

[15] Peter J Denning et al. "Computing as a Discipline". In: *Communications of the ACM* (), p. 15.

[16] Zhen Ding et al. "Gastroenterologist-Level Identification of Small-Bowel Diseases and Normal Variants by Capsule Endoscopy Using a Deep-Learning Model". In: *Gastroenterology* 157.4 (Oct. 2019), 1044–1054.e5. DOI: 10.1053/j.gastro.2019.06.025.

[17] Zihan Ding et al. "TGAN: Deep Tensor Generative Adversarial Nets for Large Image Generation". In: *arXiv:1901.09953 [cs]* (Mar. 2019).

[18] Vincent Dumoulin and Francesco Visin. "A Guide to Convolution Arithmetic for Deep Learning". In: *arXiv:1603.07285 [cs, stat]* (Jan. 2018).

[19] *Efficient Multi-Scale 3D CNN with Fully Connected CRF for Accurate Brain Lesion Segmentation — Elsevier Enhanced Reader*. https://reader.elsevier.com/reader/sd/ DOI: 10.1016/j.media.2016.10.004.

[20] David Foster. *Generative Deep Learning - Teaching Machines to Paint, Write, Compose and Play*. First Edition. O'Reilly Media, Inc., July 2019.

[21] Maayan Frid-Adar et al. "Synthetic Data Augmentation Using GAN for Improved Liver Lesion Classification". In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. Apr. 2018, pp. 289–293. DOI: 10.1109/ISBI.2018.8363576.

[22] *Gastrointestinal Lesions in Regular Colonoscopy Dataset*. http://www.depeca.uah.es/colonoscopy_dataset/. Accessed: 2019-12-12.

[23] *Gastrointestinalatlas.Com - El Atlas Gastrointestinal*. http://www.gastrointestinalatlas.com/ind

[24] *GASTROLAB - the Gastrointestinal Site*. http://www.gastrolab.net/index.htm. Accessed: 2019-12-12.

[25] Jon Gauthier. "Conditional Generative Adversarial Nets for Convolutional Face Generation". In: (), p. 9.

[26] Jonas Gehring et al. "Convolutional Sequence to Sequence Learning". In: *arXiv:1705.03122 [cs]* (July 2017).

[27] *General Python FAQ — Python 3.8.2 Documentation*. https://docs.python.org/3/faq/general.ht

[28] F. A. Gers, J. Schmidhuber, and F. Cummins. "Learning to Forget: Continual Prediction with LSTM". In: (Jan. 1999), pp. 850–855. DOI: 10.1049/cp:19991218.

[29] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 2672–2680.

[30] Caglar Gulcehre et al. "Learned-Norm Pooling for Deep Feedforward and Recurrent Neural Networks". In: *arXiv:1311.1780 [cs, stat]* (Sept. 2014). arXiv: 1311.1780 [cs, stat].

[31] Matthew Hausknecht and Peter Stone. "Deep Recurrent Q-Learning for Partially Observable Mdps". In: 2015.

[32] Douglas M. Hawkins. "The Problem of Overfitting". In: *Journal of Chemical Information and Computer Sciences* 44.1 (Jan. 2004), pp. 1–12. DOI: 10.1021/ci0342472.

[33] Haibo He and Edwardo A. Garcia. "Learning from Imbalanced Data". In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (Sept. 2009), pp. 1263–1284. ISSN: 1558-2191. DOI: 10.1109/TKDE.2008.239.

[34] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *arXiv:1512.03385 [cs]* (Dec. 2015).

[35] E Hinton. "Learning Internal Representations by Error Propagation". en. In: (), p. 49.

[36] Øyvind Holme et al. "Flexible Sigmoidoscopy versus Faecal Occult Blood Testing for Colorectal Cancer Screening in Asymptomatic Individuals". In: *Cochrane Database of Systematic Reviews* 9 (2013). DOI: 10.1002/14651858.CD009259.pub2.

[37] Alain Horé and Djemel Ziou. "Image Quality Metrics: PSNR vs. SSIM". In: *2010 20th International Conference on Pattern Recognition*. Aug. 2010, pp. 2366–2369. DOI: 10.1109/ICPR.2010.579.

[38] Gao Huang et al. "Densely Connected Convolutional Networks". In: *arXiv:1608.06993 [cs]* (Jan. 2018).

[39] Xun Huang et al. "Multimodal Unsupervised Image-to-Image Translation". In: *arXiv:1804.04732 [cs, stat]* (Aug. 2018).

[40] Q. Huynh-Thu and M. Ghanbari. "Scope of Validity of PSNR in Image/Video Quality Assessment". In: *Electronics Letters* 44.13 (June 2008), pp. 800–801. DOI: 10.1049/el:20080522.

