

Pyramidal Segmentation of Medical Images via Generative Adversarial Networks

Espen Næss
Master's Thesis Spring 2020



Abstract

Colorectal cancer accounts for 10% of all cancer cases. Early detection is crucial for survival and is obtained by regular screening of the gastrointestinal tract for precursors of gastrointestinal cancer, known as polyps. Research has shown polyp miss rates of 14% to 30% for manual classification performed by doctors. Similar problems related to human error are observed when determining other attributes, such as borders and size of findings, which motivates the use of automated segmentation. Segmentation is the process of partitioning an image into areas with specified descriptions, meaning every pixel in the image is classified to detect and locate findings. In recent years, machine learning has provided impressive results for a wide variety of fields, ranging from language translation to facial recognition and cancer detection. The focus of this thesis will be to develop new segmentation models based on recent advances in machine learning and our hypothesis that learning several degrees of segmentation precisions by segmenting within grids may aid segmentation performance. This idea was motivated by the hypothesis that segmentation performance could be improved by building upon the knowledge of performing less precise segmentations. Our results suggest that segmentations of lower precisions produce better results at the cost of less precision, which proved useful for the cases where higher precision segmentations gave limited results. However, no impact on segmentation performance of higher segmentation precisions was observed. Generally, the normal pixel-level segmentation performance of our networks was as good as experiments with corresponding state-of-the-art neural networks for segmentation.

Acknowledgments

I would like to thank the people at Simula and Augere Medical for providing help in addition to a very friendly and inviting atmosphere. In particular, I would like to thank my official supervisors

Michael, Pål and Håkon as well as Steven and Vajira. I am deeply grateful for all the good advices, friendly conversations, feedback and guidance you guys have given me. I feel very lucky to have had you all as my supervisors. I would especially like to thank you all for the help during the last months of writing when offices/universities were closing down due to corona and I had to work from home. This thesis really would not be possible without you guys!

I would also like to thank Oda and Henrik, whom I shared an office with during the writing of this thesis prior to corona. You guys made this year so much more fun and it really wouldn't be the same without you guys.

Finally, I would like to thank my father Geir and my twin sister Camilla for their support and encouragement. Especially my dad for often providing me with dinners during the long days of writing and performing experiments.

Contents

1	Introduction	1
1.1	Background and Motivation	1
1.2	Problem Statement	2
1.3	Limitations	3
1.4	Research Methods	4
1.4.1	Theory	4
1.4.2	Abstraction	4
1.4.3	Design	4
1.5	Main Contributions	5
1.6	Thesis Outline	6
2	Background and Related Works	8
2.1	Medical Background	8
2.1.1	The Gastrointestinal Tract	9
2.1.1.1	Diseases in the Gastrointestinal Tract	10
2.1.1.1.1	Polyps	10
2.1.1.1.2	Ulcerative Colitis	11
2.1.1.2	Anatomical Landmarks in the Gastrointestinal Tract	11
2.1.1.2.1	Z-line	11
2.1.1.2.2	Pylorus	12
2.1.1.2.3	Cecum	13
2.1.2	Screening Methods of the Gastrointestinal Tract	13
2.1.2.1	Gastrointestinal Endoscopy	13
2.1.2.2	Wireless Capsule Endoscopy	15

2.2	Machine Learning Background	16
2.2.1	Types of Machine Learning	16
2.2.1.1	Supervised Learning	16
2.2.1.2	Unsupervised Learning	17
2.2.1.3	Reinforcement Learning	17
2.2.2	Neural Networks	18
2.2.2.1	Artificial Neurons	18
2.2.2.2	Network Layout	19
2.2.2.3	Training	20
2.2.3	Neural Networks for Image Applications	20
2.2.3.1	Convolutional Neural Networks	21
2.2.3.1.1	Convolutional Layers	21
2.2.3.1.2	Pooling Layers	23
2.2.3.1.3	Network Layout	23
2.2.3.1.4	U-NET Architecture	24
2.2.3.2	Generative Adversarial Networks	25
2.2.3.2.1	Conditional GANs	27
2.2.3.2.2	Pix2Pix	27
2.2.3.2.3	CycleGAN	28
2.2.4	Neural Networks for Video Applications	29
2.2.4.1	Recurrent Neural Networks	30
2.2.4.2	Long Short-Term Memory RNN	30
2.2.4.3	RNN-architectures for Video Classification	31
2.2.4.4	Feature Extraction of the Video Frames	32
2.3	Summary	33
3	Grid Segmentation Networks	34
3.1	Grid Segmentation Framework	34
3.1.1	Ground Truth	35
3.1.2	Grid Encoding	36
3.1.3	Dataset Structure	37
3.1.4	Grid Segmentation Mapping	38
3.1.5	Pipeline Summary	39

3.2	Grid U-NET	39
3.2.1	Motivation	40
3.3	Grid-GAN	41
3.3.1	Motivation	42
3.4	Summary	43
4	Methods and Experiments	44
4.1	Data Material	44
4.1.1	Datasets	44
4.1.1.1	Kvasir-Capsule Dataset	45
4.1.1.2	Kvasir-SEG Dataset	45
4.1.1.3	CVCClinicDB Dataset	45
4.1.2	Issue of Data	45
4.1.3	Data Enhancement and Augmentation	46
4.1.3.1	Rotation Augmentation	46
4.1.3.2	Contrast Enhancement	47
4.2	Experiment Setup	48
4.2.1	K-fold Cross Validation	49
4.2.1.1	Cross Validation Setup for Training on Grids	49
4.2.2	Metrics	51
4.2.2.1	Intersection over Union (IoU)	51
4.2.2.2	Dice Coefficient	52
4.2.3	Basic Architecture	53
4.2.3.1	Differences from Architecture in Original Paper	54
4.2.4	System Specifications	54
4.3	GAN Experiments	55
4.3.1	Pix2Pix	55
4.3.2	Grid-GAN without Data Augmentation/Enhancement	56
4.3.3	Grid-GAN with Rotation Augmented Data	59
4.3.4	Grid-GAN with Contrast Enhanced Data	61
4.3.5	Grid-GAN with All Grid Sizes and Rotation Augmented Data	62
4.3.6	Grid-GAN with Loss Modifications	64
4.3.7	Grid-GAN with Upsampling Skip Connections	67

4.3.8	Final Small Scale Experiments with 256x256 Images	71
4.4	U-NET Baseline Experiments	72
4.4.1	U-NET	72
4.4.2	Grid U-NET with Rotation Augmented Data	73
4.4.3	Grid U-NET with Contrast Enhanced Data	75
4.4.4	Grid U-NET with Loss Modifications	76
4.4.5	Grid U-NET with Upsampling Skip Connections	77
4.5	Summary	78
5	Conclusion	80
5.1	Summary and Contributions	80
5.2	Future Work	83

List of Figures

2.1	Illustration of the gastrointestinal tract ¹	9
2.2	Example of a polyp, taken from the Kvasir-SEG dataset [43].	10
2.3	Case of ulcerative colitis with ulceration of the mucosa, as well as bleeding and swelling. Image taken from the Kvasir Dataset [63].	11
2.4	Image of a Z-line, taken from the Kvasir dataset [63]. The image clearly shows the transition from the white mucosa of the esophagus to the red mucosa of the stomach.	12
2.5	Image of a pylorus taken from the Kvasir dataset [63], which connects the stomach to the duodenum.	12
2.6	Image of a cecum taken from the Kvasir dataset [63], which is considered to be the beginning of the large intestine.	13
2.7	Illustration of how conventional endoscopy procedures are performed, either as gastroscopy or colonoscopy procedures ²	14
2.8	A typical example of a camera capsule used for WCE ³ . The data we worked with was gathered by an Olympus EC-S10 endocapsule.	15
2.9	Illustration of an artificial neuron ⁴ with three inputs X_1, X_2, X_3 that are summed and weighted with corresponding weights W_1, W_2, W_3 and passed to an activation function f to produce an output Y . This is a typical layout of a neuron for Multilayer Perceptrons (MLPs).	19
2.10	Example of a neural network ⁵ called a feed forward neural network. Neurons are arranged layerwise where each layer incorporates increasingly more advanced features the deeper the layer is situated.	19
2.11	The output of a convolutional layer ⁶ can be interpreted as a volume where each slice (called a feature map) corresponds to the result of a convolution of the input with a filter kernel of learnable weights that are learned during backpropagation and training of the convolutional neural network.	22
2.12	Example of max pooling ⁷ of a two dimensional matrix using a stride and pool size of 2.	23
2.13	Convolutional Neural Network architecture for image classification ⁸ . It is divided into two parts - one for feature learning and one for classification. The feature extraction part constitutes a stacking of pooling and convolutional layers while the classification part is a fully connected layer.	24

2.14	Original U-NET architecture from the U-NET paper ⁹ . This thesis will use a modified version of this architecture.	25
2.15	Illustration of a LSTM-cell ¹⁰ . The input gate controls to what degree a new value will flow into the cell, while the output gate controls to what degree the value stored in the LSTM cell is used to compute the output of the LSTM-cell. The forget gate determines to which extent the value in the memory of the cell should be forgotten.	31
2.16	Illustration of possible RNN architectures ¹¹ . A many-to-one architecture gives a classification of the whole video to one single category while a many-to-many architecture gives a classification of every single frame [90]. The last layer for the architectures is typically a fully connected layer that gives the final classification.	31
2.17	Illustration of how CNNs could be used for feature extraction to RNNs [20]. The weights of the LSTM and CNN are both shared over time and makes the model able to process videos of arbitrary length [20]. Each frame is fed to a Convolutional Neural Network (CNN) with the goal of obtaining a feature extraction for each of the frames. These feature extractions are then given as input to the RNN, which classifies each frame based on the contextual relationship from both previous and current frame.	32
3.1	An image of size 128x128 and the corresponding ground truth segmentations for all the different grid sizes. Black indicates no finding while grey indicates a finding. As the size of the input image is 128x128, the ground truth for a 128x128 grid size is equivalent to the normal segmentation ground truth mask.	35
3.2	Grid encodings for different grid sizes. The grid encoding for a particular grid size is given by a checkerboard pattern. This checkerboard can essentially be characterized in its very nature as a grid and is how the grid size is encoded. For higher grid sizes, the grid encoding pattern gets difficult to see due to a dithering effect of the white and black squares, which makes the grid encodings appear grey at a distance.	36
3.3	Illustration of how the datasets are structured. Grey indicates a finding while black indicates no finding. For each image in each of the training, validation and test sets, there is now a set of 3-tuples that correspond to segmentations of the image within a grid specified by the grid encoding.	37
3.4	The goal is to learn a segmentation mapping that takes a given image and a grid size (encoded as an image) and segments the image within a grid size specified by the grid encoding. In the segmented image, grey indicates a finding while black does not indicate a finding. The grid encoding for a normal pixel-level segmentation (128x128 grid size) is cluttered to a grey color due to dithering, but follows the same chess pattern as the above encodings.	38
3.5	Summary of the pipeline, corresponding to how our grid segmentation framework is applied to our neural networks. We augment each training, validation and test datasets with grid information. For each image, grid encodings and corresponding grid segmentations are generated. These are used to train, select and evaluate the given model.	39

3.6	Basic illustration of how simple Grid U-NETs work, which essentially is the grid segmentation framework applied to a U-NET architecture. The U-NET architecture is illustrated by boxes that represent layers and stapled lines representing skip connections.	39
3.7	Illustration of receptive fields [51]. The yellow pixel value in layer 3 covers a 5x5 area marked as yellow in layer 1. Traditionally in CNN architectures, the deeper a layer is situated, the larger the receptive field of the given features for that particular layer. This means that the values in the deeper layers represent features from a larger area of the input.	40
3.8	Example of the objectives for a basic Grid-GAN. This illustrates the objective of the discriminator of the Grid-GAN, which is the opposite of the objective to the generator. As shown in the illustration, the discriminator seeks to classify the ground truth as real and generated images as fake. The objective for the generator is the opposite, where it seeks to trick the discriminator into classifying generated images as real.	41
4.1	Example of contrast enhancement of an image from the CVCClinicDB dataset by using the CLAHE-algorithm. This increases the visibility of the polyp in the center of the image. Blood veins are also more visible, which can be crucial information for the classification of polyps, as the polyp often is characterized by its blood supply.	48
4.2	The basic idea of a 5-fold cross validation ¹² . There are 5 different partitions, all of which are a part of the training set and validation set at least once. For each iteration, a model is trained and evaluated. The average of the metrics gained from the different models provides an estimate of the over all segmentation performance of the model based on the data.	49
4.3	Example of how cross validation in the presence of grids have been handled in this thesis. The example is given using 2 folds and grid sizes of 2, 4, 8, 16 and 128. A cross validation is performed for each of the grid datasets and yields two iterations, where each iteration constitutes a training and validation dataset. All grid datasets corresponding to the same iterations are then concatenated such that there is one single training and validation set for each iteration. Each concatenated training set is then trained on and evaluated to produce metrics which are then averaged over to produce an estimate of the performance of the model based on the data.	50
4.4	Formula for the Intersection over Union (IoU), illustrated by the use of bounding boxes ¹³ . One box corresponds to the ground truth while the other is the prediction. A bounding box indicates the pixels which are marked as a finding. The IoU is the degree of overlap between predicted pixels marked as a finding and ground truth pixels marked as a finding divided by the total area of all these pixels.	51
4.5	Example ¹⁴ of how Intersection over Union (IoU) yields different results based on the overlap of the prediction and the ground truth, illustrated by the use of bounding boxes. The values range from 0 to 1, where values around 0.4 indicates poor segmentation performance, values of 0.73 indicates good performance and values of 0.92 indicates excellent performance.	52

4.6	Formula for the Dice coefficient, given by the use of bounding boxes, as previous illustrations ¹⁵ . Like IoU, it has a range of 0 to 1 where 1 indicates best possible performance.	53
4.7	System specifications for the machine that has been running all of our experiments.	54
5.1	Segmentation examples with grid size 32 and 64 for the Grid-GAN variant where each grid cell is a pixel value. The black color in the output indicates that it is outside of the segmentation. Gray is no finding. White is a finding.	84

List of Tables

4.1	2-fold cross validation performed on a Pix2Pix model (no grids), using the same parameters as in the paper of Pix2Pix, though with a new and stable architecture. No data enhancement was performed for this experiment.	56
4.2	2-fold cross validation for Pix2Pix model with optimized parameters and rotation data augmentation. No grid information was used.	56
4.3	2-fold cross validation results for Grid-GAN using default Pix2Pix parameters without any data enhancement besides grids. For this particular model, these parameters gave the best results. Grid segmentation for grid size 128 is a normal pixel-level segmentation, as the image input size is 128.	57
4.4	2-fold cross validation results for Grid-GAN without any data augmentation. This experiment performed worse than the previous. It ran with 219 epochs, a learning rate of 0.003 for both the generator and discriminator, a batch size of 16 and the same adam optimizers as previous runs.	58
4.5	Segmentation performance of our cross validated model in Table 4.6 for randomly selected images from our test dataset (the CVCClinicDB dataset). Gray indicates a finding, while black indicates no finding.	59
4.6	2-fold cross validation of Grid-GAN ran with rotation augmentation using default Pix2Pix parameters. This run is equivalent to the run in Table 4.3, but with rotation augmentation. As observed for the previous model, these parameters performed the best.	60
4.7	2-fold cross validation of Grid-GAN with rotation data augmentation, ran with 164 epochs, a learning rate of 0.004 for the generator and discriminator and a batch size of 16. This experiment performed worse than our experiment with default Pix2Pix parameters.	61
4.8	2-fold cross validation of Grid-GAN performed on contrast enhanced data. Ran with 224 epochs, a learning rate of 0.001 for generator and discriminator and a batch size of 16.	62
4.9	2-fold cross validation of Grid-GAN ran for all grids using rotation augmentation. The experiment was performed using a learning rate of 0.004 for both the generator and discriminator, 170 epochs and a batch size of 16.	63

4.10 Single run for Grid-GAN with a mean cross entropy grid loss with weight parameter $\lambda_2 = 10$. Kvasir-SEG for generating the training and validation sets and CVCClinicDB for generating the test set, as with all other experiments. Rotation augmented data used.	65
4.11 Performance of our cross validated model in Table 4.12 for randomly selected images from our test dataset (the CVCClinicDB dataset). Gray indicates a finding, while black indicates no finding.	65
4.12 2-fold cross validation of Grid-GAN with a mean cross entropy grid loss and rotation augmentation. Performed with a learning rate of 0.0001 and 73 epochs together with a batch size of 4. This was our best performing experiment with the mean cross entropy grid loss.	66
4.13 2-fold cross validation of Grid-GAN with a mean cross entropy grid loss and rotation augmentation. Performed with a grid loss weight of $\lambda_2 = 10$, a learning rate of 0.002 for discriminator and generator, 195 epochs and a batch size of 16. This experiment gave worse results than our previous experiment.	67
4.14 Performance of our cross validated model in Table 4.15 for randomly selected images from our test dataset (the CVCClinicDB dataset). Gray indicates a finding, while black indicates no finding.	68
4.15 2-fold cross validation of Grid-GAN with upsampling skip connections. Ran using rotation augmentation and the same parameters as default Pix2Pix with a learning rate of 0.0002, 82 epochs and a batch size of 4.	69
4.16 2-fold cross validation of Grid-GAN with upsampling skip connections, ran with 131 epochs, 0.008 in learning rate for generator and discriminator, a batch size of 16 and rotation augmented data. This experiment performed marginally worse than our previous.	70
4.17 Single Pix2Pix experiment on 256x256 images using a learning rate of 0.002, batch size of 16 and 512 epochs.	71
4.18 2-fold cross validation of ordinary U-NET without any grids, but with rotation augmented data. Performed using the same parameters from the Kvasir-SEG dataset paper with a learning rate of 0.0001, 120 epochs and a batch size of 16.	72
4.19 2-fold cross validation of Grid U-NET ran with rotation augmentation. Performed with a learning rate of 0.0004, 98 epochs and a batch size of 4.	73
4.20 2-fold cross validation of Grid U-NET ran with rotation augmentation. Performed with a learning rate of 0.003, 226 epochs and a batch size of 16.	74
4.21 2-fold cross validation of Grid U-NET ran on contrast enhanced images, A learning rate of 0.002 was used as well as a batch size of 16. The experiment was run with 283 epochs.	75
4.22 2-fold cross validation of Grid U-NET with a mean cross entropy grid loss. Performed using a learning rate of 0.0002, a batch size of 4 and 85 epochs.	76
4.23 2-fold cross validation of Grid U-NET with upsampling skip connections performed with a learning rate of 0.0002, batch size of 4 and 131 epochs.	77

4.24 Average test performance results from cross validation experiments of our best varieties of Grid-GANs, ran on 128x128 images. The test dataset was obtained from a separate dataset called CVCClinicDB, while training was performed with the Kvasir-SEG dataset.	78
4.25 Average test performance results from cross validation experiments of our best varieties of Grid U-NETs, ran on 128x128 images. The test dataset was obtained from a separate dataset called CVCClinicDB.	79

Chapter 1

Introduction

1.1 Background and Motivation

Colorectal cancer is a major cause of cancer-related mortality across the world and accounts for around 700 000 deaths annually [8, 25]. A common scenario for those who die from colorectal cancer is few sign of symptoms until the cancer has progressed beyond recovery, which has made early detection important for survival. Colorectal cancer is highly treatable when found early with a 5-year relative survival rate at about 90% [19]. Statistics show that 15% of men and women above the age of 50 are especially at risk of developing colon cancer. This part of the population should ideally be screened every 3 to 5 years [33] in order to detect potential cancer at an early stage and further increase the chance of survival. However, a mass screening of such a scale puts a huge strain on health care systems. As of 2018, no health care system can offer organized population-wide colonoscopy screening of this scale [8].

One of the main challenges of a population-wide mass screening is the processing of data from the colorectal screenings. Depending on the type of colonoscopy performed, a video from one single screening can be up to 8 hours long [60], making the video analysis very time consuming and tedious to do manually. In addition, the video analysis is not trivial and has to be done by medical specialists [4]. The analysis of the video data is also often subject to various kinds of human errors [49], with studies showing miss rates for manual classification as high as 14% to 30% for certain findings [66]. All of these challenges of processing the video data could be solved by some means of automated analysis.

Machine learning has been successfully applied to several medical applications, including the automated detection of precursors to gastrointestinal cancer [2, 38, 64, 79], also commonly known as polyps. This allows doctors to be notified about the presence of polyps in a given frame, but it does not mark the borders or the location of the polyp. Locating borders is important to be able to estimate the size and furthermore the chance of the polyp becoming cancerous, as larger polyps are more likely to develop to cancer. Furthermore, the need for locating the borders of polyps is even more crucial for polyps that are subject to removal, as a poor estimate of the polyp location might lead to a partial removal of the polyp where the polyp might continue to develop to become cancerous at a later stage. Providing an estimate of the borders of polyps can be difficult to do

manually and is subject to human error. It should ideally be automated along with detection, which could be possible with the advancement of automated segmentation techniques.

Segmentation aims to both detect and locate the borders of findings, which is a significantly more challenging task than detection. Every individual pixel in the image is classified as pertaining to a certain region, effectively partitioning the image into regions. This differs from detection which would indicate the presence of a finding by classifying all pixels in the entire image as a whole. Several machine learning algorithms have provided state-of-the-art results for medical image segmentation [5, 59, 67, 82], which motivates further research of machine learning based segmentation approaches. The development of novel image segmentation networks could address our medical problem and would be interesting to compare with current state-of-the-art solutions for medical segmentation.

1.2 Problem Statement

Based on the background and motivation in the previous section, where a main challenge is the processing of gastrointestinal data, we decided to look into improving the performance of the automated analysis of gastrointestinal data gathered from endoscopy examinations. We see the importance of investigating new approaches to improve the performance of analyzing these data, which has the potential to make mass screening more feasible and screening more precise, and by extension, save many of the lives that are today lost to colorectal cancer. Mass screening would allow for the detection of colorectal cancer at an earlier stage for many patients, which as mentioned in the previous section, could dramatically increase survival rates.

The question this thesis is trying to answer can be summarized in the following:

Can we improve the segmentation performance of a segmentation model by learning to segment within several degrees of segmentation precisions?

Our research question is motivated by our hypothesis that features used to create less precise segmentations could aid the training of pixel-level segmentations. Furthermore, different degrees of segmentation precisions can be obtained by learning to segment within grids, while the grid size can be specified as an additional input to the segmentation model and act as a parameter for controlling the segmentation precision. This is the core concept of our models, which will be used to answer our research question. To fully answer our research question, we developed three objectives to help guide us to a solution. Each objective builds on the one that came before it, and through the completion of each objective, we are better suited to make a final conclusion. The three objectives are described below.

- **Objective 1:** Research and develop a framework for learning a grid segmentation mapping that takes an image and a grid size and segments the image within the specified grid size. The segmentation mapping should also be able to perform a normal segmentation in addition to these grid segmentations, which would correspond to a grid size equal to the image size. The resulting framework should be able to work with several state-of-the-art neural networks.

- **Objective 2:** Apply the grid segmentation framework to state-of-the-art neural networks for segmentation. Investigate how the amount and quality of the data affects the learning of the grid segmentations. Compare the results with corresponding state-of-the-art neural networks without grids.
- **Objective 3:** Investigate how the learning of these networks can be fully adapted and modified to the grid segmentation framework in order to enhance segmentation performance. Again, compare these results with corresponding state-of-the-art neural networks without grids.

1.3 Limitations

We will use the medical scenario of detecting and localizing polyps as a case study for our defined research question, which will act as a representative area of application for the methods we develop. Despite the fact that our thesis is limited to this medical scenario, we acknowledge that the methods investigated over the course of this thesis will generally be applicable for several fields where machine learning can be applied.

Most of our work is limited to three datasets; Kvasir-Capsule [76], Kvasir-SEG [43] and CVC-ClinicDB [21]. The Kvasir-Capsule dataset consists of bounding box video annotations of capsule endoscopy videos. It contains 13 different classes of findings and was developed and labeled by us in cooperation with Simula. As for our research question and our grid based segmentation networks, there is a possibility that using the Kvasir-Capsule for our datasets would make the learning of the grids harder due to the bounding boxes. A small overlap of the bounding box to a nearby grid cell would effectively mark that grid cell as containing a finding, even for the case where the overlapping part of the bounding box does not contain a finding, which could make learning more intricate. Since our main goal is to develop a proof of concept for our grid based models and the bounding-box nature of the Kvasir-Capsule could make our approach more difficult to realize, we left the Kvasir-Capsule dataset for future work.