[41] Sae Hwang et al. "Polyp Detection in Colonoscopy Video Using Elliptical Shape Feature". In: *2007 IEEE International Conference on Image Processing*. Vol. 2. Sept. 2007, pp. II - 465-II –468. DOI: 10.1109/ICIP.2007.4379193.

[42] *ImageNet*. http://www.image-net.org/about-overview.

[43] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *arXiv:1502.03167 [cs]* (Mar. 2015).

[44] Phillip Isola et al. "Image-to-Image Translation with Conditional Adversarial Networks". In: *arXiv:1611.07004 [cs]* (Nov. 2018).

[45] Debesh Jha et al. "Kvasir-seg: A segmented polyp dataset". In: *arXiv preprint arXiv:1911.07069* (2019).

[46] Shuiwang Ji et al. "3D Convolutional Neural Networks for Human Action Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.1 (Jan. 2013), pp. 221–231. DOI: 10.1109/TPAMI.2012.59.

[47] Nal Kalchbrenner et al. "Video Pixel Networks". In: *arXiv:1610.00527 [cs]* (Oct. 2016).

[48] Michal F. Kaminski et al. "Increased Rate of Adenoma Detection Associates With Reduced Risk of Colorectal Cancer and Death". In: *Gastroenterology* 153.1 (July 2017), pp. 98–105. DOI: 10.1053/j.gastro.2017.04.006.

[49] Dervis Karaboga and Bahriye Akay. "A Survey: Algorithms Simulating Bee Swarm Intelligence". In: *Artificial Intelligence Review* 31.1 (Oct. 2009), p. 61. DOI: 10.1007/s10462-009-9127-4.

[50] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv:1412.6980 [cs]* (Jan. 2017).

[51] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *arXiv:1312.6114 [cs, stat]* (May 2014).

[52] Anastasios Koulaouzidis et al. "KID Project: an internet-based digital video atlas of capsule endoscopy for research purposes". In: *Endoscopy international open* 5.6 (May 2017), E477–E483. DOI: 10.1055/s-0043-105488.

[53] Yves Lechevallier and Gilbert Saporta, eds. *Proceedings of COMPSTAT'2010*. Heidelberg: Physica-Verlag HD, 2010. DOI: 10.1007/978-3-7908-2604-3.

[54] Alex X. Lee et al. "Stochastic Adversarial Video Prediction". In: *arXiv:1804.01523 [cs]* (Apr. 2018).

[55] Xiaodan Liang et al. "Dual Motion GAN for Future-Flow Embedded Video Prediction". In: *arXiv:1708.00284 [cs]* (Aug. 2017).

[56] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. "Unsupervised Image-to-Image Translation Networks". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 700–708.

[57] William Lotter, Gabriel Kreiman, and David Cox. "Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning". In: *arXiv:1605.08104 [cs, q-bio]* (Feb. 2017).

[58] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. "Rectifier Nonlinearities Improve Neural Network Acoustic Models". In: 30 (2013), p. 3.

[59] Warren S. McCulloch and Walter Pitts. "A Logical Calculus of the Ideas Immanent in Nervous Activity". In: *The bulletin of mathematical biophysics* 5.4 (Dec. 1943), pp. 115–133. DOI: 10.1007/BF02478259.

[60] Luke Metz et al. "Unrolled Generative Adversarial Networks". In: *arXiv:1611.02163 [cs, stat]* (May 2017).

[61] Mehdi Mirza and Simon Osindero. "Conditional Generative Adversarial Nets". In: *arXiv:1411.1784 [cs, stat]* (Nov. 2014).

[62] Volodymyr Mnih et al. "Playing Atari with Deep Reinforcement Learning". In: *arXiv:1312.5602 [cs]* (Dec. 2013).

[63] Marc Oliu, Javier Selva, and Sergio Escalera. "Folded Recurrent Neural Networks for Future Video Prediction". In: *arXiv:1712.00311 [cs, stat]* (Mar. 2018).

[64] *Optical Flow — OpenCV-Python Tutorials 1 Documentation*. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_lucas_kanade/py_lucas_kana

[65] Muhammad Owais et al. "Artificial Intelligence-Based Classification of Multiple Gastrointestinal Diseases Using Endoscopy Videos for Clinical Diagnosis". In: *Journal of Clinical Medicine* 8 (July 2019), p. 986. DOI: 10.3390/jcm8070986.

[66] Sinno Jialin Pan and Qiang Yang. "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (Oct. 2010), pp. 1345–1359. DOI: 10.1109/TKDE.2009.191.

[67] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. *Understanding the Exploding Gradient Problem.* /paper/Understanding-the-exploding-gradient-problem-Pascanu-Mikolov/c5145b1d15fea9340840cc8bb6f0e46e8934827f. 2012.

[68] Viorica Patraucean, Ankur Handa, and Roberto Cipolla. "Spatio-Temporal Video Autoencoder with Differentiable Memory". In: *arXiv:1511.06309 [cs]* (Sept. 2016).

[69] Konstantin Pogorelov et al. "KVASIR: A Multi-Class Image Dataset for Computer Aided Gastrointestinal Disease Detection". In: *Proceedings of ACM MMSYS*. June 2017. DOI: 10.1145/3083187.3083212.

[70] Konstantin Pogorelov et al. "Nerthus: A Bowel Preparation Quality Video Dataset". In: *Proceedings of the ACM Multimedia Systems Conference (MMSYS)*. 2017, pp. 170–174. DOI: 10.1145/3083187.3083216.

[71] Yuchen Pu et al. "Adversarial Symmetric Variational Autoencoder". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 4330–4339.

[72] Prashanth Rawla, Tagore Sunkara, and Adam Barsouk. "Epidemiology of Colorectal Cancer: Incidence, Mortality, Survival, and Risk Factors". In: *Przeglad Gastroenterologiczny* 14.2 (2019), pp. 89–103. DOI: 10.5114/pg.2018.81072.

[73] Ali Sharif Razavian et al. "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition". In: *arXiv:1403.6382 [cs]* (May 2014).

[74] Michael Riegler et al. "EIR — Efficient Computer Aided Diagnosis Framework for Gastrointestinal Endoscopies". In: *2016 14th International Workshop on Content-Based Multimedia Indexing (CBMI)*. June 2016, pp. 1–6. DOI: 10.1109/CBMI.2016.7500257.

[75] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *arXiv:1505.04597 [cs]* (May 2015).

[76] F. Rosenblatt. "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain". In: *Psychological Review* 65.6 (1958), pp. 386–408. DOI: 10.1037/h0042519.

[77] Kevin Roth et al. "Stabilizing Training of Generative Adversarial Networks through Regularization". In: *arXiv:1705.09367 [cs, stat]* (Nov. 2017).

[78] David E Rumelhart, Geoffrey E Hintont, and Ronald J Williams. "Learning Representations by Back-Propagating Errors". In: (1986), p. 4.

[79] Gavin A. Rummery and Mahesan Niranjan. *On-Line Q-Learning Using Connectionist Systems*. Vol. 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[80] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115.3 (Dec. 2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[81] Mark Sandler et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: *arXiv:1801.04381 [cs]* (Mar. 2019).

[82] Rebecca L. Siegel, Kimberly D. Miller, and Ahmedin Jemal. "Cancer Statistics, 2020". In: *CA: A Cancer Journal for Clinicians* n/a.n/a (). DOI: 10.3322/caac.21590.

[83] Juan Silva et al. "Toward embedded detection of polyps in wce images for early diagnosis of colorectal cancer". In: *International Journal of Computer Assisted Radiology and Surgery* 9.2 (2014), pp. 283–293.

[84] David Silver et al. "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm". In: *arXiv:1712.01815 [cs]* (Dec. 2017).

[85] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv:1409.1556 [cs]* (Apr. 2015).

[86] Pia H. Smedsrud et al. *Kvasir-Capsule, a video capsule endoscopy dataset.* June 2020.

[87] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: (), p. 30.

[88] *State of Data Science and Machine Learning 2019.* https://www.kaggle.com/kaggle-survey-2019.

[89] Williams Street. "Colorectal Cancer Facts & Figures 2017-2019". In: (), p. 40.

[90] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *arXiv:1512.00567 [cs]* (Dec. 2015).

[91] Nima Tajbakhsh, Suryakanth R Gurudu, and Jianming Liang. "Automated Polyp Detection in Colonoscopy Videos Using Shape and Context Information". In: *IEEE Transactions on Medical Imaging* 35.2 (2016), pp. 630–644. DOI: 10.1109/TMI.2015.2487997.

[92] Hirotoshi Takiyama et al. "Automatic Anatomical Classification of Esophagogastroduodenoscopy Images Using Deep Convolutional Neural Networks". In: *Scientific Reports* 8.1 (Dec. 2018), p. 7497. DOI: 10.1038/s41598-018-25842-6.

[93] Keras Team. *Keras Documentation: Keras Applications.* https://keras.io/api/applications/.

[94] *The Atlas of Gastrointestinal Endoscope.* http://www.endoatlas.com/atlas_1.html. Accessed: 2019-12-12.