The two datasets that ultimately were subject to our experiments were Kvasir-SEG and CVC-ClinicDB, both of which contain pixelwise annotations of polyps from colonoscopy examinations. Our research was limited to polyps due to two primary constraints. Firstly, the two datasets did not contain any other classes than polyps. The second and most important constraint is the general lack of publicly available medical datasets, especially high quality datasets, which makes it difficult to be picky about datasets and the various diseases they cover. In addition, none of the datasets we worked with were applicable for sequential learning, which is why we decided to limit the thesis purely to image based methods.

Due to hardware limitations of our equipment, we will restrict the thesis to mainly focus on images of size 128x128. RAM and the memory of our graphic processing unit were our main limitations in this case, which made us restrict the image size as mentioned under system specifications (section 4.2.4).

1.4 Research Methods

There are several ways to conduct research. We have chosen an approach based on the Association for Computing Machinerys (ACM) research methodology, which was developed in 1989 by a task force set out by the ACM Education Board. Their main goal was to assemble the core fundamentals of computer science and computer engineering into a detailed report, which is given in the article “Computing as a discipline” [15]. This article splits the discipline of computing into three main paradigms; (i) theory, (ii) abstraction and (iii) design. The research conducted during the work of this thesis has been in compliance with these three categories in a variety of ways, which will now be explained in detail in addition to a brief description of each paradigm.

1.4.1 Theory

The theory part of the methodology is rooted in mathematics and describes the development of a coherent and valid theory. These are described as four steps, (i) characterise objects of study (definition), (ii) hypothesise possible relationships among them (theorem), (iii) determine whether the relationships are true (proof), and (iv) interpret results.

This category is supported in our work by analyzing how different neural networks could be able to learn grids in such a way that it could improve its segmentation performance. In particular, we investigate how this could be possible with a U-NET architecture and Pix2Pix.

1.4.2 Abstraction

The abstraction part of this thesis is rooted in the experimental scientific method and pertains to the investigation and development of the hypothesis. The report describes four stages of the investigation: (i) form a hypothesis, (ii) construct a model and make a prediction, (iii) design an experiment and collect data, (iv) analyse results.

The experiments that we have conducted during the work of this thesis falls under this category. Based on the results of our experiments, we investigated how these results came to be. We then developed a hypothesis as to why it behaved this way and adjusted further experiments based on this hypothesis. This was subsequently put to the test by running models with modifications in accordance with our hypothesis, which were either able to verify or disprove our original theory.

1.4.3 Design

The design part is rooted in engineering and pertains to the construction of a system to solve a given problem. It consists of four steps: (i) state requirements, (ii) state specifications, (iii) design and implement the system, (iv) test the system.

Our work complies with the steps of this category by the implementation of the prototypical framework for segmenting within grids and the grid segmentation networks. These systems were then used as a part of this thesis to conduct experiments in order to show usefulness of our implemented system.

1.5 Main Contributions

Over the course of this thesis, we researched if learning several degrees of precision in the segmentation maps could improve segmentation performance. Our research question was supported by three different objectives which would be the main focus of our work during this thesis. These are addressed below:

Objective 1: *Research and develop a framework for learning a grid segmentation mapping that takes an image and a grid size and segments the image within the specified grid size. The segmentation mapping should also be able to perform a normal pixel-level segmentation in addition to these grid segmentations, which would correspond to a grid size equal to the image size. The resulting framework should be able to work with several state-of-the-art neural networks.*

This objective is supported by our development of the grid segmentation framework and the corresponding pipeline. The grid segmentation framework enables machine learning algorithms to perform segmentations with degrees of precisions, which is realized by learning to segment within different grids. The framework included an approach for the encoding of grids as well as methods for generating the various segmentations within specific grids. Our framework was successfully implemented and worked for several state-of-the-art neural networks.

Objective 2: *Apply the grid segmentation framework to state-of-the-art neural networks for segmentation. Investigate how the amount and quality of the data affects the learning of the grid segmentations. Compare the results with corresponding state-of-the-art neural networks without grids.*

We applied the grid segmentation framework to Pix2Pix [41] and U-NET models [94]. All results considered, our models showed particularly good results for segmentations of lower precisions at the cost of less precision, while segmentations of higher precisions proved more difficult. Normal pixel-level segmentations proved to be the hardest task, where we obtained at least as good performance as current state-of-the-art neural networks. The precisions obtained for lower segmentations often allowed for good approximations to be obtained when higher precisions failed. We also investigated how the amount and quality of the data affected our models by using rotation augmented data and contrast enhanced data. We obtained our best results with rotation augmentation, which offered the most significant improvements. We only observed small improvements for contrast enhancement. These results were also confirmed by our baseline experiments.

Objective 3: *Investigate how the learning of these networks could be fully adapted and modified to the grid segmentation framework in order to enhance segmentation performance. Again, compare these results with corresponding state-of-the-art neural networks without grids.*

This objective is supported by the theories we developed for how our grid segmentation networks could learn in a manner that could improve segmentation performance and the following modifications we performed to accommodate the learning process. We developed a theory that the receptive fields of the different layers could represent different grids, in which adding skip connections between all upsampling layers to the last layer could allow grid information to be propagated directly to the last layer. We also incorporated the

grid size into the loss of our models in order to try to push the models towards better learning of grids. Generally, our experiments with these models obtained nearly the same results as previous experiments.

Regarding the overall research question, we were not able to confirm the hypothesis that learning several degrees of precision in the segmentation maps would enhance segmentation performance, as implied by the answers to our objectives above. However, we did manage to successfully implement a framework and corresponding neural networks that were able to segment within different degrees of precisions, which showed particularly good results for segmentation with less precision. This proved useful for the cases where higher precision segmentations gave limited results. It was often possible to obtain a good approximation when adjusting our segmentation precision input parameter to lower precisions.

Kvasir-Capsule Paper and Dataset: Besides the system we developed and described above, we also addressed the problem of automated polyp segmentation by the development of an open access dataset for gastrointestinal data, which we published a corresponding paper for during the work of our master thesis [76]. The dataset contains 44260 bounding-box labeled and medically verified images obtained from 118 capsule colonoscopy videos.

1.6 Thesis Outline

This thesis is structured into five different chapters. The two first chapters are introductory and explain the basic background needed for the research we conducted. Chapter 3 and 4 explain our main work and the ideas we developed over the course of this thesis. The fifth and final chapter concludes the whole thesis, sums up our work and present future work. The structure of the whole thesis including a summary of each chapter (except chapter 1) is presented below.

Chapter 2 - Background and Related Works

The thesis will begin by explaining the medical and technical background for using deep learning in medical applications. The chapter is essentially structured in two parts, one for the medical background and one for the technical details of machine learning. We begin by introducing the medical background and the structure of the gastrointestinal tract. We explain the various diseases the gastrointestinal tract is subject to as well as the available screening methods to detect them. We also explain the various landmarks used by algorithms and doctors to locate particular findings.

We then present the machine learning background, which describes the theoretical foundation for the methods we will be using in our research. We also provide an overview of relevant machine learning methods for medical data. The chapter starts with explaining what machine learning is and the various types of machine learning such as supervised, unsupervised and reinforcement learning. It further continues to explain the concept of a simple artificial neural network through basic Multilayer Perceptrons.

We then expand on this knowledge by gradually introducing more complex neural networks. From Multilayer Perceptrons, we proceed to introduce CNNs. We then build upon the knowledge of CNNs and introduce GANs and its various variants for image segmentation purposes, all of which build upon CNNs. These image segmentation techniques will be essential in the research of our

objectives. Finally, we present machine learning methods for video data by extending the theory of learning on image inputs such that the networks are able to learn across inputs over time.

Chapter 3 - Grid Segmentation Networks

We present the basics of the models we have developed over the course of this thesis which we collectively refer to as Grid Segmentation Networks. We first explain the Grid Segmentation Framework and the fundamental pipeline associated with this framework, which constitute the core of our basic grid segmentation networks. We then expand upon the machine learning background from the previous chapter and introduce the basic concept of Grid-GANs and Grid U-NETs, which in their most basic form are variants of General Adversarial Networks and U-NETs where we employ the grid segmentation framework.

Chapter 4 - Methods and Experiments

This chapter presents the main work of our thesis. Experiments have been conducted in an iterative manner, where we have modified our models based on results from previous experiments. We begin this chapter with describing the data material that we have chosen to work with. It presents how the different datasets were collected, what they capture and their limitations. General issues regarding data collection are explained and possible solutions are discussed, such as data augmentation and data enhancement. Various methods to improve the amount and quality of our data are then presented, with the goal of increasing the segmentation performance of our neural networks.

We then proceed to explain the basic setup of all our experiments, including how we decided to perform cross validation for our grid networks, what metrics we used to measure the segmentation performance of the grid networks, the basic architecture used for our experiments and at last system specifications for the experiments we conducted.

We then proceed to explain the various experiments we performed, starting with basic Grid-GANs. We then improve the segmentation performance by various data enhancement/augmentation methods as well as investigate how the inclusion of various grids affect segmentation performance. Subsequent experiments are direct modifications of how the model learns, which are performed in order to learn grids better and possibly increase segmentation performance. These are modifications pertaining to the loss and skip connections. We then perform baseline experiments for a corresponding U-NET models to measure the segmentation performance of our Grid-GANs.

Chapter 5 - Conclusion

Finally, we summarize, discuss and conclude based on the presented findings and discuss potential future work.

Chapter 2

Background and Related Works

Deep learning has achieved impressive success in a wide variety of fields over the recent years, including computer vision [14, 48], language translation [31, 86] and robotics [61]. There has been a particular focus on applying these methods for medical applications as well, where they have been successfully applied for the detection of skin cancer [35], breast cancer [22], brain tumors [85] and epilepsy [65]. In the very recent times of writing this thesis, deep learning has also shown promising results for diagnosing covid-19 based on computed tomography scans of the lungs [1].

This chapter will provide the background and related works that are essential to be able to investigate our problem statement and apply deep learning techniques for the detection of gastrointestinal diseases. The background and related works will be covered over two parts, one covers the medical background while the other covers the machine learning background. We begin with the medical background as this is vital to the understanding of the medical problem that we will try to solve by deep learning. We will then proceed to the second part of this chapter, which describes the technical details of relevant deep learning algorithms for our medical problem as well as introducing the models that will be utilized over the course of this thesis.

2.1 Medical Background

We will start by presenting the gastrointestinal tract and the various diseases it is subject to which we have been working with over the course of this thesis. We will then proceed to explain the various gastrointestinal landmarks, whose main purpose is to identify the location of a particular finding, but can also be used to detect various diseases as well. Furthermore, we will present various gastrointestinal screening methods that are currently used to detect these diseases and the various landmarks. We will also explain how they are used to collect the data which we will be working with in this thesis.

2.1.1 The Gastrointestinal Tract

The gastrointestinal tract, also called the digestive tract, is the main organ of the human digestive system. Its primary focus is to process food by absorbing nutrients through digestion and disposing of the resulting waste products through feces and urine. It can generally be divided into the upper gastrointestinal tract and the lower gastrointestinal tract, though some also differentiate the small intestine from these two categories. This is because the two procedures for inspecting the gastrointestinal tract generally do not cover the small intestine, which would instead have to be inspected by other alternatives, for instance wireless capsule endoscopy. We will be using the simplest two part division between the upper and lower gastrointestinal tract.

The common distinction between upper and lower gastrointestinal tract is marked and illustrated in Figure 2.1. Digestion begins in the cavity of the mouth where food is masticated by teeth, moistened by saliva and partially broken down by enzymes in the saliva. The food then travels down the esophagus to the stomach where it is further broken down by acids and enzymes. It proceeds to the small intestine, which is where nutrients mainly are absorbed and carried into the blood stream [32]. The mucosa of the gastrointestinal wall in the small intestine secretes enzymes and bile from the pancreas and gall bladder which further breaks down the food. The remaining product is then passed to the large intestine, which absorbs the remaining water, salts and vitamins and further to the anus where it is dismissed from the body in the form of feces.

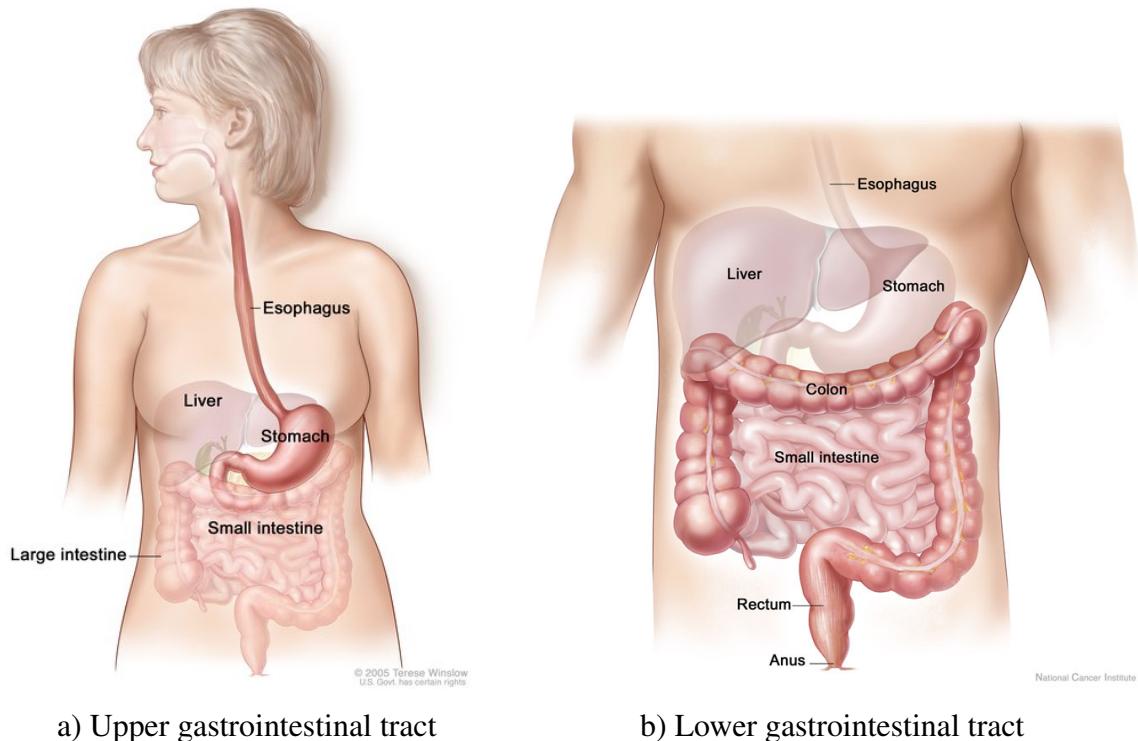


Figure 2.1: Illustration of the gastrointestinal tract ¹

¹Credit goes to author Terese Winslow and National Cancer Institute: <https://visualsonline.cancer.gov/details.cfm?imageid=7206>

2.1.1.1 Diseases in the Gastrointestinal Tract

The gastrointestinal tract is subject to a wide range of diseases, many of which can have very serious consequences for the patient, like types of cancers, infections or inflammations. Most of the work in this thesis will be centered around polyps, which are precursors to gastrointestinal cancer. During the work of this thesis, we also worked with other diseases when labeling the Kvasir-Capsule dataset, among them ulcerative colitis.

2.1.1.1.1 Polyps

Polyps are abnormal outgrowths of tissue which typically grow from a mucous membrane such as the innermost layer of the intestinal wall in the gastrointestinal tract. They can be flat, elevated or protruding into the lumen, for instance by a thin stalk [40]. They are further typically divided into two different categories called non-neoplastic and neoplastic, which describes their chance of becoming cancerous. Non-neoplastic polyps have a small chance of becoming cancerous and generally cover inflammatory polyps, hyperplastic polyps and hamartomatous polyps. The neoplastic polyps cover adenomatous polyps and serrated polyps. Both have a chance of becoming cancerous, though serrated polyps have the greatest chance of becoming cancerous.



Figure 2.2: Example of a polyp, taken from the Kvasir-SEG dataset [43].

All polyps are typically removed upon detection, as there is always a chance that they might become cancerous. The polyps are typically removed by fitting a snare around the polyp, cutting it off and further storing the polyp in a small bag for later retrieval. Polyps are often elevated prior to removal [81], especially if the polyp is very flat or otherwise not particularly easy to remove. The elevation of the polyp is done by injection of a lifting agent into the gastrointestinal wall close to the polyp. Usually the lifting agent is saline fluid for the smaller polyps, while other more viscous agents are preferred for larger polyps. This agent is in addition often dyed with a blue color which is later used to evaluate if the polyp was completely removed. This is important as polyps that are not completely removed may continue to grow and later become malignant.

2.1.1.2 Ulcerative Colitis

Ulcerative colitis is a chronic inflammatory disease which results in inflammation and chronic wounds in the form of ulcers in the colon and rectum. The disease usually develops between the age of 25 and 35. The primary symptoms are diarrhea mixed with blood, abdominal pain and cramping. Other symptoms are also anemia due to the extensive bleeding, weight loss and fevers. The course and the severity of the disease is often very individual, but it can generally become quite uncomfortable and have a large impact upon quality of life. A case of ulcerative colitis with clear findings of inflammation can be seen in Figure 2.3.

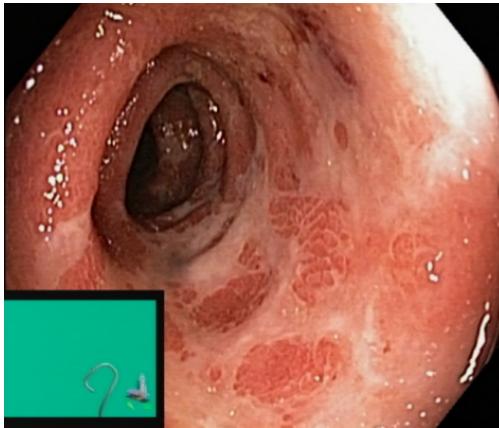


Figure 2.3: Case of ulcerative colitis with ulceration of the mucosa, as well as bleeding and swelling. Image taken from the Kvasir Dataset [63].

2.1.1.2 Anatomical Landmarks in the Gastrointestinal Tract

Anatomical landmarks are characteristic reference points in the gastrointestinal tract used by algorithms and humans to determine where the endoscopic device is currently situated in the gastrointestinal tract. These reference points are essential in determining the location of potential findings as well as when the colonoscopy is finished. Information pertaining to various diseases can also be extracted by often looking at the surrounding area of an anatomical landmark, as for instance is the case for the Z-line, which is inspected to diagnose gastroesophageal reflux disease. These landmarks were used and labeled by us during our development of the Kvasir-Capsule dataset, but in our final algorithms we ended up only detecting polyps and not these landmarks.

2.1.1.2.1 Z-line

The Z-line marks the exit of the esophagus and the entry over to the stomach. When inspecting endoscopic video data, the Z-line is known as the transition from the white mucosa of the esophagus to the red mucosa of the stomach, which can be seen in Figure 2.4. It is an important landmark which is also used for the detection of various diseases. It is typically inspected for the diagnosis of gastroesophageal reflux disease, which is caused by a weakness in the closing mechanism between

the esophagus and the stomach. This results in the rise of gastric acid into the esophagus which may cause heartburn, chest pain and regurgitation.



Figure 2.4: Image of a Z-line, taken from the Kvasir dataset [63]. The image clearly shows the transition from the white mucosa of the esophagus to the red mucosa of the stomach.

2.1.1.2.2 Pylorus

The pylorus is the muscle that connects the stomach and the first section of the small intestine (the duodenum). It ends as a sphincter which acts as a closing valve that regulates the amount of intestinal contents that are allowed to be passed over to the small intestine. From an endoscopic perspective, the pylorus can be observed as an opening encircled by pink colored mucosa, which can be observed in Figure 2.5. The stomach is often marked by characteristic gastric folds, while the small intestine is marked by intestinal villi, the latter of which are finger-like projections extending from the gastrointestinal walls of the small intestine. The pylorus is situated in the transitioning between these characteristic markings.



Figure 2.5: Image of a pylorus taken from the Kvasir dataset [63], which connects the stomach to the duodenum.

2.1.1.2.3 Cecum

The cecum is considered to be the beginning of the large intestine and is illustrated in Figure 2.6. It receives partially digested food from the final section of the small intestine (ileum) and connects to the ascending colon of the large intestine. The cecum is an important anatomical landmark as it indicates a full completion of a colonoscopy, which process the colon from end to beginning.



Figure 2.6: Image of a cecum taken from the Kvasir dataset [63], which is considered to be the beginning of the large intestine.

2.1.2 Screening Methods of the Gastrointestinal Tract

Colorectal cancer is one of the most severe diseases of the gastrointestinal tract and accounts for 10% of all cancer cases [10]. It is one of the leading causes of cancer related deaths globally, where the patients who do not survive have in many cases been nearly asymptomatic until the cancer has progressed beyond recovery [8, 25]. This is why colorectal screening has been a vital tool to reduce the number of deaths attributed to colorectal cancer. There are currently a variety of different methods for screening the gastrointestinal tract which we will now proceed to explain.

2.1.2.1 Gastrointestinal Endoscopy

Gastrointestinal endoscopy is a medical procedure for examining the gastrointestinal tract for abnormalities and diseases as those described in the previous sections (Section 2.1.1.1 and 2.1.1.2). The instrument used for the procedure is an endoscope, which in practice is a long flexible tube with a camera and is illustrated in Figure 2.7. Depending on which parts of the gastrointestinal tract that will be subject to examination, this long flexible tube is inserted either into the mouth or anus of the patient. The procedure where this instrument is inserted into the mouth is generally referred to as gastroscopy, while the procedure where the instrument is inserted into the anus is called colonoscopy. The procedure for gastroscopy and colonoscopy can be illustrated in Figure 2.7.

Colonoscopy procedures are generally very resource demanding and expensive to perform. The cost of a single colonoscopy is about 1000 USD, which makes it among one of the most expensive

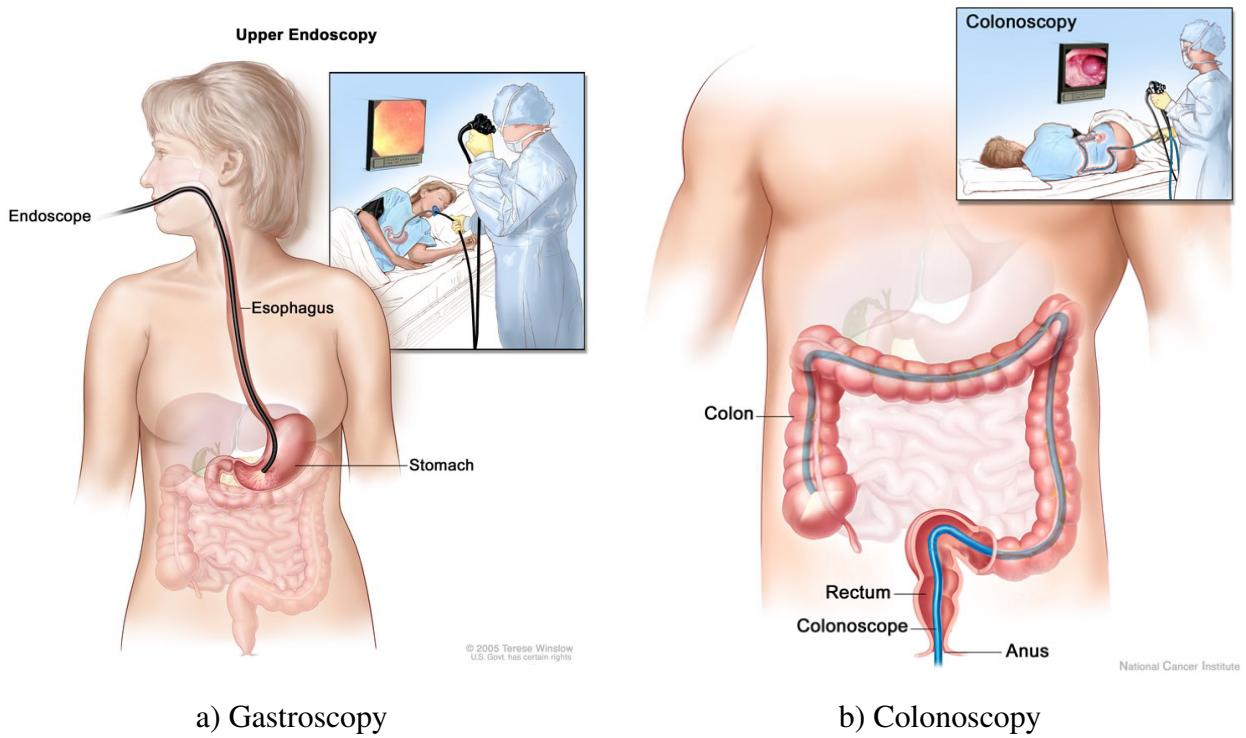


Figure 2.7: Illustration of how conventional endoscopy procedures are performed, either as gastroscopy or colonoscopy procedures ².

cancer screening procedures [68] and a significant investment for most of the U.S population. Most of the cost is attributed to the number of medically qualified personnel required and the significant amount of time required for the execution of the procedure [28]. The procedure is also known to be uncomfortable and very intrusive, which may cause patients to abstain from the procedure.