[95] Du Tran et al. "Learning Spatiotemporal Features with 3D Convolutional Networks". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. Santiago, Chile: IEEE, Dec. 2015, pp. 4489–4497. DOI: 10.1109/ICCV.2015.510.

[96] Sergey Tulyakov et al. "MoCoGAN: Decomposing Motion and Content for Video Generation". In: *arXiv:1707.04993 [cs]* (Dec. 2017).

[97] Srinivas C. Turaga et al. "Convolutional Networks Can Learn to Generate Affinity Graphs for Image Segmentation". In: *Neural Computation* 22.2 (Feb. 2010), pp. 511–538. DOI: 10.1162/neco.2009.10-08-881.

[98] *Understanding 1D and 3D Convolution Neural Network — Keras*. https://towardsdatascience.com/understar
1d-and-3d-convolution-neural-network-keras-9d8f76e29610.

[99] Gregor Urban et al. "Deep Learning Localizes and Identifies Polyps in Real Time With 96% Accuracy in Screening Colonoscopy". In: *Gastroenterology* 155.4 (Oct. 2018), 1069–1078.e8. DOI: 10.1053/j.gastro.2018.06.037.

[100] Jeroen C. van Rijn et al. "Polyp Miss Rate Determined by Tandem Colonoscopy: A Systematic Review". In: 101 (Feb. 2006), pp. 343–350. DOI: 10.1111/j.1572-0241.2006.00390.x.

[101] Nathan Wan et al. "Machine Learning Enables Detection of Early-Stage Colorectal Cancer by Whole-Genome Sequencing of Plasma Cell-Free DNA". In: *BMC Cancer* 19.1 (Aug. 2019), p. 832. DOI: 10.1186/s12885-019-6003-8.

[102] Ting-Chun Wang et al. "Video-to-Video Synthesis". In: *arXiv:1808.06601 [cs]* (Dec. 2018).

[103] Xiaolong Wang and Abhinav Gupta. "Generative Image Modeling Using Style and Structure Adversarial Networks". In: *arXiv:1603.05631 [cs]* (July 2016).

[104] Yi Wang et al. "Part-Based Multiderivative Edge Cross-Sectional Profiles for Polyp Detection in Colonoscopy". In: *IEEE Journal of Biomedical and Health Informatics* 18.4 (July 2014), pp. 1379–1389. DOI: 10.1109/JBHI.2013.2285230.

[105] Yi Wang et al. "Polyp-Alert: Near Real-Time Feedback during Colonoscopy". In: *Computer Methods and Programs in Biomedicine* 120.3 (July 2015), pp. 164–179. DOI: 10.1016/j.cmpb.2015.04.002.

[106] Z. Wang et al. "Image Quality Assessment: From Error Visibility to Structural Similarity". In: *IEEE Transactions on Image Processing* 13.4 (Apr. 2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[107] Christopher J. C. H. Watkins and Peter Dayan. "Q-Learning". In: *Machine Learning* 8.3 (May 1992), pp. 279–292. DOI: 10.1007/BF00992698.

[108] *WEO Clinical Endoscopy Atlas*. http://www.endoatlas.org/index.php. Accessed: 2019-12-12.

[109] *Why TensorFlow*. https://www.tensorflow.org/about.

[110] Svante Wold, Kim Esbensen, and Paul Geladi. "Principal Component Analysis". en. In: (), p. 16.

[111] Jason Yosinski et al. "How Transferable Are Features in Deep Neural Networks?" In: *arXiv:1411.1792 [cs]* (Nov. 2014).

[112] Xingzhi Yue, Neofytos Dimitriou, and Ognjen Arandjelovic. "Colorectal Cancer Outcome Prediction from H&E Whole Slide Images Using Machine Learning and Automatically Inferred Phenotype Profiles". In: *arXiv:1902.03582 [cs, eess]* (Feb. 2019).

[113] Matthew D. Zeiler et al. "Deconvolutional Networks". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. June 2010, pp. 2528–2535. DOI: 10.1109/CVPR.2010.5539957.

[114] Tao Zhang, Peipei Jiang, and Meng Zhang. "Inter-Frame Video Image Generation Based on Spatial Continuity Generative Adversarial Networks". In: *Signal, Image and Video Processing* 13.8 (Nov. 2019), pp. 1487–1494. DOI: 10.1007/s11760-019-01499-0.