Wireless Capsule Endoscopy (WCE) is an endoscopic procedure which have been in use since early 2000 and eliminates many of these constraints [16]. The patient swallows a camera attached pill which records the digestive system and generally requires little to no involvement from medical personnel. Today as a consequence of its currently quite limited state, it is mostly used in addition to endoscopic procedures [23]. Especially for cases where conventional endoscopies do not find any abnormalities despite that the patient still suffers clear symptoms from gastrointestinal diseases. In contrast to colonoscopy, the intestine will not be inflated with air during the procedure, which might be both an advantage and a disadvantage in terms of disease detection. Small polyps may be more visible during WCE as they will not be pushed down by the air pressure that occurs when the intestine is inflated with air. The lack of air pressure might also cause certain findings to be difficult to spot. The following section will describe WCE in more detail.

²Credit goes to author Terese Winslow and National Cancer Institute: <https://www.cancer.gov/types/stomach/patient/stomach-treatment-pdq>

2.1.2.2 Wireless Capsule Endoscopy

Wireless Capsule Endoscopy (WCE) is a procedure for screening the gastrointestinal tract using a small pill-sized camera capsule. Figure 2.8 shows the camera capsule that was used to gather data to the Kvasir-Capsule dataset we helped develop. The capsules typically contain image sensors, pH-sensors and bleeding sensors in addition to light sources, antennas, transceivers and batteries for the capsule to operate properly. The camera capsule is swallowed by the patient, which travels through the digestive system while transmitting video data to a recorder that is worn on a belt around the waist. The camera capsule is subsequently excreted and retrieved from the rectum of the patient and a screening is then performed based on the observed recordings. The procedure is generally known to be of low risk. There is a small chance that the capsule might get stuck, in which case endoscopic or surgical removal is needed.

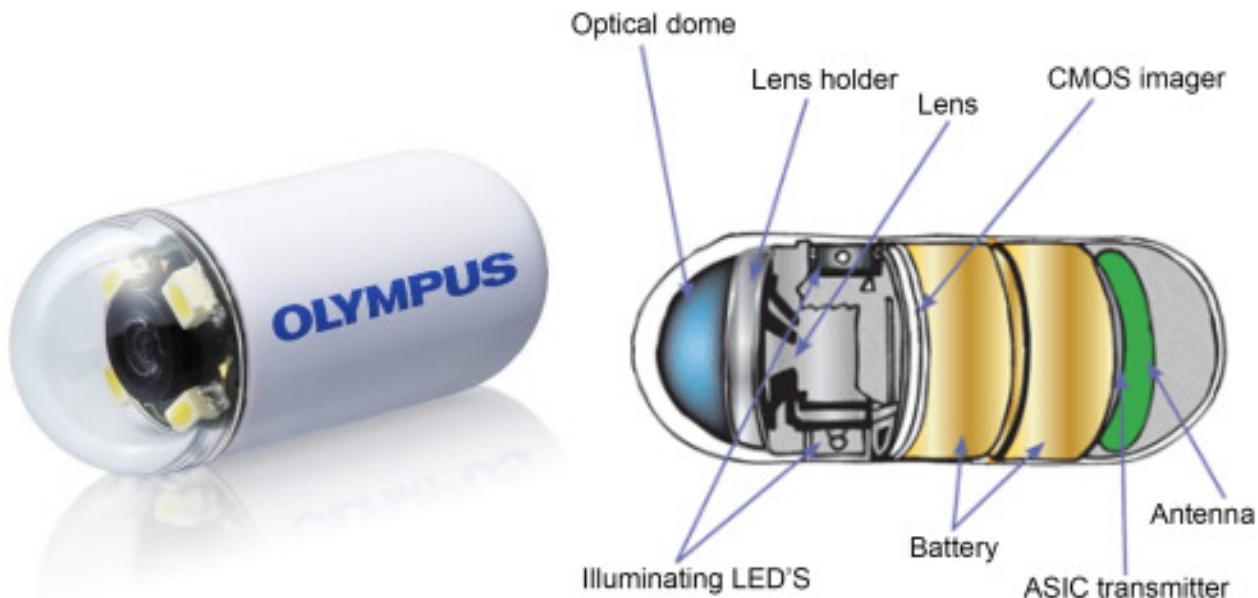


Figure 2.8: A typical example of a camera capsule used for WCE ³. The data we worked with was gathered by an Olympus EC-S10 endocapsule.

Most of the limitations for this type of screening are attributed to the small size of the camera capsule, which impose strict requirements to the size of the equipment. Camera capsules typically have low framerates and resolution compared to conventional endoscopy procedures, which could make disease detection more difficult. Through the advancement of the current limitations, there is a possibility that automated mass screening could be realized in a cheap and efficient manner by the use of these camera capsules.

The recordings can be up to 8 hours long and has to be processed. There is a similar problem for colonoscopy procedures where there generally is a lot of video data that needs to be examined, which for the case of colonoscopy is done during the procedure, with varying operator performance and detection rates [37, 50, 80]. The analysis of the video data is tedious work that is subject to

³Credit goes to Olympus Press Center: <https://www.olympus-europa.com/medical/en/Products-and-Solutions/Products/Product/ENDOCAPSULE-10-System.html>

various kinds of human errors, with miss rates of manual classification as high as 14% to 30% for certain findings [66]. Various machine learning techniques have yielded quite promising results for the analysis of these data. Among them are convolutional neural networks used for classification [2, 38] and segmentation [42, 82], as well as recurrent neural networks for classification [11]. We will now proceed to describe the various machine learning methods that have traditionally been used for the processing of these data and related data.

2.2 Machine Learning Background

In recent years, there has been an exponential growth in the popularity of machine learning. The year of 2012 sparked a wave of research into machine learning when AlexNet [48], a neural network based on deep learning, won the ImageNet challenge with a top-5 error margin over 10% lower than corresponding top competitors, corresponding to 15.3%. This was a result of the immense strides that have been made over several years in hardware, the inner workings of the algorithms and the advent of deep learning. The depth of the model is the core concept of deep learning and was made feasible by the utilization of GPUs during training.

Today, machine learning has achieved state-of-the-art performance within an extensive number of fields, including the speech and language processing [31, 89], facial recognition [71, 84], robotics [61] and a wide range of medical applications [22, 35, 65, 85]. In 2018, several convolutional neural networks were able to diagnose breast cancer at a level which was at least as good as trained pathologists [22]. The success that have been observed in the recent years motivates applying these methods to other medical applications as well which could assist hospitals and medical personnel in the diagnosis and treatment of patients. As for the scope of this thesis, our primary focus will be investigating new variants of segmentation models that detect and locate polyps found in the gastrointestinal tract.

2.2.1 Types of Machine Learning

This section will cover the major types of machine learning algorithms. The main focus of our research will be supervised segmentation models such as U-NETs and Pix2Pix. Unsupervised learning algorithms are briefly mentioned as background since some of the models applied in this thesis are related to unsupervised models. This is the case for our main model (the Pix2Pix) which is supervised [41] and closely related to CycleGAN which is unsupervised [94].

2.2.1.1 Supervised Learning

Supervised learning is a type of machine learning where a function is learned based on example input-output pairs which constitute the training dataset. This training dataset of labeled data is then used to infer a function by an iterative process of feeding samples to the model and comparing it with the corresponding ground truth. The parameters of the model are then changed based on how incorrect the prediction was. This process is repeated a given number of times or until the model has stopped improving. Common applications for this type of learning are image classification [2, 36,

48] and segmentation [43, 44, 83], speech recognition [34] and language translation [31, 88]. Some widely used supervised learning algorithms are Support Vector Machines (SVMs) [24], MLPs [26], and CNNs [48].

2.2.1.2 Unsupervised Learning

Unsupervised learning algorithms learn from unlabeled data. The goal is to find an underlying structure in the data and learn a task related to this structure. The dominant algorithms for this type of machine learning have traditionally been clustering algorithms, such as k-means clustering. These algorithms can for instance find an underlying structure in the data that could represent different classes, for instance different findings in the gastrointestinal tract. The task of the unsupervised clustering algorithm is then to group data together based on this underlying structure it has found [75]. The use of unsupervised clustering algorithms can greatly reduce the time needed to label data, which is particularly useful when there is a lack of experts to label data. There is a general lack of gastrology experts to label data, which makes these algorithms particularly important and interesting for gastrointestinal data.

The development of unsupervised learning algorithms has lately been particularly focused on generative models, such as variants of Generative Adversarial Networks (GANs) [29] and Variational Autoencoders (VAEs) [47]. Common applications for generative models are generating new media, such as text [91], music [12], or images [52]. For image applications, the goal may be to perform an image-to-image translation to generate something new, like translating real photos to paintings or translating a picture from summer to winter conditions. CycleGAN is a type of GAN that has gained a lot of traction lately, which has the ability to perform these tasks. It will be briefly covered in Section 2.2.3.2.2 due to its relationship to Pix2Pix, which will be used extensively in this thesis. Other common unsupervised algorithms are variants of autoencoders and latent dirichlet allocations.

2.2.1.3 Reinforcement Learning

Reinforcement learning algorithms learn based on the experience that it gains by interacting with a particular environment. The goal is to maximize a given award that is given by interacting with the environment. A significant difference between supervised learning is that it is typically not known whether a particular action was desired or not before many steps have passed, which differs from supervised learning where this is known the instant the action would be performed. This means that it trains considerably slower compared to supervised learning as gradient updates are rarer than for supervised learning.

Supervised learning also learns from labeled data, but this is not strictly the case of reinforcement learning. There is also a known target reward for the given actions which suggests reinforcement learning would not be entirely unsupervised either. The algorithms are often used in applications where there exists desirable outcomes and an environment to learn from and interact with. It has thus often found use in the task of learning to play games such as chess [74] or learning the actions of autonomous cars, which all have desirable outcomes and environments they interact with. Commonly used algorithms for reinforcement learning are Q-learning, Deep Q-networks and policy learning [56, 57, 73].

2.2.2 Neural Networks

Artificial Neural Networks (ANNs), or neural networks, are computational models vaguely inspired by the biological networks in the human brain. Although originally motivated by the behavior of this biological structure, it has since diverged from its original motivation in search of optimizing for state of the art machine learning results. It is because of this development that we will rather describe and motivate the different neural networks in this chapter on the basis of technical perspectives rather than biological perspectives.

In general, the computational architecture of neural networks can be described as a collection of interconnected artificial neurons. The output of an artificial neuron to a particular input is dependent upon numeric weights which are learned during training of the network. The response of the network to a particular input could be determined by configuring these weights. As the network learns to respond correctly when presented with given examples, the network can at the same time learn to generalize what it has learned to inputs it has never directly observed.

MLPs are among some of the simplest types of neural networks alongside perceptrons. They are also considerably more powerful than a simple perceptron as they can be universal function approximators, which is proven by the universal approximation theorem [17]. This gives them a wide variety of applications within several fields, ranging from cyber security [87] and e-mail spam filtering [18] to several medical applications. Particular medical applications include outcome prediction of colorectal cancer patients and prediction of epileptic seizures of epilepsy patients [9, 65]. Due to their simplicity and wide range of applications, we will proceed to explain the basic concepts of neural networks on the basis of MLPs.

2.2.2.1 Artificial Neurons

The basic building block of a neural network is the artificial neuron. They contain parameters known as weights and biases that determine how the particular neuron responds to a particular input. For the case of MLPs, every neuron performs a weighted sum of its input which is then offset by a bias term. The role of the bias is to shift the value of the weighted sum and hence be able to represent more complex outputs. This is subsequently passed to a non-linear activation function. An illustration of the basic ideas of an artificial neuron is given in Figure 2.9.

The activation function is one of the most important parts of the architecture of the neural network, as it introduces non-linearity into the model. Without it or if the activation function was linear, each value in the resulting output vector in every layer would be a linear function of its input, which would severely restrict the capabilities of the model. A classification model with this architecture would for instance only be able to learn linear decision boundaries.

The output of the neuron (that is, the subsequent result of applying the activation function) is passed either to the next layer or given as the output of the neural network, which suggests that the network architecture is feed forward. Every neuron in an MLP is also fully connected, which in addition to the feed forward layout of the network means that the input of a particular neuron is the outputs of all neurons from the previous layer.

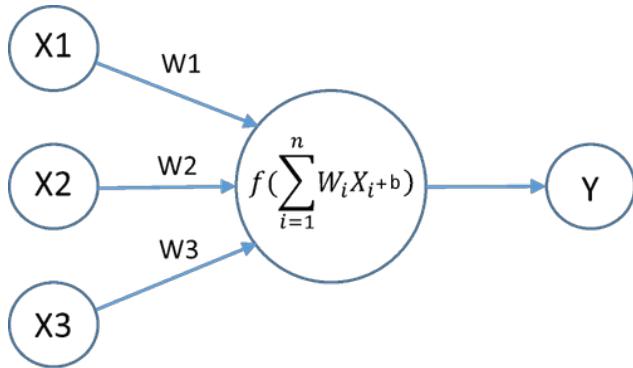


Figure 2.9: Illustration of an artificial neuron⁴ with three inputs X_1, X_2, X_3 that are summed and weighted with corresponding weights W_1, W_2, W_3 and passed to an activation function f to produce an output Y . This is a typical layout of a neuron for MLPs.

2.2.2.2 Network Layout

A neural network is structured layerwise where each layer is a collection of neurons. In the case of MLPs, each layer feeds to the next where the architecture typically consists of one input layer, an optional number of hidden layers and finally one output layer. This architectural layout can be seen in Figure 2.10. At least one hidden layer is required in order for the network to meet the requirements of the universal approximation theorem and to be used as a universal function approximator.

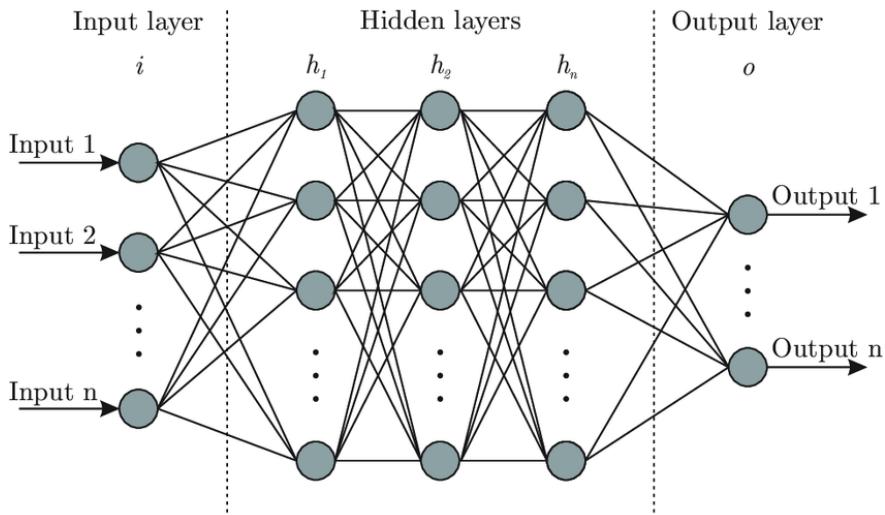


Figure 2.10: Example of a neural network⁵ called a feed forward neural network. Neurons are arranged layerwise where each layer incorporates increasingly more advanced features the deeper the layer is situated.

⁴Credit goes to author Arindam Mallik: http://www1.semi.org/eu/sites/semi.org/files/events/presentations/02_Arindam_Mallik_imec.pdf

⁵Credit goes to author Facundo Bre: <https://cutt.ly/sd2gWKW>

2.2.2.3 Training

Neural networks learn by a procedure known as training. This involves presenting the network with data from a training set which the network learns and infers relationships from. How these relationships are inferred is dependent upon the type of machine learning that is utilized. The general ideas for each type were explained in Section 2.2.1. This section will expand upon the theory presented for supervised learning, which is the main focus of this thesis. The main idea is that the network gains knowledge by being presented with examples, much like how the humans learn. But in comparison to the human brain, neural networks require at the very least thousands of examples to perform nearly as well as humans.

The training of the network consists of learning network parameters by iteratively comparing predicted output of the network with the corresponding ground truth data through an error function (also called a loss function or a cost function). Weights are subsequently learned by performing gradient descent on the chosen error function with respect to the corresponding gradients of the weights. The idea is that the network will learn to recognize general patterns from the ground truth data which in turn makes the network respond correctly when it is presented with similar input.

The gradients during training are calculated by an algorithm known as backpropagation, which computes the gradient of the loss function with respect to each weight one layer at a time, iterating backward from the last layer [93]. Each gradient is computed by use of the chain rule and dynamic programming. A lookup table of calculated gradients is generated to avoid calculating the same gradient repeatedly and is the cornerstone for the efficiency of the backpropagation algorithm.

How hyperparameters impact training

Hyperparameters impact the training and have to be carefully chosen and explored in order to get a model with optimal performance. The learning rate is the size of the corrective step that is taken to adjust for the error during gradient descent. A low learning rate makes learning considerably slower than a high learning rate, but has the potential for greater accuracy than a higher learning rate. Higher learning rates have lower accuracy but shorten the training time.

2.2.3 Neural Networks for Image Applications

MLPs are not particularly well suited for learning tasks pertaining to image inputs. Every layer is fully connected, which cause the parameter space to rapidly increase with the size of the input and furthermore the depth of the model. This makes learning difficult as the optimization problem for finding the minimum of the loss will get increasingly complicated the larger the parameter space is. This is known as the curse of dimensionality [27] and gives rise to a difficult optimization problem that will complicate the learning of the model.

In addition to the learning difficulties resulting from the curse of dimensionality, MLPs do not take the spatial structure of the input image directly into account. Input pixels that are far apart are effectively treated the same way as pixels that are close together as a consequence of the fully connected nature of each neuron in the network. This is very unfortunate as the spatial structure

of an image is very important information, which in the case for image inputs will furthermore complicate an already quite difficult learning problem.

2.2.3.1 Convolutional Neural Networks

A CNN is a type of feed-forward neural network that works on the concept of local connectivity instead of the full connectivity seen in MLPs. As will be shown throughout this section, this particular choice of design avoids many of the problems described in Section 2.2.3 when learning image related tasks with MLPs. Besides the design choice of local connectivity, CNNs share many of the same fundamental ideas of MLPs that were described in Section 2.2.2, both in terms of how they learn (that is, updating weights by backpropagation) and their fundamental structure.

The development of CNNs was historically inspired by the behavior of the primary visual cortex of the brain. It later diverged from the origin of its biological motivation, akin to the development of MLPs. The visual cortex of the brain consists of cells whose main goal is to detect light in small, overlapping subregions that are called receptive fields. The cells in the visual cortex can be interpreted as filters applied to the input space. This particular interpretation corresponds directly to the interpretation of a convolutional layer in a CNN.

2.2.3.1.1 Convolutional Layers

Local features and their corresponding descriptors have traditionally been the building blocks of computer vision algorithms. In this regard, fully connected layers are not optimal for computer vision related tasks, as they mainly extract global features (features pertaining to the entire image). The global extraction of features follows from the fully connected nature of the layers, which globally transforms the entirety of the input. For the purpose of computer vision related tasks, it would rather be preferable to have layers that can extract local features pertaining to a certain part of the image, which is in its very essence what convolutional layers are designed to do.

Convolutional layers are the fundamental building blocks of CNNs. They extract local features by locally transforming the input image through convolutions with filter kernels of learnable weights. This introduces local connectivity and ensures that the spatial structure of the input is taken into account, which is crucial as images are dominated by spatially local input patterns. The local connectivity that is introduced into the model also reduces the number of learnable parameters and yields a more parameter efficient model that limits the problems described in Section 2.2.3 associated with the curse of dimensionality when minimizing the loss. This will facilitate learning as the loss will become easier to minimize.

The basics of how layers of artificial neurons work were presented in Section 2.2.2 and explained by the use of MLPs. Much of the basic theory regarding backpropagation and artificial neurons remain valid when explaining the details of how convolutional layers work, which is why this section will proceed to explain the main differences from what has already been presented and expand upon this. As shown in Section 2.2.2, the output of a fully connected layer is a vector where each element is a weighted sum of the entire input which is subsequently passed to an activation function. That is, each element in the resulting vector from a fully connected layer is a dot product

between a vector of weights and the entire input, which is subsequently passed to an activation function.

In a convolutional layer however, this dot product is substituted for a convolution. Hence every element in this output vector would be the result of a convolution between the input and a filter kernel of weights which is then passed to an activation function. The output of a convolutional layer could thus be interpreted as a 3D volume where each slice (shown in Figure 2.11, called a feature map) would correspond to the result of one of these convolutions, which would be a 2D matrix of local transformations of the input.

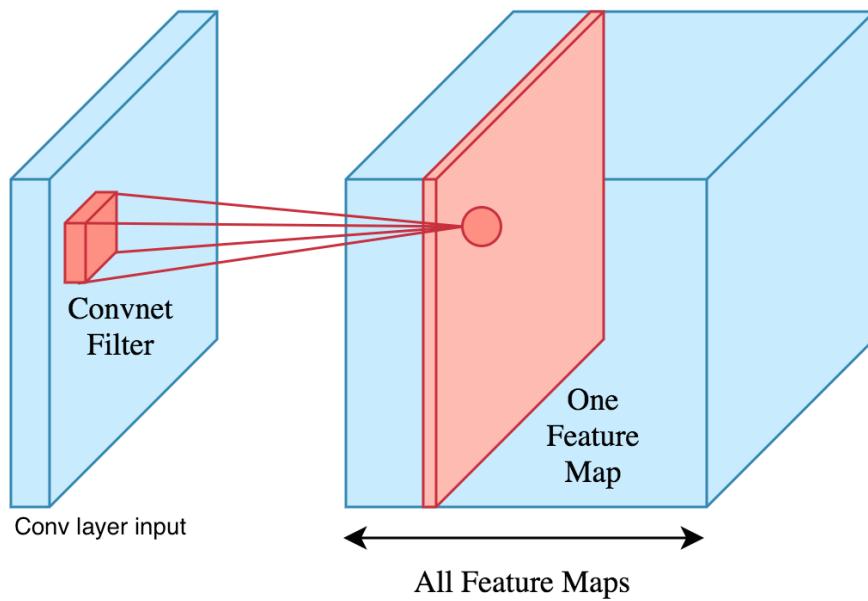


Figure 2.11: The output of a convolutional layer⁶ can be interpreted as a volume where each slice (called a feature map) corresponds to the result of a convolution of the input with a filter kernel of learnable weights that are learned during backpropagation and training of the convolutional neural network.

Thus convolutional layers transform the input locally through convolutions, which differs from fully connected layers that transform the entirety of the input through dot products. The output of the convolutional layer is furthermore passed to some non-linear activation function, which in the case of convolutional networks often is a Rectified Linear Unit (ReLU). This is then optionally followed by a pooling layer or some kind of downsampling.

⁶Credit goes to author Brilliant.org: <https://brilliant.org/wiki/convolutional-neural-network/>

2.2.3.1.2 Pooling Layers

Pooling layers have traditionally been a key component of convolutional neural network architectures and are typically placed after convolutional layers. They reduce spatial information of the feature map from the preceding convolutional layer by downsampling along the spatial dimensions. This ensures that the features are generalized to more positionally and rotationally invariant features, which can be important in order to recognize objects in a variety of different positions and rotations.

The downsampling performed by the pooling layer also reduces the number of parameters, which can have a variety of good effects which have already been mentioned in Section 2.2.3.1. It can facilitate learning as the optimization problem of minimizing the loss may become easier and it also reduces the overall number of computations, which can be important for the ease of use of large convolutional networks.

The downsampling is typically done by a max or average pooling. The operation is similar to a convolution in the sense that it is a sliding window. The size and stride of the particular windows are often set to 2. An example of this can be seen in Figure 2.12. The pooling layer does not contain any weights and is simply a downsampling, which implies that during learning and backpropagation, the layer would simply perform routing of gradients.

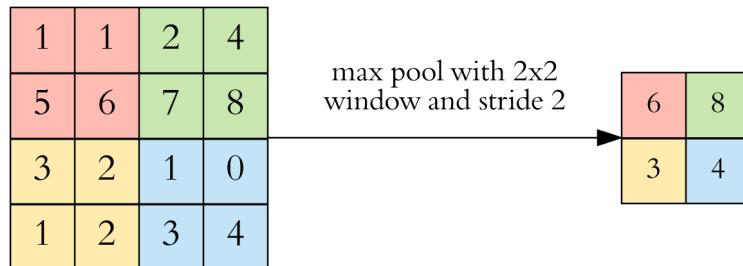


Figure 2.12: Example of max pooling⁷ of a two dimensional matrix using a stride and pool size of 2.

2.2.3.1.3 Network Layout

A classic unit structure of a CNN consists of one or more convolutional layers that are subsequently followed by a pooling layer. The stacking of such units can in principle be repeated indefinitely and is illustrated in Figure 2.13. Deeper layers of convolutional layers increase the size of receptive fields and the complexity of the recognized features. With sufficient depth, such a network would be known as a deep convolutional neural network, falling into the category of deep learning.

⁷Credit goes to author Hans Holten-Lund: http://program.fpgaworld.com/2017/More_information/extramaterial/Hans_Holten_Lund_2017.pdf

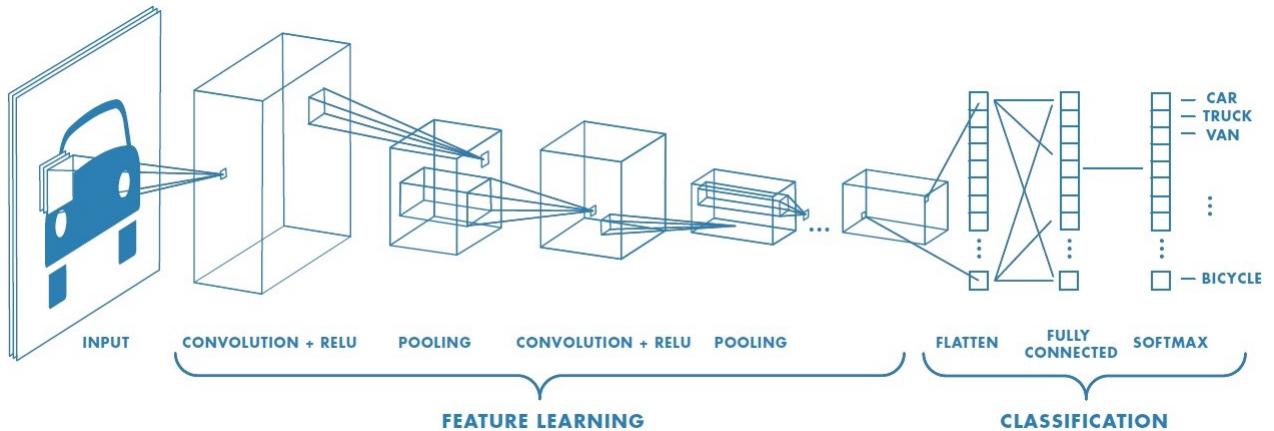


Figure 2.13: Convolutional Neural Network architecture for image classification⁸. It is divided into two parts - one for feature learning and one for classification. The feature extraction part constitutes a stacking of pooling and convolutional layers while the classification part is a fully connected layer.

2.2.3.1.4 U-NET Architecture

U-NET is a CNN architecture specifically developed for image segmentation [67]. It was published in 2015 and originally developed at the University of Freiburg in Germany for the purpose of segmentating biomedical images. It has since provided state of the art segmentation performance for medical image segmentation tasks and has been one of the most popular architectures among the medical imaging community. The U-NET architecture was originally based on the fully convolutional network, but this section will motivate the architecture from the perspective of the traditional CNN architecture described in Section 2.2.3.1. The U-NET architecture works in two parts, a downsampling part and an upsampling part which are more or less symmetric and results in a u-shaped architecture which is illustrated in Figure 2.14. This u-shaped architecture of the U-NET is what gave name to the architecture.

The downsampling part of U-NET transforms the input to a feature space which describes in high level features what the image contains. It incorporates few spatial details as to where exactly these findings are situated. More spatial details are needed in order to construct the exact location of the segmented objects, and this is where the upsampling part comes in to play. The upsampling part of the U-NET takes the result of the downsampling layer which describes what the image contains and fills in the details as to where these features are situated by information from skip connections.

The skip connections are illustrated as grey arrows in Figure 2.14 and are in essence feature information with better spatial resolution that are retrieved from the downsampling part of the network. They contain feature information from smaller receptive fields and thus represent features with finer spatial details. The upsampling part uses these features of finer spatial detail to recover spatial information which will be used to determine the exact location of a segmented object within the image.

⁸Credit goes to authors Ponnuru and Nidamarty from the Intel team: <https://software.intel.com/en-us/articles/speech-recognition-using-deep-learning-on-intel-architecture>

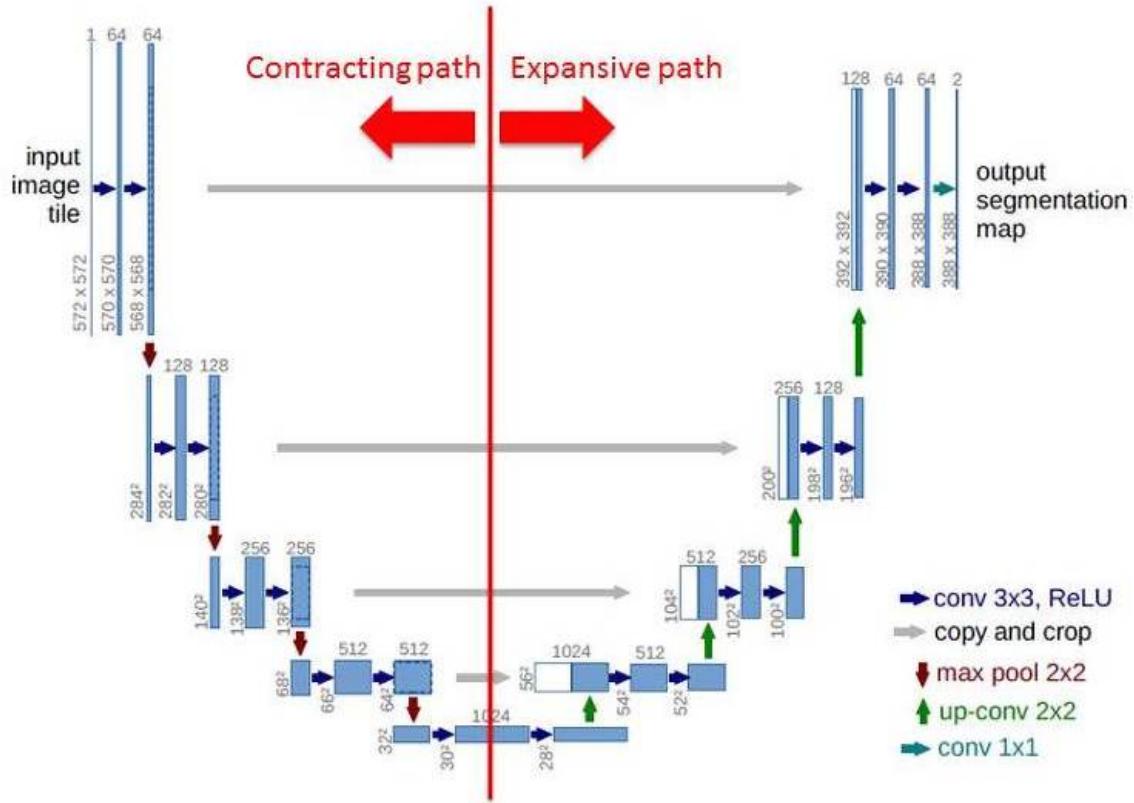


Figure 2.14: Original U-NET architecture from the U-NET paper⁹. This thesis will use a modified version of this architecture.

The segmented image is constructed by a series of transformations and upsamplings of the high level feature vector obtained from the last layer of the downsampling part. For each layer in the upsampling part, there is a skip connection that is realized by concatenating the input of that layer with the output from the corresponding layer in the downsampling part. This is illustrated in Figure 2.14. The concatenated result is then transformed and upsampled by transposed convolution layers. These layers perform an upsampling based on learnable parameters. The details for how they are implemented are implementation dependent, but the effect is in essence very similar to a zero padding the input and performing a normal convolution with filters of learnable parameters.

2.2.3.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) are a class of machine learning algorithms where two neural networks compete against each other in order to learn to perform a specific task [29]. Traditionally, this task has been to learn a particular distribution inferred from a training dataset in order to generate candidates from this distribution. A GAN could for instance learn to produce real looking photographs given a training set of photographs which the GAN is able to infer a distribution from [45]. This is accomplished by training two neural networks called a generator and

⁹Credit goes to authors of U-NET paper Olaf Ronneberger, Philipp Fischer, and Thomas Brox: <https://arxiv.org/pdf/1505.04597.pdf>

discriminator in tandem, where the generator generates candidates while discriminator evaluates these candidates.

During learning, the discriminator will try to infer the underlying data distribution from the training set. The discriminator will further try to learn to reject all samples that are not from this data distribution and in addition try to reject all samples generated by the generator. The generator will actively seek to fool the discriminator such that the data samples it generates are accepted by the discriminator. The generator does this by trying to generate data as close as possible to the data from the true data distribution inferred from the training data.

As the main focus of this thesis is based on image applications, this section will proceed to explain generative adversarial networks in this context, where the goal is to learn to generate realistic images from a distribution inferred from the training data. It should be noted that the theory presented here can be extended to other kinds of inputs and outputs as well, such as music [12], text [91] etc.

The generator does image synthesis by taking a randomized input that is sampled from a latent space, which often in practice is from a multivariate normal distribution. This sample is then mapped to an image from the true data distribution (inferred from the training data), which is done in order for the generator to be able to fool the discriminator. The discriminator will then perform steps to ensure that it recognizes the images generated by the generator and rejects them. Further, the generator will again perform steps to ensure that it fools the discriminator, which it does by trying to generate images which are as close as possible to samples from the inferred distribution of the training data.

As the two networks iteratively improve in this manner by learning from each others mistakes, the generator will be able to generate realistic images from the data distribution inferred from the training data. This tandem training of the generator and discriminator keeps running either until the performance of the generator is sufficient or until the generator and discriminator stops improving.

Common issues of GAN architectures

GANs are known for being a very difficult model to train, as they try to obtain an equilibrium between the the generator and the discriminator. This stands in stark contrast to the more traditional machine learning models where no equilibrium is required and there is merely the goal of minimizing a single loss function. Due to the fact that two neural networks are involved, GANs are also often known to be considerably more time consuming than other models.

Another common problem is mode collapse [53], in which the generator has the tendency to generate images containing the same features. Real data is often multimodal which suggests that the model has to infer a multimodal distribution when training. During the learning of this distribution however, some of these modes may have the tendency to collapse, due to the adversarial learning between the generator and discriminator. Partial mode collapse is more common than full mode collapse, which may manifest as a common set of colors that are always produced when generating images with the generator.

2.2.3.2.1 Conditional GANs

One of the drawbacks of traditional GANs described above is that it is in general not possible to specify a particular characteristic to the generator such that the generator only generates images with that particular characteristic. To be able to do this with a traditional GAN, the complex relationship between the latent space input of the generator and the generated images has to be figured out. This relationship is often so complex that in practice, different latent space inputs to the generator would have to keep being fed to the generator until one of the generated images has the given characteristic that the user would be looking for.

Conditional GANs were introduced to solve this problem, which it does by conditioning on the particular characteristic, which can be any type of auxillary information [55]. In practice, this condition is realized by acting as an additional input to the generator and discriminator. Training is then performed adversarially as described in Section 2.2.3.1.4, which will make the generator generate images of a particular distribution that is dependent on the condition and inferred from the training data.

2.2.3.2.2 Pix2Pix

Pix2Pix is a Generative Adversarial Network designed for image-to-image translation [41]. It is a supervised model and has a wide variety of applications, including image segmentation, translation of aerial photos or satellite imagery to maps and translation of pictures from day to night, to name a few. Pix2Pix is a type of conditional GAN (described in Section 2.2.3.2) where the generation of the output image is conditioned on the image that is to be translated. A significant difference from the previously described GANs is that the generator no longer takes a randomized input, which effectively makes the generated output deterministic. The authors observed that the model learned to ignore the input noise, which is why the stochastic input was removed. This means that the model is unable to match any other distribution other than the delta function, and this is described to still be an open problem by the authors.

The deterministic output may be disadvantageous for artistic applications but is ideal for segmentation purposes, which is what this thesis will investigate. This would ensure that the segmentation model always generates the same segmentation for a corresponding image. The model is supervised and learns based on paired image examples of inputs and outputs of the model in the adversarial manner described in Section 2.2.3.1.4. The generator architecture of the Pix2Pix is a U-NET architecture analogous to the architecture and implementation described in Section 2.2.3.1.3, while the discriminator is a PatchGAN discriminator.

The goal of the PatchGAN discriminator is to enforce the generation of high frequency details, which is done by penalizing structures in local image patches. A PatchGAN discriminator is effectively a convolutional neural network that classifies different patches of the input image. The final result of the PatchGAN discriminator is an average of all these patchwise classifications. While the PatchGAN discriminator enforces the generation of high frequency details, a $L1$ -loss term is added to the generator to make sure the generator generates low frequency details correctly:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y}[\|y - G(x)\|_1] \quad (2.1)$$

As one of the topics this thesis will investigate is how this model can be improved for the purpose of segmentation, we will introduce the details of the different losses. These losses are fundamental for the ability of the model to learn. The adversarial loss for both the discriminator and the generator can be described as

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log(D(x,y))] + \mathbb{E}_x[\log(1 - D(x, G(x)))] \quad (2.2)$$

where x is given as the input image and y is the corresponding ground truth, while G is the generator and D is the discriminator. The first term in Equation 2.2 can be described as the ability of the discriminator to learn to classify the ground truth. The second term describes the ability of the discriminator to classify generated images as fake. The output of the discriminator in this case indicates whether the given image pair is a valid image translation or not and gives an output of 0 if it rejects the image translation and 1 if it accepts the image translation.

The discriminator will seek to maximize the adversarial loss (Eq. 2.2) with respect to the parameters of the discriminator while the generator will seek to minimize the adversarial loss with respect to the parameters of the generator. A maximization of the adversarial loss will imply that the discriminator classifies ground truth as real (as is the goal of the first term of the equation) while images generated by the generator are classified as fake by the discriminator (as is the goal of the second term of the equation). A minimization will imply the opposite goal, where the generated images are classified as real by the discriminator.

In summary, the generator loss that is to be minimized with respect to the parameters of the generator G can be summarized as (combining Equation 2.1 and 2.2):

$$\mathcal{L}_G(G, D) = \arg \min_G \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G) \quad (2.3)$$

While the discriminator loss which is to be maximized with respect to the parameters of the discriminator D can be summarized as (from Equation 2.2):

$$\mathcal{L}_D(G, D) = \arg \max_D \mathcal{L}_{cGAN}(G, D) \quad (2.4)$$

In each training step, gradients are calculated for both the generator and the discriminator based on their respective losses given in Equation 2.3 and 2.4. The gradients of each of the neural networks are then respectively applied using an Adam optimizer [46] that updates the weights of the discriminator and generator based on these calculated gradients.

2.2.3.2.3 CycleGAN

CycleGAN is another Generative Adversarial Network designed for image-to-image translation [94]. It is effectively an extension of many of the concepts seen in Pix2Pix, but with the main difference that it is designed to be unsupervised. It does not learn from paired images, but seeks to find the image-to-image mapping by finding a underlying structure in the data by itself. CycleGAN has provided especially well results for artistic applications, such as mapping a picture to a painting. It has also shown to work well for many of the same applications mentioned for Pix2Pix [52] and many similar tasks, such as translating sketches of particular objects to realistic photos [39].

Though CycleGAN and Pix2Pix are related, CycleGAN is a considerably more advanced model. CycleGAN also learns an inverse map in order to learn the final mapping better in an unsupervised manner. It is effectively two conditional GANs that learn two different mappings, unlike Pix2Pix which consists of one single conditional GAN that learns one single mapping. Each of these conditional GANs are in essence very similar to a Pix2Pix in that they both use a PatchGAN discriminator and a U-NET architecture for the generation of images. There is however a difference in how these discriminators and generators learn, where the key differences are found in the different losses that are being optimized.

The core concept of the network is the cycle-consistency loss, which also gave name to the network as CycleGAN. This is the reconstruction loss of doing the image translation and then going back to the original image by the inverse mapping. Ideally this reconstructed image should be the same as the original input image, and this is what the cycle-consistency loss capture. In detail, there is one generator G that transforms the image from a domain A to a codomain B , and one generator F from codomain B to domain A . The cycle-consistency loss can be given as minimizing both $F(G(a)) - a$ and $G(F(a)) - a$ given an image a from the domain A . These are respectively the reconstruction losses going from domain A , then B , then A (called the forward cycle-consistency loss) and from domain B to A to B (called the backward cycle-consistency loss).

An identity loss is also minimized, with the goal of preserving the color temperature of the image. This is done by feeding the images in domain A to the generator that translates from domain B to domain A . The identity loss ensures that the generator gives the same images back as the CycleGAN should be able to understand the domain that the given image is located in. In order to learn this, unnecessary changes to an image are penalized. For instance, if the goal of the CycleGAN is to generate paintings from real images, then a painting can not be made into a painting. It should ideally be an idempotent operator and yield the same image. The same penalization is applied to the inverse transformation.

There is also an adversarial loss for each of the conditional GANs, which in essence is the same as described in Section 2.2.3.2.1.

2.2.4 Neural Networks for Video Applications

In all the networks described so far, every observation will be processed individually without considering any previously classified observations as there exists no memory of previously classified inputs [69]. This is unfortunate for the classification of video frames as previously classified frames have no influence on the classification of current video frames. That is, the contextual relationship between the video frames would not be considered in each video frame classification.

By contrast, there exist other kinds of networks that have an internal state over time that allows information pertaining to the contextual relationship to previous frames to be considered. Such networks are known as Recurrent Neural Networks (RNNs) [72]. Given a procedure that is able to perform a feature extraction of the video frames, features of the individual frames could be fed directly to a RNN. The RNN would then be able to consider the contextual relationship to previous video frames.

2.2.4.1 Recurrent Neural Networks

A Recurrent Neural Network can be thought of as an extension of a feed forward network with a feedback loop in every hidden layer such that the network structure is a directed acyclic graph [69]. A RNN can consequently be interpreted as an extension of an ordinary feed forward network where cyclic nerve cell connections are allowed. Such a network would have an internal memory of previously calculated observations which would allow for previously processed observations to impact future classifications.

Training and learning across time

All parameters for a single RNN-layer are typically shared over time with the goal of learning over multiple elements in the sequential data. As a consequence of this, the backpropagation through the network with regards to one of these shared parameters would lead to a result that is bounded by a power of the shared parameter. The power is equal to the number of layers that have been backpropagated through. The fundamental cyclic structure of an ordinary RNN is typically such that the network gets a significantly deep structure which leads to this exponent dominating the resulting gradient value.

This makes certain gradients in the network either explode in value or go towards zero, known as the vanishing and exploding gradient problem [6]. This is very unfortunate for the learning process, as these gradients are used to update the network parameters. In practice, this problem would make the sequential depth very limited, usually between 10-20 observations according to research done by Goodfellow [30]. In other words, the network would have memory of the latest 10-20 observations that have been processed.

This can be severely limiting depending on the frame rate. The best case is for WCE videos as they typically have some of the lowest frame rates of all endoscopy videos, typically around 2-7 fps. At its worst, when the frame rate is at its highest, the RNN would only be able to remember a couple of seconds of content from previous frames based on the assumption that only 10-20 frames would be remembered, as was indicated by the research performed by Goodfellow. Even in this best case scenario, this indicates that there would be a very limited learning based on previous frames with RNNs.

2.2.4.2 Long Short-Term Memory RNN

A Long Short-Term Memory RNN (LSTM RNN or LSTM in short) is a type of RNN whose main goal is to avoid the vanishing and exploding gradient problem. A LSTM extends the theory of ordinary RNNs with gates that control the flow of gradients [69] in order to avoid that gradients explode or vanish during backpropagation.

A common architecture of a LSTM-cell is composed of three gates that control the updates of the memory in the cell; one input gate, one forget gate and one output gate [69]. An illustration of these gates can be seen in Figure 2.15. Together the gates control the flow in and out of the cell and its memory. The behavior of the gates is controlled by weights which are learned during the training process. Due to these control gates, LSTMs can remember thousands of discrete time

steps ago [70]. Gated Recurrent Units (GRU) have a similar structure as LSTMs and can be used in sequence classification problems as well [13].

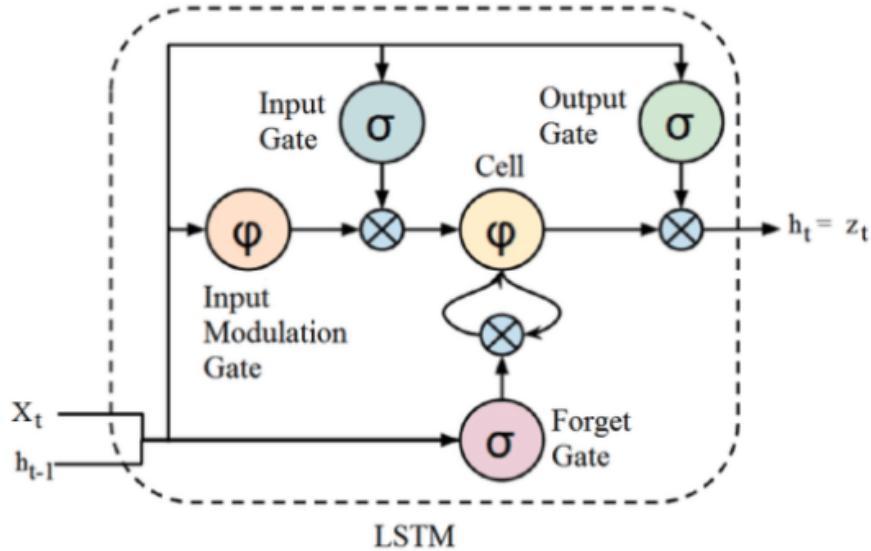


Figure 2.15: Illustration of a LSTM-cell ¹⁰. The input gate controls to what degree a new value will flow into the cell, while the output gate controls to what degree the value stored in the LSTM cell is used to compute the output of the LSTM-cell. The forget gate determines to which extent the value in the memory of the cell should be forgotten.