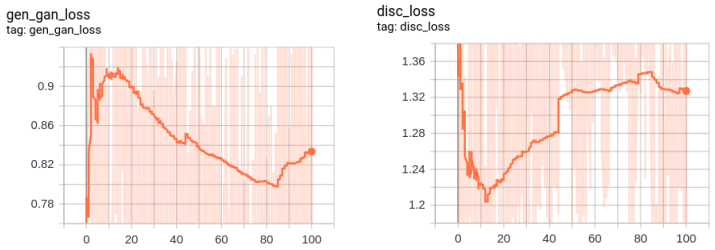
[115] Barret Zoph et al. "Learning Transferable Architectures for Scalable Image Recognition". In: *arXiv:1707.07012 [cs, stat]* (Apr. 2018).

Appendices

Appendix A

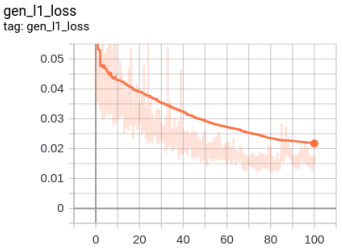
Loss Graphs

The graphs visualizes loss during training of the generative model experiments in our thesis.

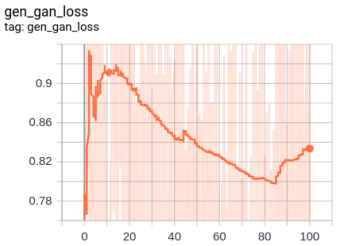


(a) Discriminator Loss

(b) Generator Loss

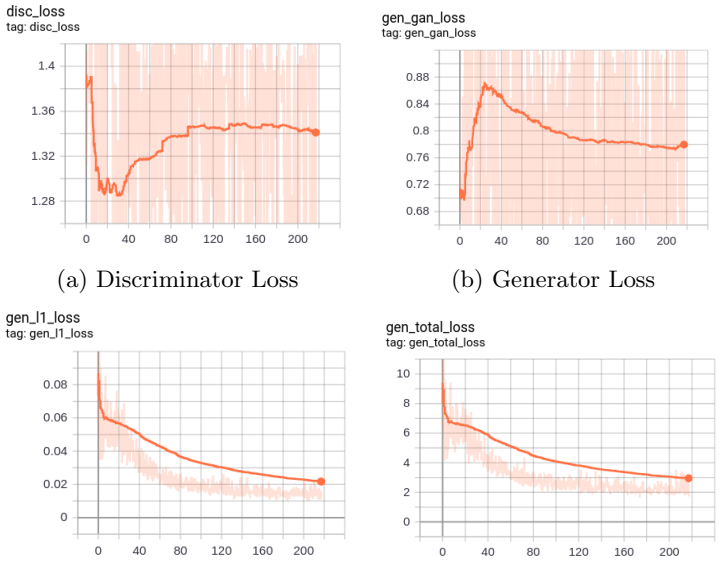


(c) L1 Loss



(d) Total Generator Loss

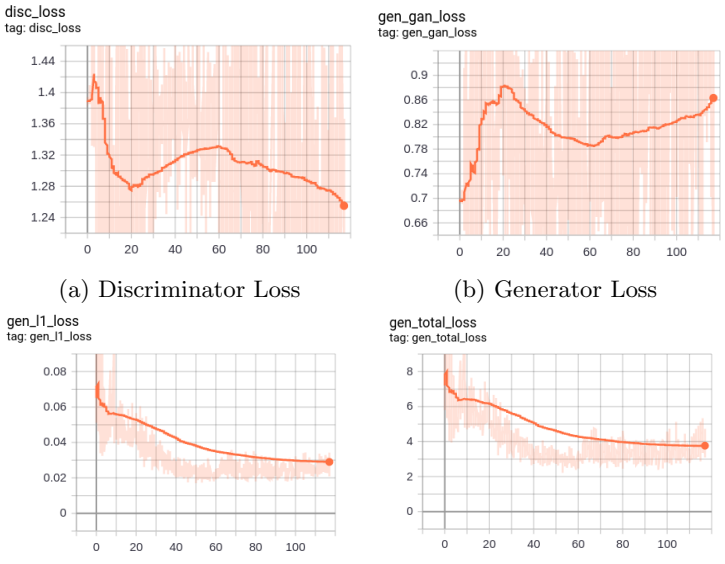
Figure A.1: The figure shows loss graphs for the Original Pix2Pix experiment using early stopping