2.2.4.3 RNN-architectures for Video Classification

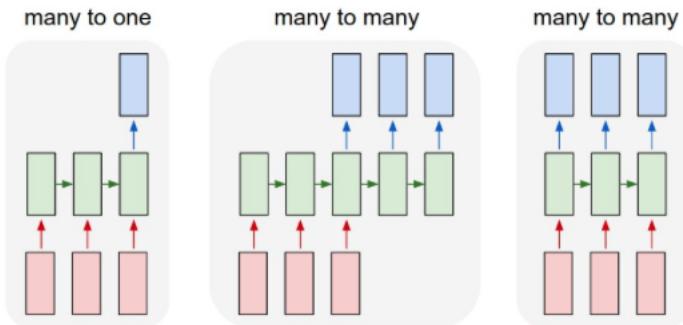


Figure 2.16: Illustration of possible RNN architectures ¹¹. A many-to-one architecture gives a classification of the whole video to one single category while a many-to-many architecture gives a classification of every single frame [90]. The last layer for the architectures is typically a fully connected layer that gives the final classification.

¹⁰Credit goes to author Sergio Guadarrama: <https://tiny.cc/lstm-cell>

¹¹Credit goes to author Andrej Karpathy: <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

2.2.4.4 Feature Extraction of the Video Frames

The premise of using Recurrent Neural Networks for the classification of video data is as previously mentioned a well performed feature extraction from the video frames. These extracted features could further be given to the RNN in order to classify the video frames by one of the architectures mentioned in Section 2.2.4.3. The feature extraction of the video frames can be a manual or automated process. A manual feature extraction could be done by a manual inspection of the images where appropriate features are directly calculated based on apparent visual clues central to the characteristics of the classes. It can often be difficult to find good manual features especially for colonoscopy applications as mentioned by Hovde et al [58]. An automated feature extraction technique through for instance a neural network is thus often preferred.

Automated feature extraction through Convolutional Neural Networks

An automated feature extraction could be performed by a CNN, which has the ability to find relevant features through the depth of its convolutional layers [20]. The last convolutional layer could further feed directly into the following RNN model, depicted in Figure 2.17.

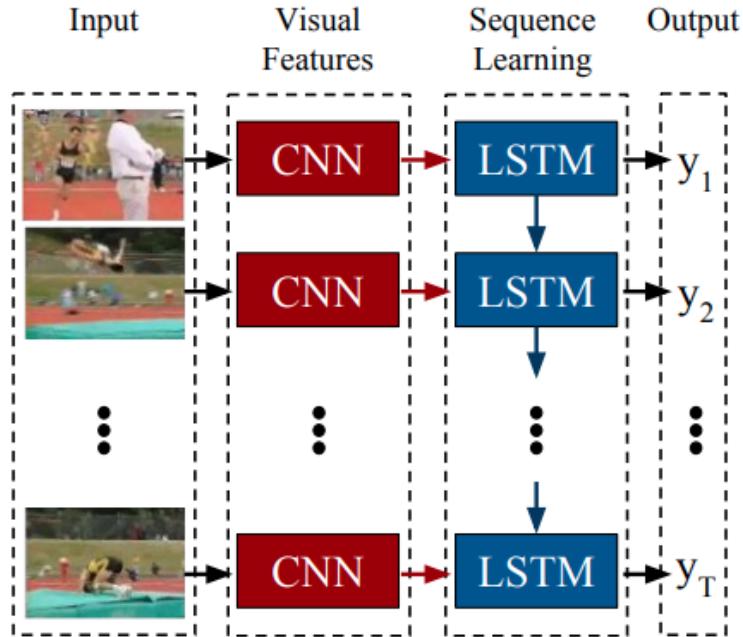


Figure 2.17: Illustration of how CNNs could be used for feature extraction to RNNs [20]. The weights of the LSTM and CNN are both shared over time and makes the model able to process videos of arbitrary length [20]. Each frame is fed to a CNN with the goal of obtaining a feature extraction for each of the frames. These feature extractions are then given as input to the RNN, which classifies each frame based on the contextual relationship from both previous and current frame.

2.3 Summary

In this chapter, we presented the background and related works necessary to detect diseases in the gastrointestinal tract, including both the medical aspects and the machine learning aspects pertaining to this problem. We described how screening of the gastrointestinal tract and the early detection of polyps is key to lower mortality of colorectal cancer. We learned the importance of locating the borders of polyps, which are essential to evaluate size and furthermore the chance of the polyp becoming cancerous, as large polyps generally have a greater chance of developing to cancer. We also learned how crucial it is to estimate polyp borders correctly during the removal of polyps, as they might continue to grow and become malignant if they are only partially removed. In addition, many polyps are also overlooked during manual screenings. All of these reasons motivate the use of automated segmentation for screening of the gastrointestinal tract. We provided several related examples where machine learning has obtained very promising results, which motivate the use of machine learning for this task.

Over the course of the machine learning background, we covered several state-of-the-art networks relevant for automated polyp segmentation. The U-NET has been one of the most popular architectures for segmentation in the medical imaging community, while several GANs have also shown very promising results for image segmentation. Among the GANs that have been very promising for segmentation is the Pix2Pix, which also has the ability to learn its loss function during training.

Based on these current state-of-the-art neural networks for segmentation, our goal is to investigate novel networks for the segmentation of polyps. The segmentation performance of these networks will then be compared with the segmentation performance of the corresponding state-of-the-art networks applied to the same task. The goal with the developed networks will be to answer our research question in Section 1.2, which questions the possibility of enhancing segmentation performance by learning several degrees of precision in the produced segmentation maps. The idea is that features used to create less precise segmentations can aid the training of pixel-level segmentation models. These different degrees of precisions could further be realized by learning to segment within grids.

The upcoming chapters will address the three separate steps that make up our solution, which is the development of the grid segmentation framework, how the framework is applied to our models and how the learning of the models could be better adapted to the framework. The first two steps will be covered in the next chapter, while the last step will be covered in the methods and experiments chapter, as the modifications we do are dependent upon observed outcomes from our initial experiments.

Chapter 3

Grid Segmentation Networks

The purpose of this chapter is to explain the essential core parts of the networks which could be used to test the hypothesis that motivated our research question in Section 1.2. Our hypothesis was based on the idea that segmentation models could improve their learning by building upon the knowledge of performing less precise segmentations. The degree of preciseness for the segmentations could further be realized by learning to segment within grids, where the neural networks that conform to this idea can be referred to as Grid Segmentation Networks. These networks generally operate by taking an image and a grid size parameter, where the latter is used to control the segmentation precision. These two inputs are then used to produce a corresponding segmentation within a grid corresponding to the input grid size. This chapter will explain the basics of how these networks can be built.

First, we will begin this chapter by describing a general framework which enables machine learning models to learn segmentations within several degrees of precisions by the use of grids. This framework will be the foundation of all our grid segmentation networks. We will then proceed to explain the motivations behind our models and how the framework could work in detail with these models to enhance performance.

For this chapter, we apply this grid segmentation framework to basic Pix2Pix and U-NET architectures without changing any of the core concepts of how these networks learn and behave. We are aware that such modifications may be necessary to make the networks better adapted to our framework, but this will be investigated and experimented with in the next chapter (chapter 4).

3.1 Grid Segmentation Framework

The grid segmentation framework is one of the core parts of the grid segmentation networks. The framework can essentially be envisioned as augmenting each of the training, validation and test datasets with grid information and use this data to learn a segmentation mapping that segments within a grid size given as a parameter to the mapping.

3.1.1 Ground Truth

For each image in each of the datasets, we generate a corresponding ground truth for all the different grid sizes. As we work with 128x128 images, the possible grid sizes are 2, 4, 8, 16, 32, 64 and 128. Examples for all the generated ground truths for a given image are given in Figure 3.1.

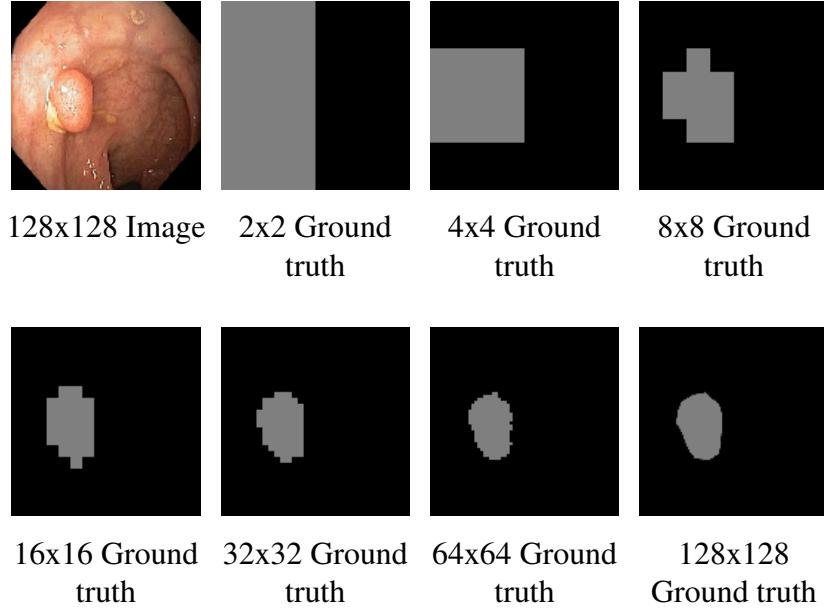


Figure 3.1: An image of size 128x128 and the corresponding ground truth segmentations for all the different grid sizes. Black indicates no finding while grey indicates a finding. As the size of the input image is 128x128, the ground truth for a 128x128 grid size is equivalent to the normal segmentation ground truth mask.

When creating the ground truth segmentation for a particular grid, we essentially divide the original ground truth segmentation into a grid. If the grid cell contains any parts of the finding, all cells within the grid cell are marked (which in this case is marked by a grey color).

Large grid size segmentations would be very close to the full non-grid segmentation of the image, as could be seen in Figure 3.1. From this observation, we hypothesize that these grids would be the most difficult to learn. There is the possibility that large grid size segmentations would be interpreted as normal pixel-level segmentations with noise as a consequence of their similarities. Based on this hypothesis, we initially restricted the focus to a few selected grid sizes which we hypothesized would serve best for a proof of concept of our theory. We picked the most distinct grid sizes, which were grid sizes of 2, 4, 8, 16 and 128.

3.1.2 Grid Encoding

Image segmentation models typically operate on three-dimensional inputs, namely a spatial and color dimension. To make the model understand what level of segmentation granularity should be produced by a given input, we need some way of ingraining this information into the image itself. To do this, we opted to add an additional channel to the color-dimension that represents what type of segmentation the model should produce. This additional channel uses a checkerboard pattern, where the number of the tiles signifies how precise the segmentation should be. An example of all the generated grid size encodings for each of the grid sizes can be seen in Figure 3.2.

For the case of GAN and U-NET architectures, another possibility is to encode the grid size in a small vector that is concatenated with the input to the upsampling part of the network. This is left for future work and is explained in detail in Chapter 5. Though this approach is interesting as this would be a more efficient representation of the few grid sizes that the encoding will represent, we chose the image grid encoding approach mentioned above. The grid nature of the image grid encoding could aid the model in learning features related to grids, even though it is less effective. This image grid encoding approach would also make the framework more generalizable.

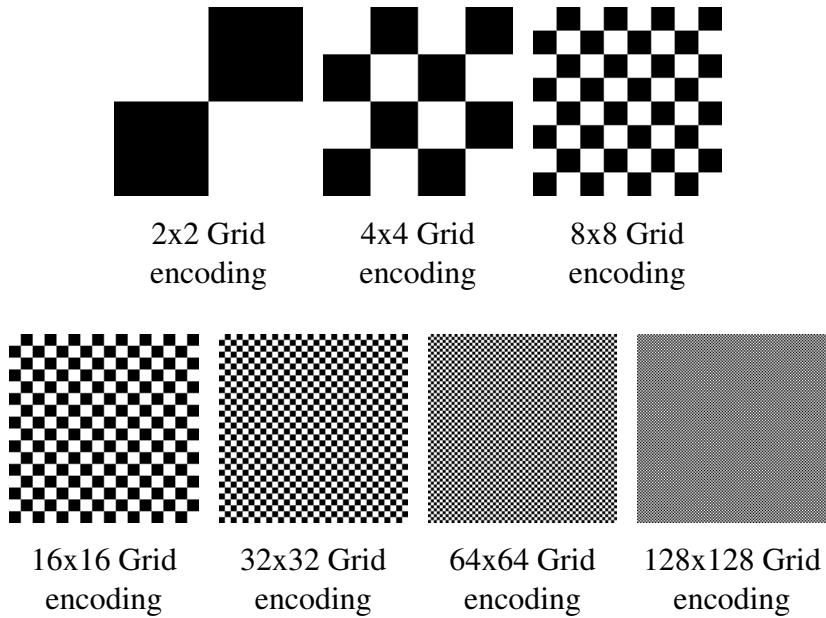


Figure 3.2: Grid encodings for different grid sizes. The grid encoding for a particular grid size is given by a checkerboard pattern. This checkerboard can essentially be characterized in its very nature as a grid and is how the grid size is encoded. For higher grid sizes, the grid encoding pattern gets difficult to see due to a dithering effect of the white and black squares, which makes the grid encodings appear grey at a distance.

3.1.3 Dataset Structure

The final augmented datasets that will be the basis of our training can be envisioned as datasets of 3-tuples, which consist of an image, a grid encoding corresponding to a grid size and the ground truth segmentation for the given grid size. This is illustrated in Figure 3.3. For later evaluation, we also create a dataset for each grid size in order to later evaluate the segmentation performance for each particular grid. This is also necessary in order to evaluate the networks segmentation performance for a normal pixel-level segmentation, which corresponds to segmentation performance for the largest possible grid encoding (which is equivalent to the image size).

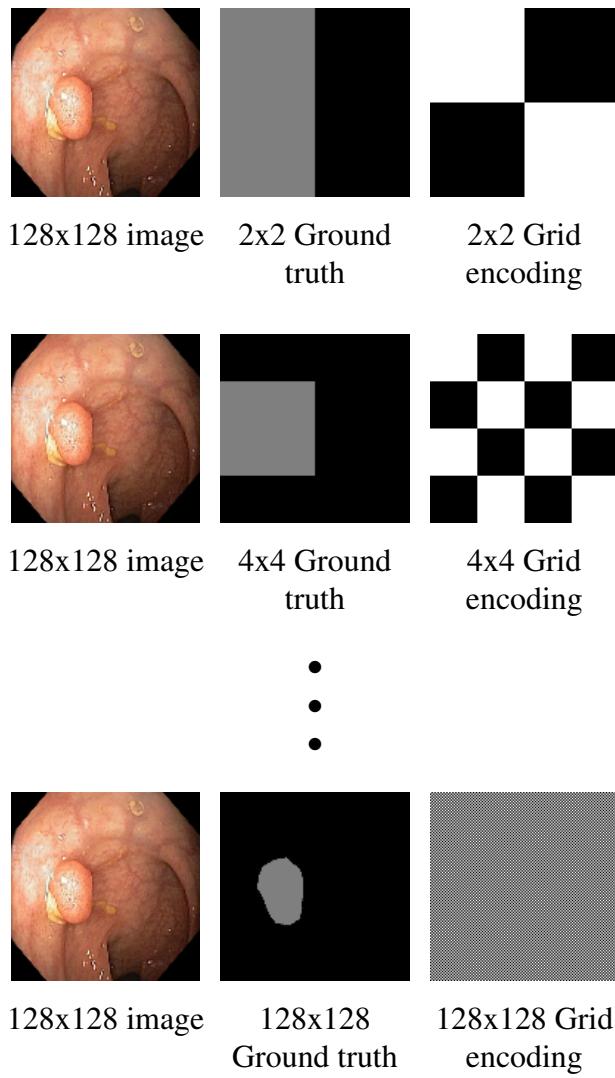


Figure 3.3: Illustration of how the datasets are structured. Grey indicates a finding while black indicates no finding. For each image in each of the training, validation and test sets, there is now a set of 3-tuples that correspond to segmentations of the image within a grid specified by the grid encoding.

3.1.4 Grid Segmentation Mapping

The grid segmentation mapping that we would like to learn can be illustrated in Figure 3.4. This is how all our final segmentation models will operate.

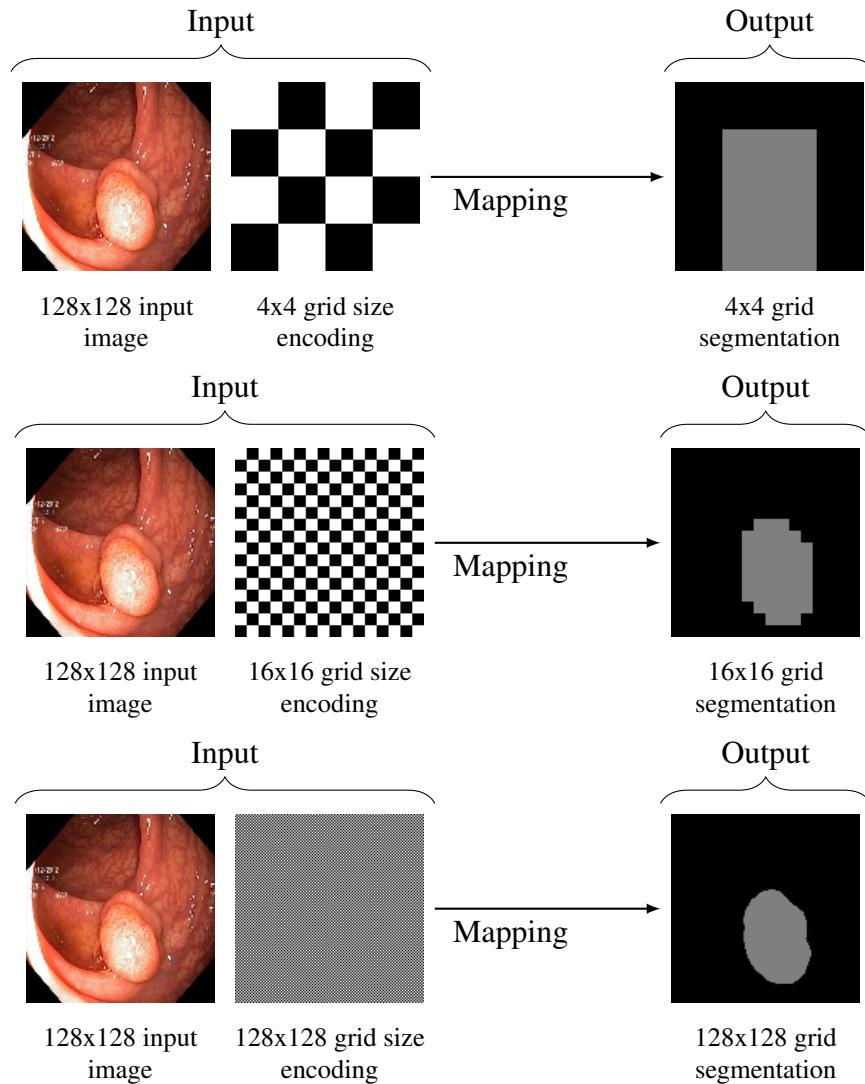


Figure 3.4: The goal is to learn a segmentation mapping that takes a given image and a grid size (encoded as an image) and segments the image within a grid size specified by the grid encoding. In the segmented image, grey indicates a finding while black does not indicate a finding. The grid encoding for a normal pixel-level segmentation (128x128 grid size) is cluttered to a grey color due to dithering, but follows the same chess pattern as the above encodings.

3.1.5 Pipeline Summary

We summarize the entire pipeline described above, which in essence can be illustrated in Figure 3.5.

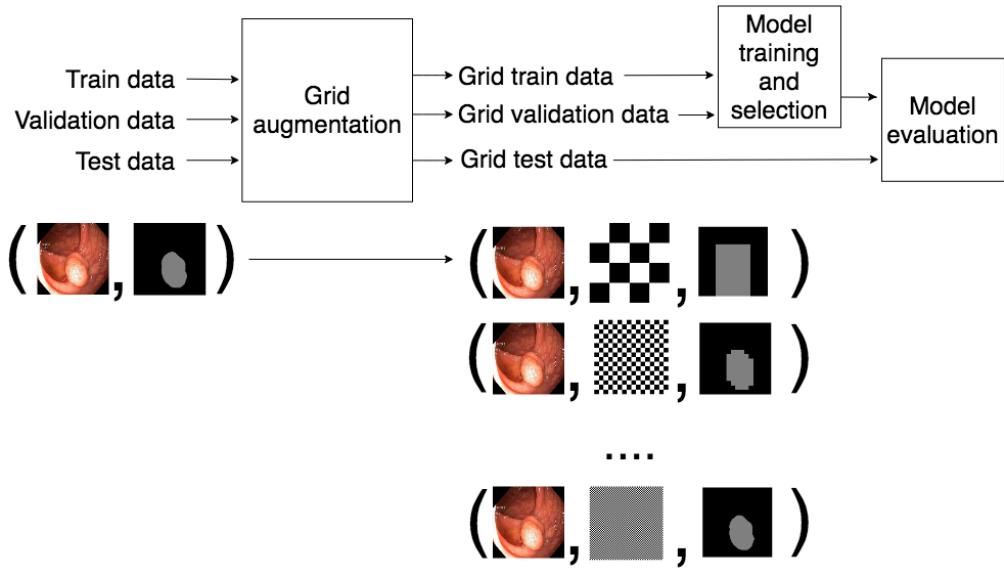


Figure 3.5: Summary of the pipeline, corresponding to how our grid segmentation framework is applied to our neural networks. We augment each training, validation and test datasets with grid information. For each image, grid encodings and corresponding grid segmentations are generated. These are used to train, select and evaluate the given model.

3.2 Grid U-NET

In its very basic form, the Grid U-NET is a U-NET architecture where we apply the grid segmentation framework to learn the mapping given in Figure 3.4. It will be used as a baseline model for the Grid-GAN which will be the grid segmentation network that is subject to the most of our focus.

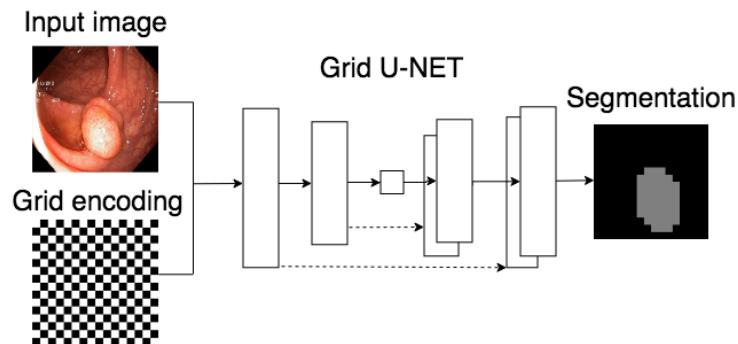


Figure 3.6: Basic illustration of how simple Grid U-NETs work, which essentially is the grid segmentation framework applied to a U-NET architecture. The U-NET architecture is illustrated by boxes that represent layers and stapled lines representing skip connections.

3.2.1 Motivation

Parts of our motivation for this particular model were based on a theory we developed for how the model architecture could work together with the grid segmentation framework in a way which could improve segmentation performance. This theory is based on the observation that it might be possible for the U-NET architecture to learn information pertaining to different grid segmentations through the different layers in the upsampling part of the network.

The key to understanding this observation is the various receptive fields for the features in each of the upsampling layers. The first upsampling layers are situated in the deeper middle part of the U-NET architecture described and depicted in Section 2.2.3.1.3 (background). These deeper layers represent image features with regards to large receptive fields and the reasoning behind this can be illustrated in Figure 3.7. These large receptive fields could in theory represent information which would correspond to small grids. This is because the large receptive field in this case could be interpreted as a grid cell of a small grid.

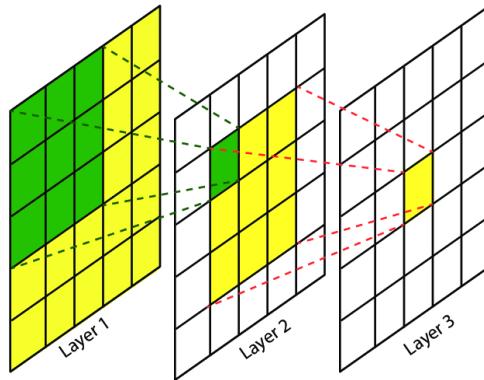


Figure 3.7: Illustration of receptive fields [51]. The yellow pixel value in layer 3 covers a 5x5 area marked as yellow in layer 1. Traditionally in CNN architectures, the deeper a layer is situated, the larger the receptive field of the given features for that particular layer. This means that the values in the deeper layers represent features from a larger area of the input.

The last upsampling layers contain features of narrower receptive fields which are retrieved from skip connections and correspond to the output of the initial downsampling layers of the model, as described in Section 2.2.3.1.3. These features of narrower receptive fields from the last upsampling layers could be able to represent information corresponding to larger grids, as the receptive field could be interpreted as a grid cell of a large grid.

3.3 Grid-GAN

The Grid-GAN is an extension of the Grid U-NET described in the previous section such that it learns in an adversarial manner by competing against a Patch-GAN discriminator. The goal is to learn the segmentation mapping in Figure 3.4 based on this type of adversarial learning. The basic form of the Grid-GAN is essentially equivalent to applying the grid segmentation framework to a Pix2Pix network. The generator network is a Grid U-NET while the discriminator is a Patch-GAN discriminator that is extended to fit the grid segmentation framework.

Both the generator architecture and discriminator architecture of this model is in accordance with the theory already presented in the background (Section 2.2.3.2.1), but with the extension to two different inputs. The discriminator is a CNN that takes an image, a grid size encoding and the corresponding grid segmentation of the image, which can be denoted as a 3-tuple (image, grid size encoding, segmentation). The discriminator will learn to recognize what a valid segmentation looks like by learning to recognize which of these tuples that correspond to valid segmentations. The discriminator will also to reject all segmentations generated from the generator as invalid segmentations. This is summarized in Figure 3.8.

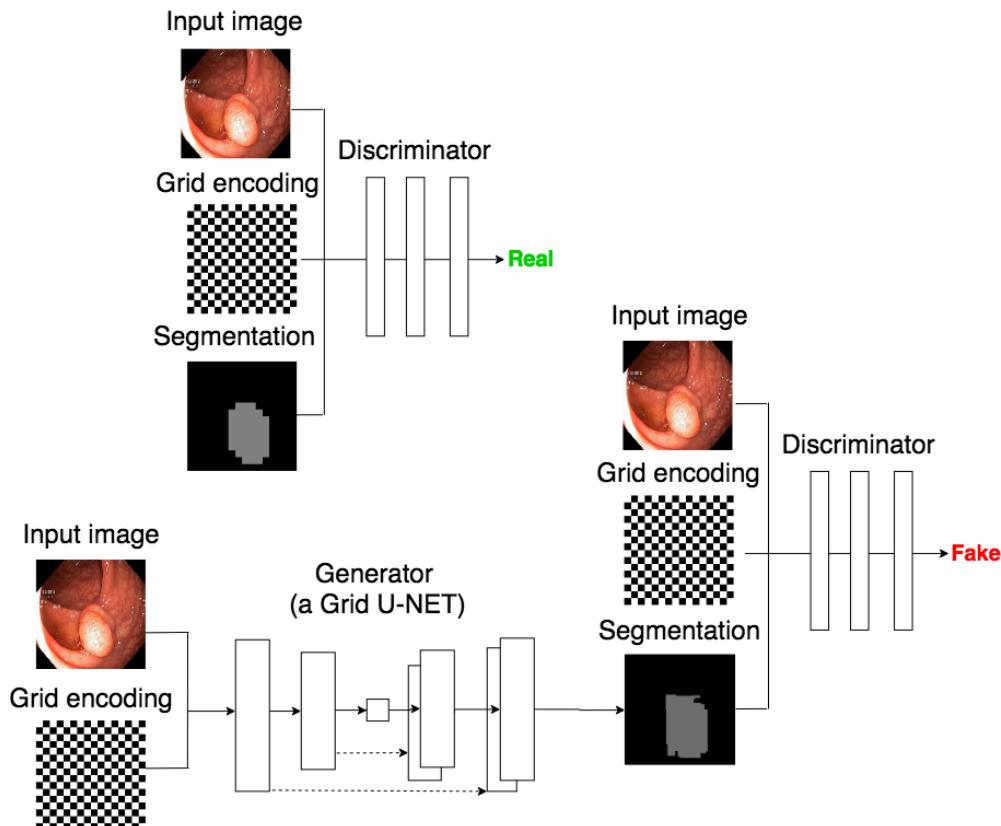


Figure 3.8: Example of the objectives for a basic Grid-GAN. This illustrates the objective of the discriminator of the Grid-GAN, which is the opposite of the objective to the generator. As shown in the illustration, the discriminator seeks to classify the ground truth as real and generated images as fake. The objective for the generator is the opposite, where it seeks to trick the discriminator into classifying generated images as real.

The goal of the generator is to trick the discriminator to classify all segmentations it generates as valid segmentations. As the discriminator learns to recognize only valid segmentations, the generator will be driven to generate valid segmentations.

3.3.1 Motivation

The goal with the Grid-GAN is in essence to learn a Grid U-NET by adversarial learning. Thus would the motivations described in the Grid U-NET section (section 3.2.1) largely apply here as well. In addition, there are also the motivations of the adversarial learning that is introduced by the GAN model, which incited a particular focus on these generative models during the research of our objectives.

Pix2Pix and CycleGAN are two types of GANs that have gained a lot of traction lately in the field of image-to-image translation and segmentation. These types of GANs are particularly interesting due to their ability to learn their loss functions as a part of the training of the network. This differs from other segmentation methods where a lot of manual effort still goes into designing effective losses. The loss for these GANs is simply based on a high level goal which states that the segmentation should be indistinguishable from the reality presented by the ground truth. This high level goal is then incorporated into the training in order to learn the mapping.

Due to time constraints of the thesis and the similarities between these two GANs, we have decided to limit our research to only one of these two models. Pix2Pix learns one generator and one discriminator in order to find a segmentation mapping. A CycleGAN has about twice the complexity, as it learns two pairs of generators and discriminators, one for the segmentation map and one for the inverse of that segmentation map, through minimization of a cycle-consistency loss. We have decided to go for the Pix2Pix as it is the simplest model and serves well for a proof of concept of our research question. In addition we would like to investigate grid segmentations using supervised learning, but CycleGAN is unsupervised.

Section 2.2.3.2.1 explained how the Pix2Pix can be used to segment images in addition to learning a wide variety of other image-to-image mappings. We seek to adapt the Pix2Pix for the purpose of learning to segment within different grids with the goal of improving segmentation performance. Corresponding U-NET architectures will be used as a baseline when evaluating the segmentation performance of the models, which are realized as Grid U-NETs.

3.4 Summary

This chapter relates to our first research objective, which is to develop a general grid segmentation framework for learning a grid segmentation mapping that can segment images within a specified grid size. This grid segmentation framework be envisioned as augmenting the dataset and using it to train a model to learn this mapping. The augmentation procedure takes each image and corresponding ground truth and creates grid segmentation ground truths for all the possible grid sizes. The items in our final datasets are essentially 3-tuples of an image, a grid encoding and a corresponding grid segmentation, which are used to train a model for segmenting images within a particular grid.

We describe how the framework could work with U-NET and Pix2Pix architectures to theoretically enhance segmentation performance and why we have chosen to work with these models. We hypothesize that it could be possible for these models to incorporate grid features layerwise as the receptive field of a particular layer could in theory correspond to a grid cell of a particular grid. The features pertaining to higher grid sizes could thus build upon feature information from lower grid sizes represented in the lower layers. We explain the reasoning behind our particular focus on Pix2Pix, which we suspect could be especially interesting to research due to its ability to learn its loss function during training.

The next chapter will address the last and final step of the overall solution presented in this thesis, which is how the learning of the models could be better adapted to the framework. This is in part motivated on the basis of our conducted experiments.

Chapter 4

Methods and Experiments

The previous chapter explained our motivations behind the grid segmentation networks and how they in their most basic form could work. The goal of this chapter is to put these basic models to the test and further investigate how the networks could be adapted to fit the grid segmentation framework better, which could improve segmentation performance. The experiments will be presented iteratively and new changes to the architectures will be proposed based on observations from the outcomes of previous experiments.

We begin by explaining the data material we used for conducting our experiments. We will then proceed to explain the basic setup of our experiments, including how we adapted k-fold cross validation to our grid data, what metrics we used to measure segmentation performance of our grid segmentation networks, the basic architecture we used for all our experiments and the system specifications for the system we ran our experiments with. Finally, we will proceed to the various experiments, starting with Grid-GAN experiments. We observe results from experiments and develop new theories which we again put to the test. At last, we present corresponding Grid U-NET experiments and summarize our findings.

4.1 Data Material

This section will explain the datasets used for our experiments, how these datas were handled and the various issues associated with the gastrointestinal data we worked with, in particular the lack of data. This issue leads to the introduction of various data augmentation and enhancement techniques which we chose to explore for our grid segmentation networks.

4.1.1 Datasets

The training and evaluation of our models involve two different datasets; Kvasir-SEG, CVCClinicDB. The Kvasir-Capsule dataset we developed was left for future work. All of our experiments were conducted using Kvasir-SEG for training, as this is the largest and most diverse dataset of the two datasets for pixel-wise segmentation, while CVCClinicDB was used as test data.

4.1.1.1 Kvasir-Capsule Dataset

Kvasir-Capsule contains 44260 labeled and medically verified images obtained from 118 capsule colonoscopy videos, which we developed and labeled in cooperation with Simula and Augere Medical [76]. The video data was collected from patient examinations at the Department of Medicine at Bærum Hospital in Norway and recorded with a Olympus Endocapsule 10 System. These videos were then subsequently labeled with bounding boxes by us in cooperation with a doctor working at Augere Medical. Finally, we performed a split of each of the video findings into respective images.

This dataset was ultimately not prioritized for our experiments and had to be left for future work. The limitations which lead us to this decision were described in Section 1.3.

4.1.1.2 Kvasir-SEG Dataset

Kvasir-SEG is a dataset of 1000 colored polyp images including its corresponding pixel-wise polyp segmentation [43]. The images were collected using endoscopic equipment from Vestre Viken Health Trust in Norway, which comprises of four different hospitals that provides health care to about 500 000 people. The data is annotated by medical experts from the Cancer Registry of Norway (CRN) and Vestre Viken Health Trust. This dataset is the largest and most diverse of the pixel-wise segmentation datasets, which is why it will be used for the training of our neural networks.

4.1.1.3 CVCClinicDB Dataset

This dataset consists of 612 images of polyps extracted from colonoscopy videos [21] with corresponding segmentations of the polyps. The images were extracted from 23 standard patient colonoscopy examinations performed at the Hospital Clinic of Barcelona in Spain [7]. All the sequences containing each finding were extracted which resulted in 31 sequences of 31 different polyps. On average, 25 frames were obtained from each polyp. In total, the dataset consists of 612 images of size 576x768. Clinical experts drew segmentations of each polyp in each of these images. This dataset will be used as an independent test dataset to evaluate our models.

4.1.2 Issue of Data

One of the most significant challenges of applying deep learning to medical applications is the lack of labeled medical data. There are several reasons for this, many of which are due to the legal and ethical challenges of using peoples data, especially patient data. Patient data comes within the most intimate sphere of a persons life, which is especially evident for data from colonoscopy procedures due to the intrusive nature of how the procedure is conducted. These types of data are often subject to some of the strictest data laws there is, making them particularly difficult to obtain and work with.

There is also currently a general shortage of medical experts specialized to gastrology and its expected to grow due to a growing elderly population in many developed countries. Even though

there are cases where medical data is available, there is the problem of having medical experts available to label them. Many medical experts may also not be willing to perform the tedious work of labeling the data which furthermore exacerbates the problem of getting labeled data.

The lack of data in the medical field is a significant challenge as neural networks needs massive amounts of training data. The amount of available medical data is often not enough to train and evaluate for many different use cases and could lead to bad performance. A solution to this problem is to make the best out of what data is available and improve the quality and amount of the available dataset. This can be done through various data enhancement and augmentation techniques.

4.1.3 Data Enhancement and Augmentation

Medical datasets often contain high dimensional feature spaces with relatively few samples, which can lead to poor performance when applied to machine learning models. This is why data enhancement techniques are particularly important for medical applications. It is often necessary to use some kind of data enhancement on medical datasets in order to produce models with good performance, which is why we will spend some time investigating and explaining these techniques.

The data used for training lays the foundation of what the neural network will be able to recognize, which suggests that a natural way to increase performance accuracy of the neural network could be to improve the quality and/or the amount of data that is being used to train the neural network. One approach is to artificially increase the existing dataset with transformed samples created from the existing dataset which could be logical for the network to learn.

This will introduce variability into the dataset which the neural network may learn from, thus showing the network that objects exist in a variation of different states. If the transformation of the images is not logical for the network to learn, which would suggest that the transformation produces unrealistic images, then the remaining option is to enhance the entire dataset by this transformation and let the transformation be a part of the pipeline.

4.1.3.1 Rotation Augmentation

A camera capsule may be angled in any position which suggest there is no logical up or down for a given video frame. Hence a rotation transformation would be natural and logical to use in the case of data augmentation. For every image in the training dataset, we create a set of images where we alter the angle of the image. These sets of images are subsequently added to the dataset in addition to the original data.

We perform the data augmentation on every labeled frame with respect to a set of equidistant angles. This is done in order to avoid skewed data, as the equidistant angles of the images assure that they are all as distinctly different as they can be in terms of their angled orientation. For a given frame, three rotated variants are created with angles 90° , 180° and 270° in addition to the original image.

4.1.3.2 Contrast Enhancement

Many of the features that are of great importance for the detection of gastrointestinal diseases are bound by different color and border properties, which have been the target for several polyp detection systems [3, 54, 92]. The occurrence and the amount of bleeding are for instance bound by different color and border properties and could be very important for the detection of polyps. In order to learn that information pertaining to color and borders is crucial, contrast enhancement of the data may be of interest. A contrast enhancement would make these border regions and colors more pronounced and may thus improve the learning of different patterns pertaining to borders and colors.

Adaptive Histogram Equalization (AHE) is a technique to improve contrast in images and is a local adaption of ordinary histogram equalization [62]. Ordinary histogram equalization transforms the entirety of the image using the normalized cumulative histogram for the entire image scaled with the maximum range value of the image. This results in an image where the global contrast is enhanced. AHE works by adapting this algorithm to local neighbourhoods and would instead improve contrast locally. AHE transforms each pixel in the image based on its local neighbourhood. The transformation function is thus adapted to local areas in the image, resulting in better contrast in the more homogenous parts of the image. The transform is in its simplest form completely analogous to its counterpart in ordinary histogram equalization. Every pixel in the image is transformed using a scaled normalized cumulative histogram of the local neighbourhood of the pixel.

AHE can however result in the overamplification of noise due to the transformation of each pixel being bound by the values in a small neighbourhood of the pixel. An alternative variant of this algorithm known as CLAHE [62] limits noise by clipping each of the local histograms at a predefined limit and redistributes the clipped values equally across the entire histogram. This preserves the clipped values while simultaneously limiting the overamplification of noise. When doing contrast enhancement, we will be using CLAHE on every RGB channel.

The histogram equalization used in CLAHE often produces unrealistic effects in photographs. This is further worsened in color images when applying the contrast enhancement to each respective channel in the RGB space as the color balance will not be preserved. Due to these unrealistic effects, the color enhanced images cannot be used in addition to the original data, but rather in place of the original data. This would also entail a change in the pipeline such that every image would have to be contrast enhanced prior to being fed to the neural network.

However in theory, this type of contrast enhancement of the data using CLAHE could improve the learning of the characteristics of colors and borders in the image and result in better segmentation performance of the model. An example of contrast enhancement using CLAHE can be seen in Figure 4.1. We used a clip limit of 2.0 and a window of size 8×8 for all of our experiments involving contrast enhancement, including the example image below.

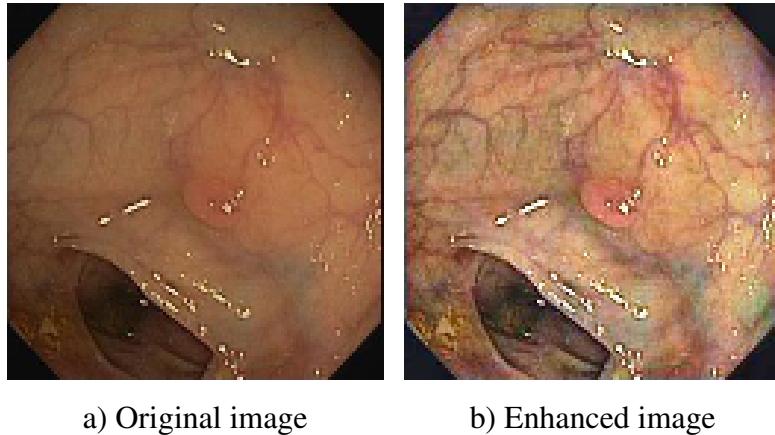


Figure 4.1: Example of contrast enhancement of an image from the CVCClinicDB dataset by using the CLAHE-algorithm. This increases the visibility of the polyp in the center of the image. Blood veins are also more visible, which can be crucial information for the classification of polyps, as the polyp often is characterized by its blood supply.

4.2 Experiment Setup

The number of epochs for all of our experiments was determined by early stopping. We use this technique to obtain an estimate of when overfitting occurs. This can be estimated by observing a declining trend in the performance accuracy of the model based on the validation set which allows us to obtain an estimate of the best performance accuracy of the model prior to the overfitting.

This was in practice accomplished by saving the parameters of the model with the highest performance accuracy measured on the validation set continuously during training in addition to the corresponding number of epochs. The training was stopped and the final model and the number of epochs were saved upon observing a trend where the validation performance degraded. The number of epochs for cross validated experiments was determined in the same manner by doing a run prior to the cross validation.

Most of our experiments were focused on five grid sizes; 2, 4, 8, 16 and 128. This is in accordance with our hypothesis presented in Section 3. The Kvasir-SEG dataset is used for generating the training and validation data, while the CVCClinicDB dataset is used for generating the test data. For each image in these datasets, we generate a corresponding grid segmentation for all the different grid sizes. The Kvasir-SEG dataset of 1000 labeled images will be used to generate 1000 images for each grid size, which would suggest that all the data generated on the basis of the Kvasir-SEG dataset would constitute 5000 images.

We do the same for our test dataset which is generated on the basis of the CVCClinicDB dataset with 612 images. These 612 images are used to generate a corresponding grid segmentation for each grid size, resulting in a test dataset of $5 \cdot 612 = 3060$ images. This is essentially the grid segmentation framework presented in Section 3.1. Furthermore, using these datasets, we use an adapted version of k-fold cross validation to our grid segmentation framework and the above requirements. Almost all of our experiments are k-fold cross validations.

4.2.1 K-fold Cross Validation

Training the network based on one single partitioning of the dataset is often suboptimal as the training would not be generalized based on all the available data. The single partition used could be a deviation, which based on the available data would imply an overly optimistic or pessimistic result. This would pose a problem when comparing performance accuracy across different datasets. k -fold cross validation can be used to accommodate this problem and gives an approximation of how well the model performs based on similar datasets.

A k -fold cross validation partitions the dataset into k number of equally sized partitionings. A single iteration of a k -fold cross validation amounts to using one partition as validation while the other partitions are used as training data. The result of the k -fold cross validation is the average of the results of all the k iterations, which ensures that the final result is close to independent from the partitioning of the data. By using this, all the data will be a part of the validation set and the training set at least once. This algorithm can be summarized by Figure 4.2, which illustrates the idea of a 5-fold cross validation.

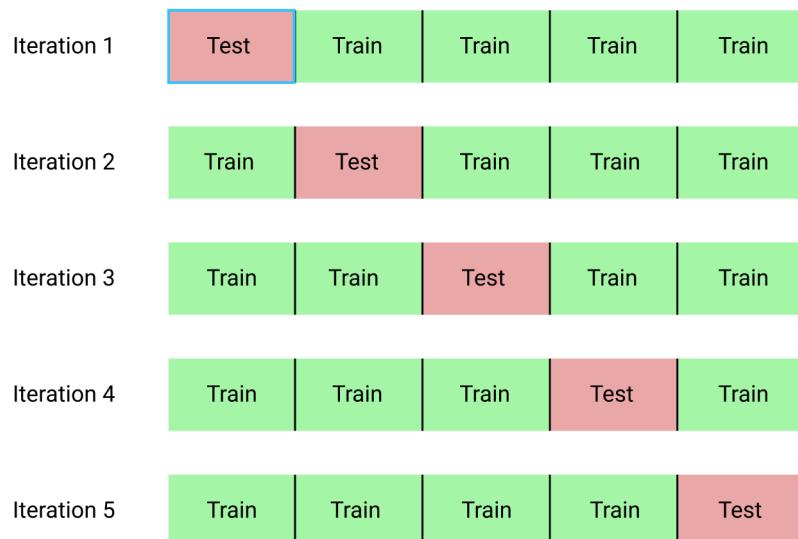


Figure 4.2: The basic idea of a 5-fold cross validation ¹. There are 5 different partitions, all of which are a part of the training set and validation set at least once. For each iteration, a model is trained and evaluated. The average of the metrics gained from the different models provides an estimate of the over all segmentation performance of the model based on the data.

4.2.1.1 Cross Validation Setup for Training on Grids

In order to ensure that each training set of the k -fold cross validation is equally diverse in terms of the amount of grids, we had to do adaptions to the ordinary cross validation algorithm described above. We perform a cross validation for each of the grid datasets, which yields a k number of

¹Credit goes to author Raheel Shaikh: <https://cutt.ly/0dLE7T4>

iterations for each of the cross validations where each iteration constitutes a training dataset and a validation dataset.

The training and validation sets for each iteration are then respectively concatenated to one single training and validation set per iteration. This is illustrated in Figure 4.3 using $k = 2$ folds. The concatenated training dataset for each iteration is then used to train a model. The average of the resulting metrics from all of the trainings and evaluations is then calculated to give an estimate of the segmentation performance based on the available data.

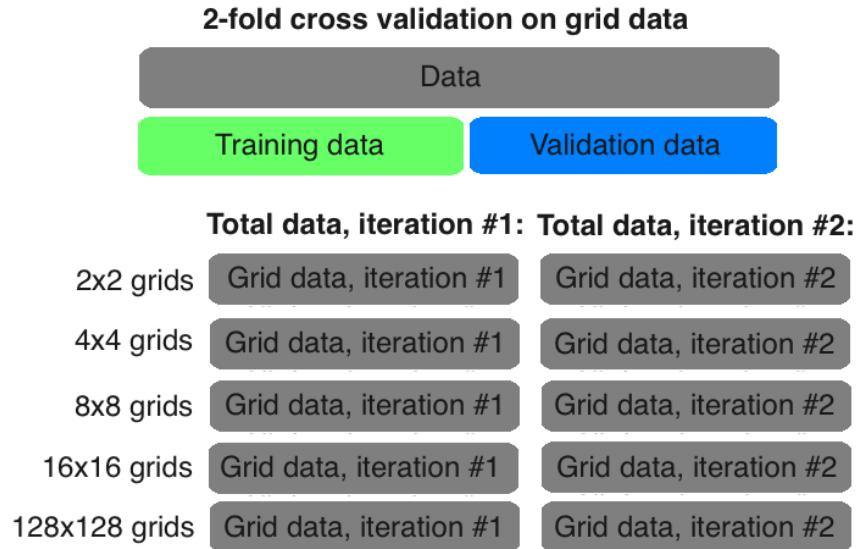


Figure 4.3: Example of how cross validation in the presence of grids have been handled in this thesis. The example is given using 2 folds and grid sizes of 2, 4, 8, 16 and 128. A cross validation is performed for each of the grid datasets and yields two iterations, where each iteration constitutes a training and validation dataset. All grid datasets corresponding to the same iterations are then concatenated such that there is one single training and validation set for each iteration. Each concatenated training set is then trained on and evaluated to produce metrics which are then averaged over to produce an estimate of the performance of the model based on the data.

As previously described, all experiments will be using data from Kvasir-SEG exclusively for training and validation while the data from CVCClinicDB will exclusively be used for testing. In order to accomplish this with the cross validation scheme presented above, we only perform the cross validation scheme for the data generated from the Kvasir-SEG dataset, which yields a single training and a validation set for each iteration. A model is trained for each of these iterations which results in k models, where k is the number of folds. These models are subsequently evaluated on the test dataset which is generated on the basis of the data from the CVCClinicDB dataset. The evaluations are then averaged to produce an estimate of how well the model performs based on the available data.