(a) Discriminator Loss (b) Generator Loss

(c) L1 Loss (d) Total Generator Loss

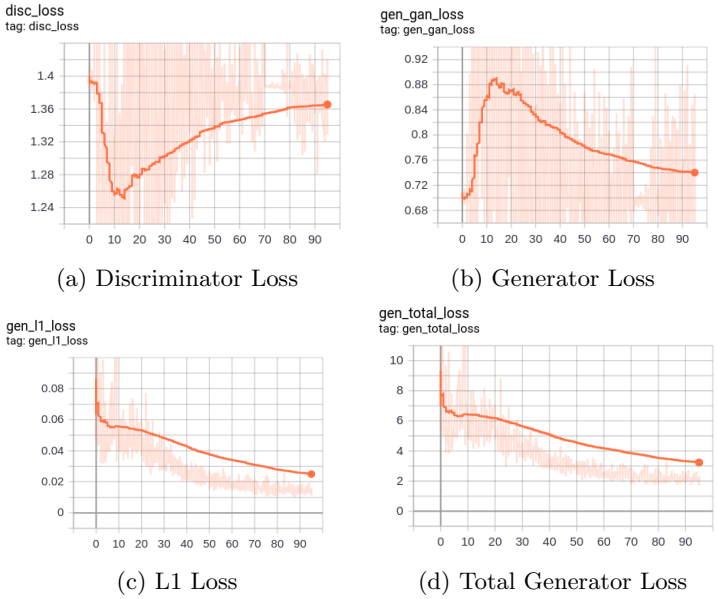
Figure A.2: The figure shows loss graphs for the "Replace with 3D layers" experiment using early stopping



(a) Discriminator Loss (b) Generator Loss

(c) L1 Loss (d) Total Generator Loss

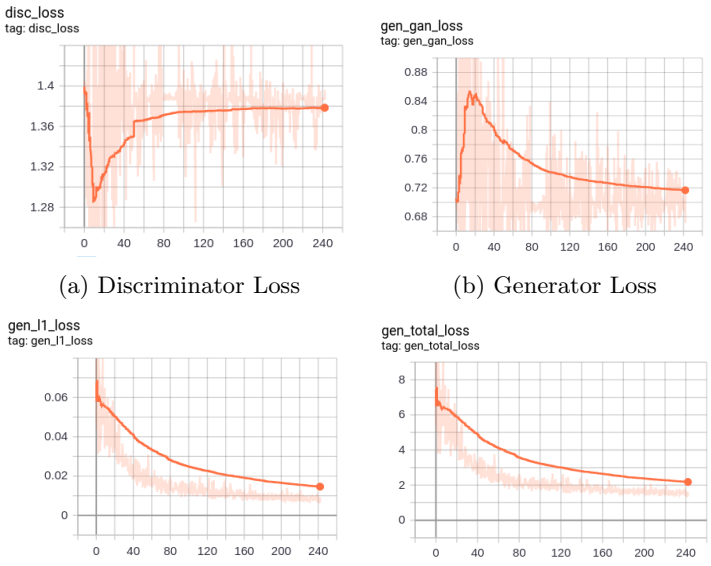
Figure A.3: The figure shows loss graphs for the "change filter size" experiment using early stopping



(a) Discriminator Loss (b) Generator Loss

(c) L1 Loss (d) Total Generator Loss

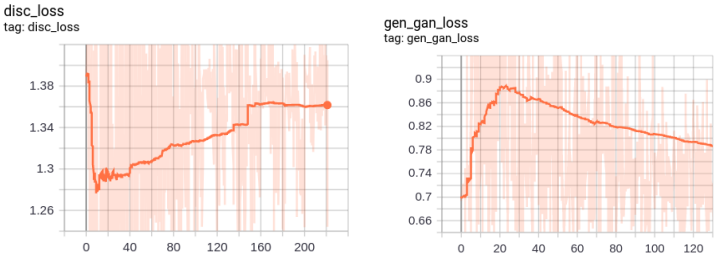
Figure A.4: The figure shows loss graphs for "offset downsampling" experiment using early stopping



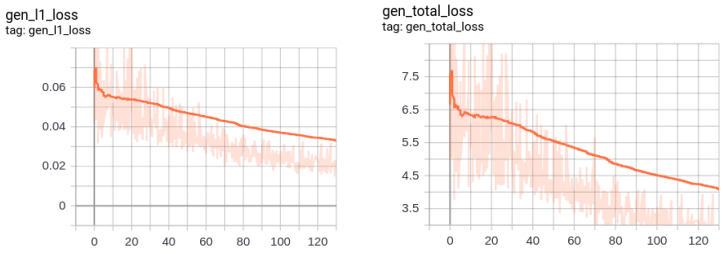
(a) Discriminator Loss (b) Generator Loss

(c) L1 Loss (d) Total Generator Loss

Figure A.5: The figure shows loss graphs for the "reduce discriminator complexity" experiment using early stopping

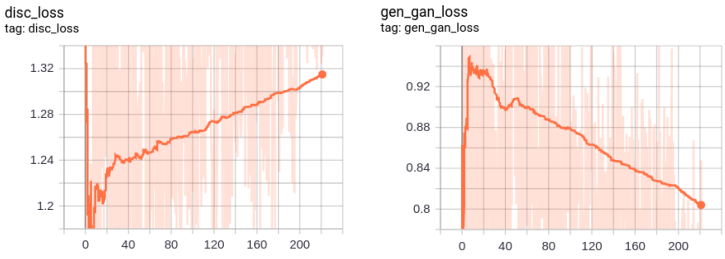


(a) Discriminator Loss (b) Generator Loss



(c) L1 Loss (d) Total Generator Loss

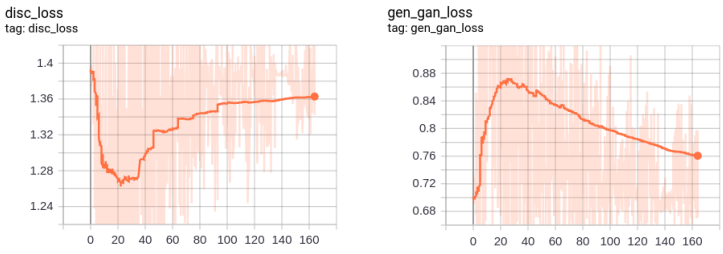
Figure A.6: The figure shows loss graphs for the "keep more features" experiment using early stopping



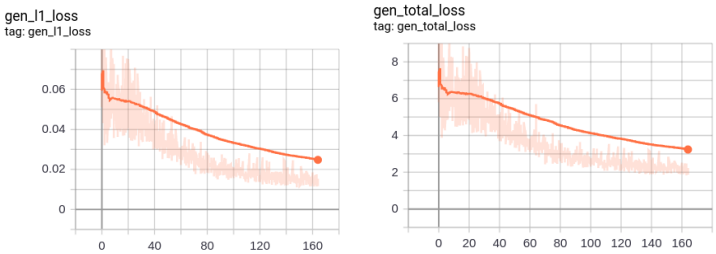
(a) Discriminator Loss (b) Generator Loss

(c) L1 Loss (d) Total Generator Loss

Figure A.7: The figure shows loss graphs for the "add noise" experiment using early stopping



(a) Discriminator Loss (b) Generator Loss



(c) L1 Loss (d) Total Generator Loss

Figure A.8: The figure shows loss graphs for the "2D output" experiment using early stopping

Appendix B

Model Architecture

Generator model and discriminator model architectures for our final solution.