We limit the number of folds to two for all of our cross validation experiments as GANs are notoriously time consuming to train and given the time constraints of the master thesis.

4.2.2 Metrics

In order to evaluate how well a given neural network performs, different metrics are needed. The purpose of the metrics below is to evaluate the similarity between the ground truth and the segmentations generated by the machine learning models in order to give an estimate of how well the model performs.

4.2.2.1 Intersection over Union (IoU)

Intersection over Union (IoU), also known as the Jaccard index and the Jaccard similarity coefficient, is a metric for evaluating the segmentation performance of the machine learning model. It measures the degree of positive overlap between a given prediction and the corresponding ground truth for a given class. That is, it measures the degree of overlap of pixels declared as a finding for a single class for both the ground truth and a given prediction. It is illustrated in Figure 4.4 by the use of bounding boxes where all pixels within a bounding box are declared as a finding of a particular class. The concept can be expanded for multiple different classes by a metric known as meanIoU, which is the mean of the IoUs for all the classes available in the data.

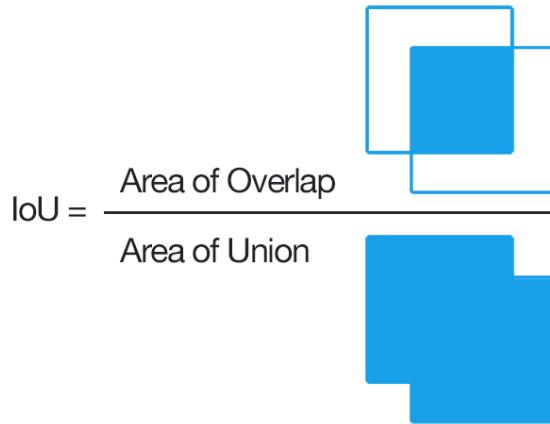


Figure 4.4: Formula for the Intersection over Union (IoU), illustrated by the use of bounding boxes². One box corresponds to the ground truth while the other is the prediction. A bounding box indicates the pixels which are marked as a finding. The IoU is the degree of overlap between predicted pixels marked as a finding and ground truth pixels marked as a finding divided by the total area of all these pixels.

The relationship between the IoU and the segmentation performance of the model based on a given prediction can be seen in Figure 4.5, again explained by the use of bounding boxes. As the figure shows, an IoU of about 0.4 suggests poor segmentation performance and indicates little overlap between the ground truth and the prediction. An IoU of about 0.73 indicates a substantial overlap and implies good performance. An IoU of about 0.92 indicates almost complete overlap between the ground truth and the prediction and suggests excellent performance.

²Credit goes to Adrian Rosebrock: <https://cutt.ly/EdLUNjv>



Figure 4.5: Example³ of how Intersection over Union (IoU) yields different results based on the overlap of the prediction and the ground truth, illustrated by the use of bounding boxes. The values range from 0 to 1, where values around 0.4 indicates poor segmentation performance, values of 0.73 indicates good performance and values of 0.92 indicates excellent performance.

4.2.2.2 Dice Coefficient

The Dice coefficient formula can be seen in Figure 4.6 and is very similar to IoU, as can be seen when comparing with the corresponding IoU formula in Figure 4.4. Though the the two metrics are very similar, the Dice coefficient tends to be a better measurement of average segmentation performance while IoU measures something closer to worst case segmentation performance. This is because errors in single pixelwise classifications tends to be penalized more by IoU than the Dice coefficient.

This is also the reason that we will include both the Dice and IoU as a measurement of our models in our experiments. Worst case performance is especially important for the case of medical applications. Particularly bad classifications could in worst case cost human lives if the model was ever put into real world use, which is why the IoU is an important metric in this case. The Dice coefficient serves to give an indication of how well the model performs in general on the testing data and is not greatly influenced by particular cases of bad classification.

The IoU and Dice coefficient are always positively correlated. If one of the metrics indicates that a given model is better than another model, the other metric would indicate the same. They are also within the same range (0 being worst, 1 being best), which makes them well suited to be reported in conjunction with each other.

³See footnote 2

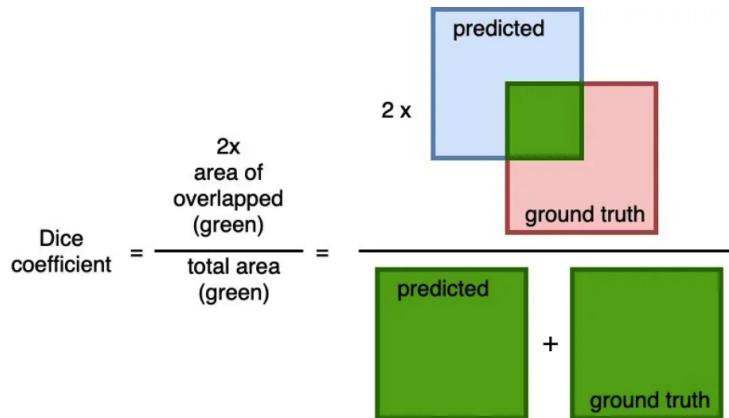


Figure 4.6: Formula for the Dice coefficient, given by the use of bounding boxes, as previous illustrations⁴. Like IoU, it has a range of 0 to 1 where 1 indicates best possible performance.

4.2.3 Basic Architecture

Based on the default architecture provided in the Pix2Pix paper, we found an architecture that provided stable learning and in general the best results for our experiments. This architecture will be used for all our GAN experiments due the computationally demanding nature of the neural networks and time limitations of the master thesis. Some experiments however will be conducted using an extended version of this architecture. The generator architecture of this Pix2Pix architecture (a U-NET architecture) will also be used for conducting corresponding U-NET experiments.

The discriminator architecture is largely composed of architectural units that are made up of a convolutional layer with filter size 4 and stride 2, followed by a batch norm layer and a LeakyReLU. Two such units were stacked with 32 and 8 filters respectively. This was then zero padded with a padding of 1 and fed to the final layer. The final layer is a convolutional layer that performs one single filtering with a stride of 1 and a filter size of 4.

The generator architecture is divided into an downsampling and an upsampling part. The architectural unit of the downsampling part is equivalent to the architectural unit described for the discriminator. The downsampling part consists of 7 of these units stacked layerwise, with the number of filters being 64, 128, 256, 512, 512, 512 and 512 respectively for each of the 7 convolutional layers. No batchnorm is applied for the first layer. The architectural unit of the upsampling part is equivalent to the architectural unit of the downsampling besides that the convolutional layer is replaced with a transposed convolutional layer for the purpose of upsampling. There are 6 of these stacked units where number of filters for each of the convolutional layers are respectively the reverse of the downsampling part. That is, 512, 512, 512, 256, 128, 64. The first three upsampling layers employ dropout [78]. The final layer is a transposed convolutional layer with the number of filters being the number of output channels for the image. In this case, we generate a gray scale segmented image, which would suggest one single channel. Skip connections are added between the downsampling and upsampling part as described in Section 2.2.3.1.3.

⁴Credit goes to Hong Jing: <https://jinglescode.github.io/datascience/2019/11/07/biomedical-image-segmentation-u-net/>

4.2.3.1 Differences from Architecture in Original Paper

Both the U-NET and Pix2Pix implementations used as a basis for our Grid U-NET and Grid-GAN models deviate in part from the original implementations described in their corresponding original papers. We use the default tensorflow implementation for both architectures which are actually modified versions of the versions presented in the original papers. The original implementations of U-NET and Pix2Pix were described in Section 2.2.3.1.3 and Section 2.2.3.2.1 respectively. We will now describe the major differences between the original implementations and the default tensorflow implementations we utilized.

The main differences in both the U-NET and Pix2Pix architectures is the absence of pooling layers, which have been replaced by a stride parameter of 2 which is implemented into the preceding convolutional layer. These two ways of downsampling are very similar, though the stride approach has multiple advantages that fit both of these models well. Pooling layers have no learnable parameters as mentioned in Section 2.2.3.1.1, but convolutional layers do. The integration of the downsampling into the convolution operation makes the downsampling to some extent learnable, which is not possible with pooling layers. Combining the operation of convolution and downsampling is also considerably cheaper than doing them separately. Another significant change is specific to the U-NET model, which in its original implementation performs cropping of the feature maps that are retrieved from the skip connections. The modified U-NET which we use for Grid U-NET and Grid-GAN performs no cropping of these feature maps but instead makes use of the full detail of the feature maps retrieved from the skip connections.

4.2.4 System Specifications

Our system specifications are given in Table 4.7. Our main limitation pertaining to the system specifications were the amount of RAM and memory of the GPU, which unfortunately restricted the size of the images we were able to work with.

Level	Category	Name	Version
Hardware	GPU	GeForce GTX 1080 Ti	
	CPU	Intel Core i5-4590 @ 3.30GHz x 4	
	Memory	15,6 GB of DDR3-1600MHz SDRAM	
Software	Operating System	Ubuntu Bionic Beaver	18.04.4
	Library	Python	3.7.6
		Tensorflow	2.0

Figure 4.7: System specifications for the machine that has been running all of our experiments.

4.3 GAN Experiments

Initial experiments with Pix2Pix and Grid-GANs were conducted using an architecture analogous to the Pix2Pix paper, but the discriminator was too powerful and resulted in the discriminator loss being fully minimized and thus no stable learning. This happened in the initial phase of the training, which could indicate that the discriminator is too sensitive to the noise that the generator generates during the initial training phase. The generated images will initially be easily classified as fake, thus resulting in the discriminator loss quickly being minimized.

There are several ways to solve this problem, all of them involves giving the generator an advantage compared to the discriminator. The generator could for instance be given an advantage by receiving gradient updates more often than the discriminator. Another approach is to simplify the architecture of the discriminator, which reduces the expressiveness of the discriminator and gives the generator an advantage. We did the latter approach which resulted in stable training and the architecture described in Section 4.2.3. This architecture will be used as a basis for all our experiments.

4.3.1 Pix2Pix

We first investigate the segmentation performance of Pix2Pix models on our datasets without using any grid information. These simple Pix2Pix experiments will be a part of our baseline in order to investigate if the different Grid-GAN models have any effect on performance.

Our initial experiment was performed with the same parameters as used in the Pix2Pix-paper, though with the new and stable architecture described in Section 4.2.3. No data augmentation or enhancement was used for this experiment. It did not yield satisfactory results as can be seen in Table 4.1. The generator and the discriminator had the same learning rate of 0.0002 and the experiment was run with 338 epochs together with a batch size of 4. Both the generator and discriminator utilized an adam optimizer with momentum parameters of $\beta_1 = 0.5$ and $\beta_2 = 0.999$ for each of the two adam optimizers.

We used the default parameters from this experiment as our starting point for further exploring the parameter space. Our final experiment for this model was performed with rotation augmentation, a generator and discriminator learning rate of 0.001 and a batch size of 16 and 270 epochs. The same parameters for the adam optimizers were used as in the original paper with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. Results can be seen in Table 4.2.

Train (Kvasir-SEG)					
Dice			IoU		
Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
0.98	0.98	0.98	0.98	0.97	0.97
Validation (Kvasir-SEG)					
Dice			IoU		
Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
0.55	0.55	0.55	0.44	0.44	0.44
Test (CVCClinicDB)					
Dice			IoU		
Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
0.43	0.35	0.39	0.33	0.35	0.30

Table 4.1: 2-fold cross validation performed on a Pix2Pix model (no grids), using the same parameters as in the paper of Pix2Pix, though with a new and stable architecture. No data enhancement was performed for this experiment.

Train (Kvasir-SEG)					
Dice			IoU		
Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
0.98	0.97	0.98	0.97	0.96	0.97
Validation (Kvasir-SEG)					
Dice			IoU		
Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
0.64	0.65	0.65	0.54	0.55	0.55
Test (CVCClinicDB)					
Dice			IoU		
Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
0.5	0.48	0.49	0.41	0.39	0.4

Table 4.2: 2-fold cross validation for Pix2Pix model with optimized parameters and rotation data augmentation. No grid information was used.

4.3.2 Grid-GAN without Data Augmentation/Enhancement

These experiments investigate the segmentation performance of a basic Grid-GAN, which in practice is the segmentation performance of applying the grid segmentation network to Pix2Pix. No data augmentation was performed besides the grid information from the grid segmentation framework and no adaptions were made to any of the core concepts of how the Pix2Pix learns.

Our first attempt was based on the same default parameters as in the Pix2Pix-paper, which implies that all the configurations of this experiment are in fact equivalent to the experiment in Section 4.3.1. Of all the experiments we conducted with this particular model, these default parameters gave the best results, which are given in Table 4.3. The learning rate of the generator and the discriminator was set to 0.0002. We performed 270 epochs together with a batch size of 4. We utilized an adam optimizer for each of the generator and discriminator with momentum parameters of $\beta_1 = 0.5$ and $\beta_2 = 0.999$.

Train (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.99	0.99	0.99	0.99	0.99	0.99
4	0.99	0.99	0.99	0.99	0.99	0.99
8	0.99	0.99	0.99	0.99	0.99	0.99
16	0.99	0.99	0.99	0.99	0.99	0.99
128	0.98	0.98	0.98	0.97	0.97	0.97
All grids:	0.99	0.99	0.99	0.99	0.99	0.99

Validation (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.79	0.8	0.8	0.71	0.72	0.71
4	0.66	0.67	0.66	0.54	0.55	0.54
8	0.58	0.62	0.6	0.46	0.5	0.48
16	0.53	0.58	0.55	0.42	0.46	0.44
128	0.50	0.54	0.52	0.40	0.43	0.41
All grids:	0.61	0.64	0.63	0.50	0.53	0.52

Test (CVCClinicDB)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.72	0.69	0.71	0.63	0.6	0.61
4	0.54	0.46	0.5	0.44	0.36	0.4
8	0.45	0.43	0.44	0.35	0.34	0.35
16	0.38	0.39	0.38	0.3	0.3	0.3
128	0.36	0.35	0.36	0.28	0.28	0.28
All grids:	0.49	0.47	0.48	0.4	0.38	0.39

Table 4.3: 2-fold cross validation results for Grid-GAN using default Pix2Pix parameters without any data enhancement besides grids. For this particular model, these parameters gave the best results. Grid segmentation for grid size 128 is a normal pixel-level segmentation, as the image input size is 128.

Further experiments for this model did unfortunately not yield better results. We used these default Pix2Pix parameters as a basis for exploring the parameter space. Another experiment can be seen in Table 4.4 which was performed with 219 epochs, a learning rate of 0.003 for both the generator and discriminator and a batch size of 16. The same adam optimizers were also used as in previous experiments.

Train (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.99	0.99	0.99	0.99	0.99	0.99
4	0.99	0.99	0.99	0.99	0.99	0.99
8	0.99	0.99	0.99	0.99	0.99	0.99
16	0.99	0.99	0.99	0.99	0.99	0.99
128	0.97	0.98	0.97	0.95	0.96	0.95
All grids:	0.99	0.99	0.99	0.98	0.99	0.98

Validation (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.8	0.8	0.8	0.72	0.72	0.72
4	0.65	0.64	0.64	0.53	0.52	0.53
8	0.55	0.58	0.57	0.44	0.46	0.45
16	0.49	0.54	0.51	0.39	0.43	0.41
128	0.48	0.51	0.49	0.38	0.4	0.39
All grids:	0.59	0.61	0.6	0.49	0.51	0.5

Test (CVCClinicDB)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.72	0.7	0.71	0.63	0.61	0.62
4	0.51	0.45	0.48	0.41	0.36	0.38
8	0.43	0.37	0.4	0.33	0.29	0.32
16	0.36	0.32	0.34	0.27	0.25	0.26
128	0.34	0.3	0.32	0.26	0.23	0.25
All grids:	0.47	0.43	0.45	0.38	0.35	0.37

Table 4.4: 2-fold cross validation results for Grid-GAN without any data augmentation. This experiment performed worse than the previous. It ran with 219 epochs, a learning rate of 0.003 for both the generator and discriminator, a batch size of 16 and the same adam optimizers as previous runs.

4.3.3 Grid-GAN with Rotation Augmented Data

We can observe from previous experiments that no data enhancement or augmentation besides grids did not yield satisfactory results. We decided from this point to investigate how the amount and quality of the data affect segmentation performance for the grid segmentations, which could be investigated by data enhancement and data augmentation techniques. Our first approach was to augment the data by rotation. We increased the amount of images for the training set by performing rotations of the original data, which we subsequently added to the training dataset. We performed rotation augmentation with angles 90° , 180° and 270° in addition to the original image, in accordance with the theory presented in Section 4.1.3.1.

As with previous experiments, our initial experiment was performed using the default parameters presented in the Pix2Pix paper. The generator and the discriminator had the same learning rate of 0.0002 and the experiment was run with 105 epochs together with a batch size of 4. Both the generator and discriminator utilized an adam optimizer with momentum parameters of $\beta_1 = 0.5$ and $\beta_2 = 0.999$ for each of the two adam optimizers. Results can be seen in Table 4.6. This type of data augmentation made the amount of training data 4 times larger and resulted in an increase in test performance from 0.48 (in Dice coefficient, for all grids) to 0.59 as can be seen when comparing Table 4.6 and Table 4.3.

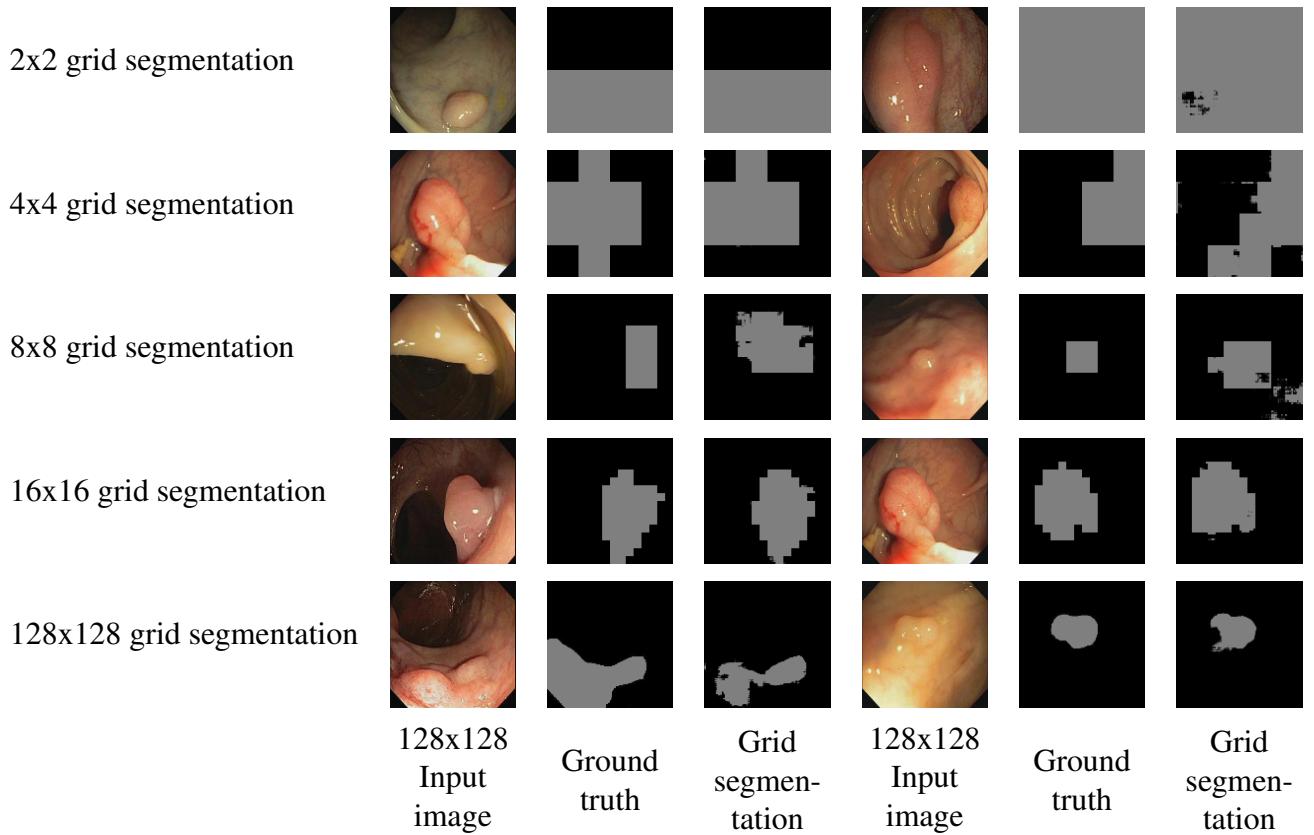


Table 4.5: Segmentation performance of our cross validated model in Table 4.6 for randomly selected images from our test dataset (the CVCClinicDB dataset). Gray indicates a finding, while black indicates no finding.

Train (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.99	0.99	0.99	0.99	0.99	0.99
4	0.99	0.99	0.99	0.99	0.99	0.99
8	0.99	0.99	0.99	0.99	0.99	0.99
16	0.99	0.99	0.99	0.98	0.98	0.98
128	0.97	0.96	0.97	0.94	0.94	0.94
All grids:	0.99	0.98	0.99	0.98	0.98	0.98
Validation (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.83	0.82	0.83	0.76	0.75	0.76
4	0.72	0.73	0.73	0.62	0.63	0.62
8	0.68	0.69	0.69	0.57	0.59	0.58
16	0.66	0.67	0.66	0.55	0.56	0.56
128	0.64	0.64	0.64	0.53	0.54	0.54
All grids:	0.71	0.71	0.71	0.61	0.62	0.61
Test (CVCClinicDB)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.77	0.74	0.76	0.69	0.66	0.67
4	0.63	0.61	0.62	0.52	0.51	0.52
8	0.58	0.56	0.57	0.47	0.45	0.46
16	0.55	0.49	0.52	0.44	0.39	0.42
128	0.55	0.48	0.51	0.44	0.38	0.41
All grids:	0.61	0.58	0.59	0.51	0.48	0.5

Table 4.6: 2-fold cross validation of Grid-GAN ran with rotation augmentation using default Pix2Pix parameters. This run is equivalent to the run in Table 4.3, but with rotation augmentation. As observed for the previous model, these parameters performed the best.

Again, we explored the parameter space on the basis of the default parameters used in the previous run. The experiment shown in Table 4.7 concluded our research for this particular model. It did not perform better than our initial experiment and was performed using a learning rate of 0.004 for both the generator and discriminator, 164 epochs and a batch size of 16. The same adam optimizer was used for both the generator and discriminator as for Pix2Pix.

Train (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.99	0.99	0.99	0.99	0.99	0.99
4	0.98	0.99	0.99	0.97	0.98	0.98
8	0.97	0.98	0.98	0.96	0.97	0.97
16	0.96	0.97	0.97	0.94	0.96	0.95
128	0.95	0.95	0.95	0.9	0.92	0.91
All grids:	0.97	0.98	0.97	0.95	0.97	0.96
Validation (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.84	0.83	0.83	0.77	0.76	0.76
4	0.72	0.73	0.72	0.61	0.62	0.62
8	0.66	0.64	0.65	0.55	0.54	0.55
16	0.65	0.59	0.62	0.54	0.49	0.52
128	0.63	0.6	0.61	0.52	0.5	0.51
All grids:	0.7	0.68	0.69	0.6	0.59	0.59
Test (CVCClinicDB)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.77	0.75	0.76	0.68	0.66	0.67
4	0.63	0.61	0.62	0.52	0.51	0.52
8	0.54	0.46	0.5	0.43	0.38	0.4
16	0.51	0.39	0.45	0.4	0.31	0.36
128	0.5	0.39	0.45	0.4	0.32	0.36
All grids:	0.59	0.52	0.55	0.49	0.44	0.46

Table 4.7: 2-fold cross validation of Grid-GAN with rotation data augmentation, ran with 164 epochs, a learning rate of 0.004 for the generator and discriminator and a batch size of 16. This experiment performed worse than our experiment with default Pix2Pix parameters.