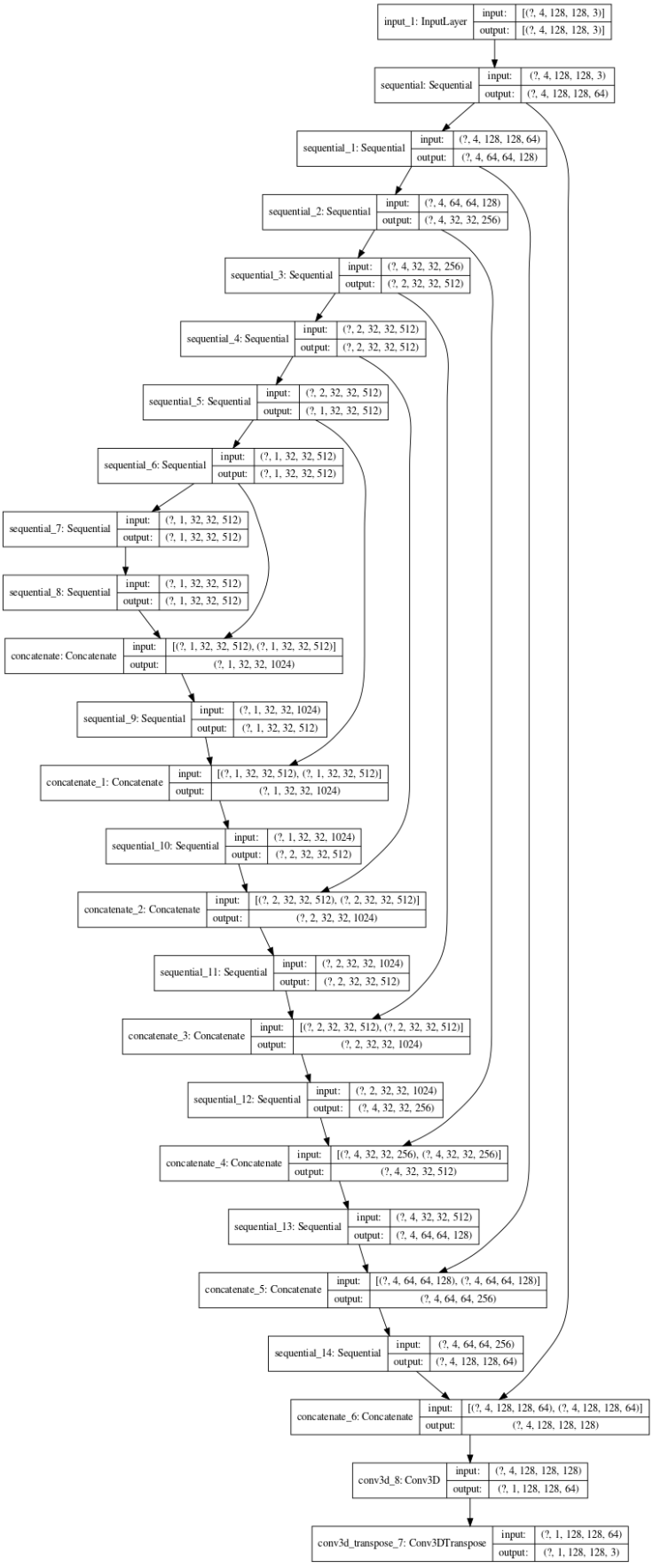


Figure B.1: Generator Model Architecture

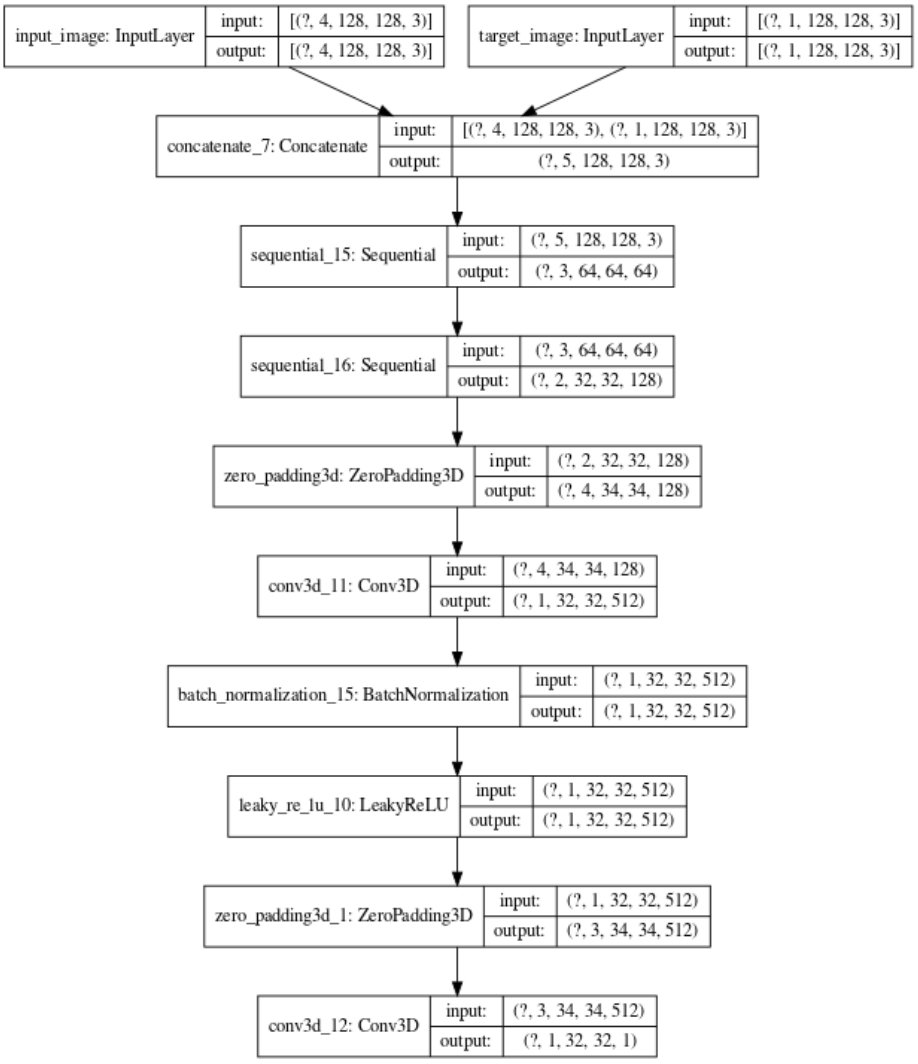


Figure B.2: Discriminator Model Architecture