4.3.4 Grid-GAN with Contrast Enhanced Data

We enhance all images for all datasets using the CLAHE algorithm as described in Section 4.1.3.2. We then investigate if this leads to better segmentation performance than for the corresponding runs without contrast enhancement (which were performed in Section 4.3.2). The experiments show little improvement and we get comparable results to the models that were performed with no data enhancement/augmentation, which can be seen by comparing Table 4.3 and Table 4.8. There may be several possible reasons for this result. The result may indicate that the model is actually learning to contrast enhance the images by itself as a part of its effort to segment the image. The lack of improvement can also be due to a big parameter space that has not been sufficiently explored. The learning rate of the generator and discriminator was set to 0.001 for this particular experiment. We ran the experiment using 224 epochs together with a batch size of 16. Both the generator and

discriminator utilized an adam optimizer with momentum parameters of $\beta_1 = 0.5$ and $\beta_2 = 0.999$ for each of the two adam optimizers. Results can be seen in Table 4.8.

Train (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.99	0.99	0.99	0.99	0.99	0.99
4	0.98	0.99	0.99	0.96	0.99	0.98
8	0.95	0.99	0.97	0.92	0.99	0.96
16	0.93	0.99	0.96	0.89	0.99	0.94
128	0.93	0.98	0.95	0.88	0.96	0.92
All grids:	0.96	0.99	0.97	0.93	0.99	0.96

Validation (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.81	0.81	0.81	0.72	0.72	0.72
4	0.63	0.65	0.64	0.63	0.53	0.52
8	0.55	0.58	0.57	0.55	0.46	0.44
16	0.5	0.55	0.52	0.5	0.43	0.41
128	0.48	0.52	0.5	0.48	0.41	0.39
All grids:	0.59	0.62	0.61	0.48	0.51	0.49

Test (CVCClinicDB)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.72	0.73	0.73	0.62	0.64	0.63
4	0.48	0.55	0.52	0.37	0.44	0.41
8	0.42	0.46	0.44	0.32	0.35	0.34
16	0.34	0.41	0.37	0.25	0.31	0.28
128	0.34	0.39	0.37	0.26	0.3	0.28
All grids:	0.46	0.51	0.49	0.37	0.41	0.39

Table 4.8: 2-fold cross validation of Grid-GAN performed on contrast enhanced data. Ran with 224 epochs, a learning rate of 0.001 for generator and discriminator and a batch size of 16.

4.3.5 Grid-GAN with All Grid Sizes and Rotation Augmented Data

As stated in the last paragraph of Section 3.1.1, we decided to limit our experiments to be based on five different grid sizes; 2, 4, 8, 16, 128. Parts of our motivation were based on the idea that restricting the number of grid sizes would serve better for a proof of concept of our grid segmentation framework, as the segmentations for the higher grid sizes are very similar. However, it would be interesting to see what effect there would be for the segmentation performance if all grid sizes were included.

The following experiment was performed with rotation augmentation, a learning rate of 0.004 for both the generator and discriminator, 170 epochs and a batch size of 16. It did not achieve any jump in segmentation performance compared to the previous corresponding experiments with a limited number of grid sizes given in Section 4.3.3. Results are given in Table 4.9.

Train (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.98	0.96	0.97	0.97	0.94	0.96
4	0.95	0.93	0.94	0.92	0.89	0.9
8	0.91	0.89	0.9	0.86	0.84	0.85
16	0.89	0.88	0.89	0.83	0.82	0.83
32	0.88	0.88	0.88	0.81	0.81	0.81
64	0.87	0.86	0.87	0.79	0.79	0.79
128	0.87	0.88	0.88	0.8	0.81	0.81
All grids:	0.91	0.9	0.9	0.86	0.84	0.85
Validation (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.83	0.83	0.83	0.75	0.76	0.76
4	0.69	0.73	0.71	0.58	0.62	0.6
8	0.6	0.63	0.62	0.5	0.53	0.51
16	0.6	0.59	0.59	0.5	0.49	0.49
32	0.59	0.62	0.6	0.49	0.51	0.5
64	0.59	0.6	0.6	0.49	0.5	0.49
128	0.59	0.62	0.6	0.49	0.51	0.5
All grids:	0.64	0.66	0.65	0.54	0.56	0.55
Test (CVCClinicDB)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.78	0.75	0.77	0.69	0.66	0.68
4	0.63	0.59	0.61	0.52	0.49	0.5
8	0.49	0.44	0.46	0.4	0.36	0.38
16	0.48	0.39	0.43	0.38	0.31	0.35
32	0.46	0.43	0.45	0.36	0.35	0.35
64	0.46	0.43	0.44	0.36	0.34	0.35
128	0.47	0.44	0.46	0.38	0.36	0.37
All grids:	0.54	0.5	0.52	0.44	0.41	0.43

Table 4.9: 2-fold cross validation of Grid-GAN ran for all grids using rotation augmentation. The experiment was performed using a learning rate of 0.004 for both the generator and discriminator, 170 epochs and a batch size of 16.

4.3.6 Grid-GAN with Loss Modifications

Our previous experiments investigated how we could manipulate our datasets to increase segmentation performance. We will now turn to investigate how the model itself can be modified in order to fit the grid segmentation framework better and possibly enhance segmentation performance. We will be using rotation augmentation for all future Grid-GAN experiments unless stated otherwise, as previous experiments obtained better segmentation performance with this type of augmentation. We could observe from previous experiments that the Grid-GAN struggled to learn large grid sizes the most. We hypothesized that it could be because the loss function that is being learned by the model does not learn to depend sufficiently on grid size. Even though the Grid-GAN has the ability to learn its loss function during training, it might be the case that we have to push the learning of the loss in the direction of depending sufficiently upon grid size in order to better incorporate grid information into the model.

The generator loss was explained in Section 2.2.3.2.1 and incorporates an adversarial loss $\mathcal{L}_{cGAN}(G, D)$ and a 11-loss $\mathcal{L}_{L1}(G)$.

$$\begin{aligned}\mathcal{L}_{cGAN}(G, D) &= \mathbb{E}_{x,y}[\log(D(x,y))] + \mathbb{E}_x[\log(1 - D(x, G(x)))] \\ \mathcal{L}_{L1}(G) &= \mathbb{E}_{x,y}[||y - G(x)||_1]\end{aligned}$$

For this experiment, we propose a direct approach to incorporate the grid size into the loss of the generator by adding a grid loss in addition to the adversarial loss and the 11-loss. Our first proposal was adding a mean cross entropy grid loss $\mathcal{L}_{grid-CE}(G)$ which measures the cross entropy between each grid cell and averages them. The final generator loss is summarized in Equation 4.1. We kept the original weight of the 11-loss to $\lambda = 100$ and mainly experimented with the weight λ_2 of the mean cross entropy grid loss in our final experiments.

$$\mathcal{L}_G(G, D) = \arg \min_G \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G) + \lambda_2 \mathcal{L}_{grid-CE}(G) \quad (4.1)$$

This Grid-GAN model included a new parameter λ_2 that had to be explored, which is the weight of the mean cross entropy grid loss that was added to the generator loss. For this parameter, we based our parameter search on the weight of the 11-loss, which in the default Pix2Pix architecture is set to 100. The other parameters, such as learning rate, batch size etc were explored on the basis of default Pix2Pix parameters as in previous experiments.

Our initial choice of value of 100 for the weight of the grid loss turned out to be unfortunate for the learning of the generator, as the discriminator loss became minimized within the first epochs. We experimented with different weights for the average cross entropy grid loss and found that lower values had a tendency to perform best in general. We ran an experiment where we halved the value of λ_2 to 50, which resulted in a Dice coefficient at 0.63 at its highest. We furthermore lowered the parameter down to 10, which gave the results seen in Table 4.10. This experiment was in addition performed with a learning rate of 0.004 and a batch size of 16 and 117 epochs. We then performed another run for a lower weight of $\lambda_2 = 3$ which did not result in a considerable jump in segmentation performance.

Grid size	Single experiment with $\lambda_2 = 10$							
	Train		Val		Test		Dice	IoU
	Dice	IoU	Dice	IoU	Dice	IoU		
2	0.99	0.98	0.77	0.67	0.74	0.65		
4	0.99	0.98	0.66	0.54	0.62	0.51		
8	0.98	0.97	0.70	0.6	0.53	0.42		
16	0.96	0.94	0.62	0.51	0.5	0.38		
128	0.94	0.91	0.64	0.55	0.46	0.35		
All grids:	0.97	0.96	0.68	0.58	0.57	0.46		

Table 4.10: Single run for Grid-GAN with a mean cross entropy grid loss with weight parameter $\lambda_2 = 10$. Kvasir-SEG for generating the training and validation sets and CVCClinicDB for generating the test set, as with all other experiments. Rotation augmented data used.

Our final cross validated run with optimized parameters can be found in Table 4.12, along with examples on how it performed on the test dataset in Table 4.11. This experiment was run with a learning rate of 0.00001 for the generator and discriminator, 73 epochs and a batch size of 4.

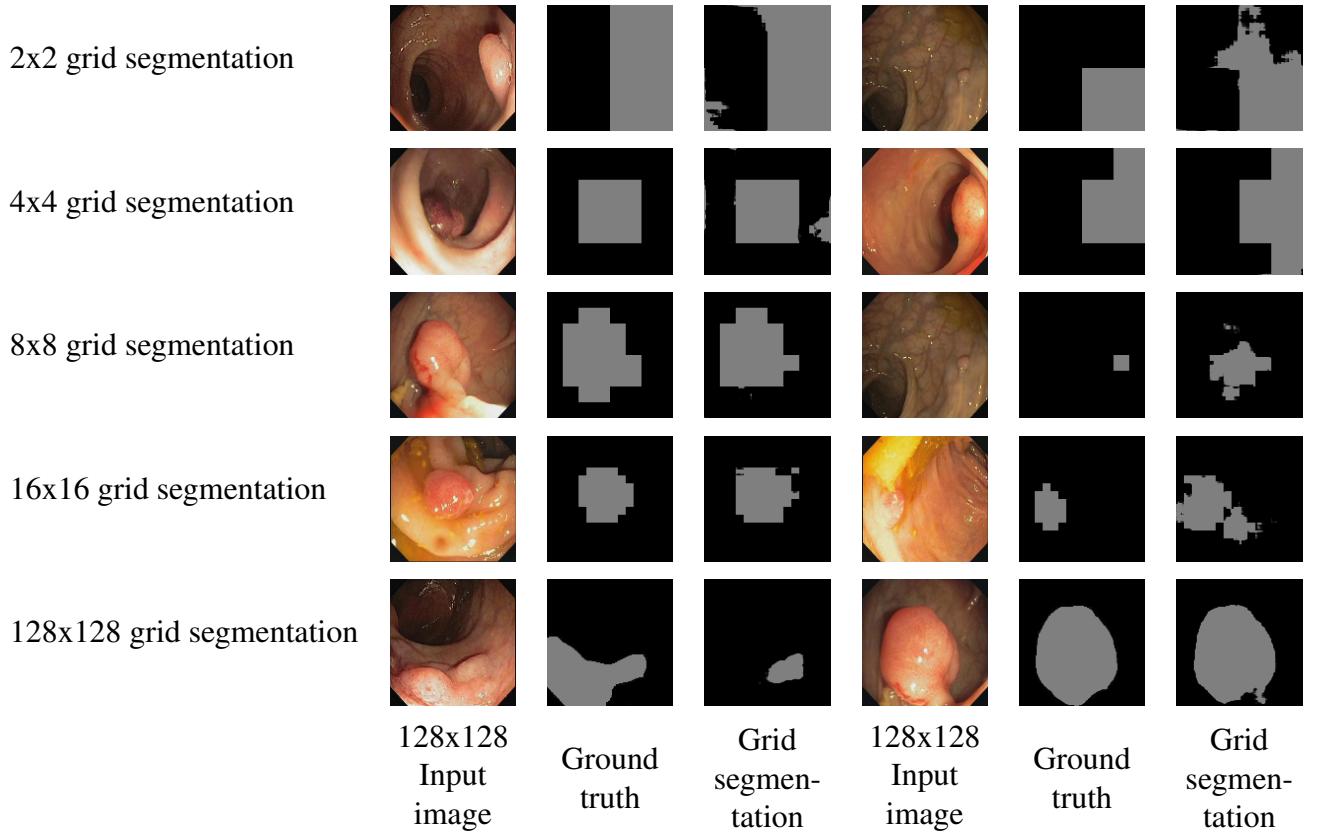


Table 4.11: Performance of our cross validated model in Table 4.12 for randomly selected images from our test dataset (the CVCClinicDB dataset). Gray indicates a finding, while black indicates no finding.

Train (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.99	0.99	0.99	0.99	0.99	0.99
4	0.98	0.99	0.99	0.98	0.99	0.98
8	0.97	0.99	0.98	0.95	0.99	0.97
16	0.95	0.98	0.97	0.92	0.98	0.95
128	0.93	0.96	0.94	0.88	0.93	0.9
All grids:	0.96	0.98	0.97	0.94	0.97	0.96
Validation (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.83	0.82	0.83	0.76	0.75	0.76
4	0.72	0.73	0.73	0.62	0.63	0.62
8	0.69	0.69	0.69	0.58	0.58	0.58
16	0.68	0.64	0.66	0.57	0.54	0.56
128	0.65	0.64	0.64	0.54	0.54	0.54
All grids:	0.72	0.7	0.71	0.62	0.61	0.61
Test (CVCClinicDB)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.78	0.74	0.76	0.7	0.66	0.68
4	0.66	0.61	0.63	0.55	0.51	0.53
8	0.62	0.55	0.59	0.51	0.45	0.48
16	0.58	0.48	0.53	0.47	0.39	0.43
128	0.56	0.5	0.53	0.46	0.41	0.42
All grids:	0.64	0.58	0.61	0.54	0.48	0.51

Table 4.12: 2-fold cross validation of Grid-GAN with a mean cross entropy grid loss and rotation augmentation. Performed with a learning rate of 0.0001 and 73 epochs together with a batch size of 4. This was our best performing experiment with the mean cross entropy grid loss.

We also tried with a higher learning rate, which gave a worse result. This experiment was performed with a learning rate of 0.002 for the generator and discriminator, grid loss weight of $\lambda_2 = 10$, a batch size of 16 and 195 epochs. Results can be seen in Table 4.13.

Train (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.99	0.99	0.99	0.99	0.99	0.99
4	0.99	0.99	0.99	0.99	0.99	0.99
8	0.99	0.99	0.99	0.99	0.99	0.99
16	0.99	0.99	0.99	0.98	0.98	0.98
128	0.97	0.97	0.97	0.94	0.95	0.95
All grids:	0.99	0.99	0.99	0.98	0.98	0.98

Validation (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.81	0.83	0.82	0.73	0.76	0.75
4	0.69	0.71	0.7	0.59	0.61	0.6
8	0.66	0.66	0.66	0.56	0.57	0.56
16	0.64	0.63	0.63	0.53	0.53	0.53
128	0.61	0.62	0.62	0.51	0.52	0.52
All grids:	0.68	0.69	0.69	0.59	0.6	0.59

Test (CVCClinicDB)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.74	0.72	0.73	0.66	0.64	0.65
4	0.57	0.55	0.56	0.47	0.46	0.47
8	0.53	0.46	0.5	0.43	0.38	0.41
16	0.49	0.41	0.45	0.40	0.33	0.37
128	0.47	0.41	0.44	0.39	0.33	0.36
All grids:	0.56	0.51	0.53	0.47	0.43	0.45

Table 4.13: 2-fold cross validation of Grid-GAN with a mean cross entropy grid loss and rotation augmentation. Performed with a grid loss weight of $\lambda_2 = 10$, a learning rate of 0.002 for discriminator and generator, 195 epochs and a batch size of 16. This experiment gave worse results than our previous experiment.

4.3.7 Grid-GAN with Upsampling Skip Connections

We originally hypothesized that it may be possible to learn information pertaining to different grid segmentations through the different layers in the upsampling part of the network. This is because the different layers can represent information with regards to different receptive fields, which could be interpreted as grids. The first upsampling layer may for instance represent segmentation information from the lowest grid size, while the last layers represent information from higher grid size segmentations. This idea was explained in the introductory chapter of chapter 3. We further hypothesize based on this idea that the original U-NET architecture of the generator may be sub-optimal for learning on grids. It may actually be disadvantageous to propagate the information through all the upsampling layers as they also have the task of representing different grids. In the

case for the first upsampling layers representing the lower grid sizes, this information should rather be directly forwarded to the last layer instead of being further transformed through the remaining upsampling layers. This could theoretically aid the learning of the different grid segmentations.

The U-NET architecture (the generator of the Grid-GAN) has skip connections from its downsampling part to its upsampling part. We propose a new U-NET architecture where additional skip connections are added to improve the learning of grids. Skip connections are added from every layer in the upsampling part to the last layer of the upsampling part. This is in practice done by zero padding and concatenating the input of the last layer with the outputs of the previous upsampling layers. The outputs are zero padded to match the dimension of the input. The zero padding of the outputs and the subsequent transformation can be interpreted as just a bigger upsampling. Equivalent to previous experiments, we first performed a run with default Pix2Pix parameters for our model. The experiment was run with 82 epochs together with a batch size of 4 and a learning rate of 0.0002 for both the generator and the discriminator. Both the generator and discriminator utilized an adam optimizer with momentum parameters of $\beta_1 = 0.5$ and $\beta_2 = 0.999$ for each of the two adam optimizers. Results are given in Table 4.15 along with examples of how it performed on individual images from test set in 4.14.

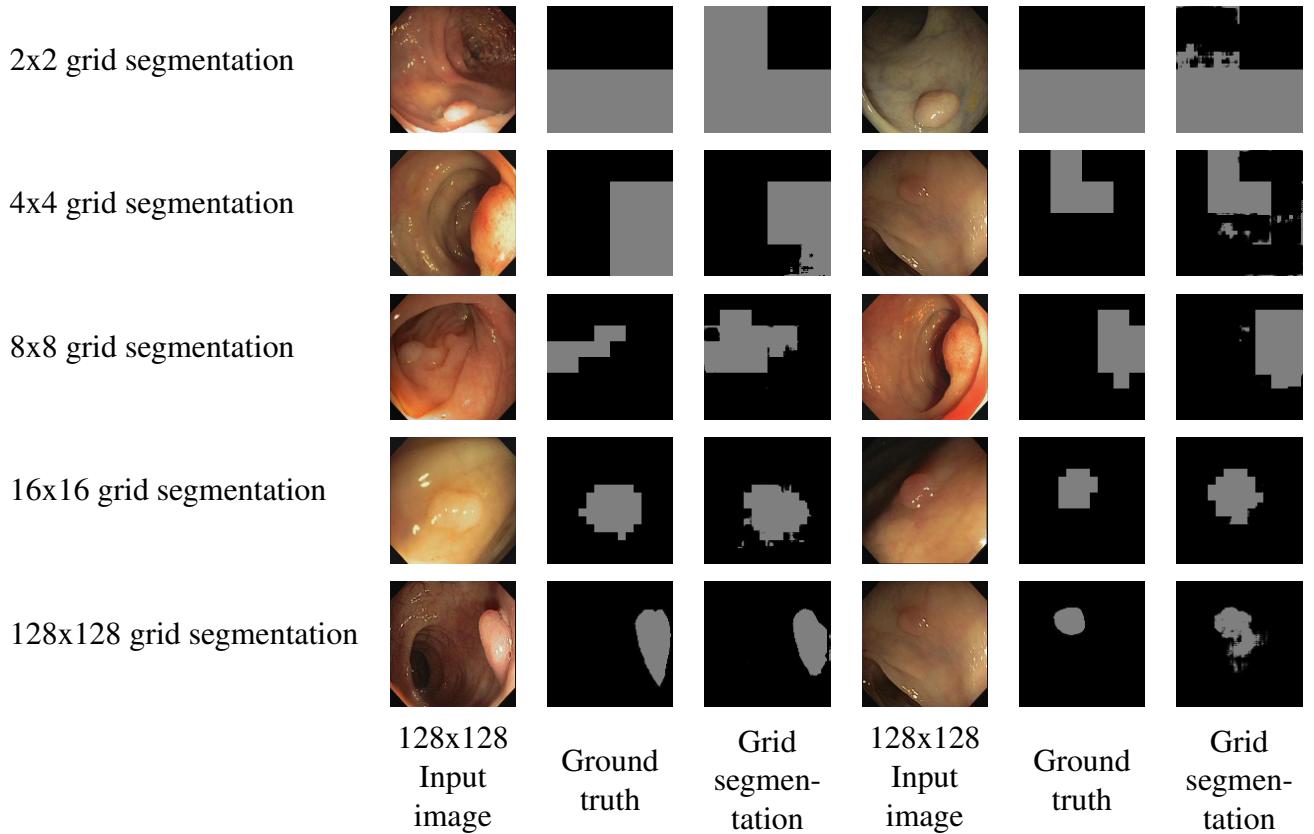


Table 4.14: Performance of our cross validated model in Table 4.15 for randomly selected images from our test dataset (the CVCClinicDB dataset). Gray indicates a finding, while black indicates no finding.

Train (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.99	0.99	0.99	0.99	0.99	0.99
4	0.99	0.99	0.99	0.99	0.99	0.99
8	0.98	0.99	0.99	0.98	0.99	0.98
16	0.97	0.99	0.98	0.95	0.98	0.97
128	0.95	0.96	0.96	0.92	0.93	0.92
All grids:	0.98	0.98	0.98	0.96	0.98	0.97
Validation (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.83	0.83	0.83	0.76	0.76	0.76
4	0.74	0.73	0.74	0.64	0.63	0.63
8	0.69	0.69	0.69	0.58	0.59	0.59
16	0.67	0.67	0.67	0.56	0.56	0.56
128	0.65	0.64	0.65	0.54	0.54	0.54
All grids:	0.72	0.71	0.72	0.62	0.62	0.62
Test (CVCClinicDB)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.78	0.75	0.77	0.7	0.67	0.68
4	0.68	0.62	0.65	0.57	0.52	0.54
8	0.62	0.56	0.59	0.51	0.46	0.48
16	0.57	0.51	0.54	0.46	0.41	0.43
128	0.57	0.49	0.53	0.46	0.39	0.43
All grids:	0.64	0.58	0.61	0.54	0.49	0.51

Table 4.15: 2-fold cross validation of Grid-GAN with upsampling skip connections. Ran using rotation augmentation and the same parameters as default Pix2Pix with a learning rate of 0.0002, 82 epochs and a batch size of 4.

We explored the parameter space based on the default Pix2Pix parameters, as done in the majority of our previous experiments. We performed an experiment using 131 epochs, a learning rate of 0.008 for the generator and discriminator and a batch size of 16, which performed marginally worse than the previous experiment. We also did an experiment where we increased the learning rate to 0.03 and performed about 1000 epochs. This resulted in a performance of 0.49 in Dice and 0.41 in IoU, which was worse than our default configuration.

Train (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.99	0.99	0.99	0.98	0.98	0.98
4	0.98	0.98	0.98	0.97	0.97	0.97
8	0.98	0.97	0.97	0.96	0.95	0.95
16	0.97	0.95	0.96	0.94	0.92	0.93
128	0.95	0.94	0.94	0.91	0.89	0.9
All grids:	0.97	0.97	0.97	0.95	0.95	0.95
Validation (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.8	0.84	0.82	0.72	0.77	0.75
4	0.69	0.73	0.71	0.59	0.62	0.6
8	0.65	0.69	0.67	0.54	0.58	0.56
16	0.62	0.65	0.64	0.52	0.54	0.53
128	0.61	0.64	0.62	0.5	0.53	0.52
All grids:	0.67	0.71	0.69	0.57	0.61	0.59
Test (CVCClinicDB)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.75	0.75	0.75	0.66	0.67	0.66
4	0.61	0.6	0.6	0.51	0.5	0.5
8	0.55	0.56	0.55	0.44	0.46	0.45
16	0.5	0.49	0.49	0.4	0.4	0.4
128	0.49	0.5	0.5	0.39	0.41	0.41
All grids:	0.58	0.48	0.58	0.58	0.49	0.48

Table 4.16: 2-fold cross validation of Grid-GAN with upsampling skip connections, ran with 131 epochs, 0.008 in learning rate for generator and discriminator, a batch size of 16 and rotation augmented data. This experiment performed marginally worse than our previous.

4.3.8 Final Small Scale Experiments with 256x256 Images

The amount of data generated by our grid segmentation framework made it difficult to train models based on 256x256 images given our available equipment. This lead to our initial decision of focusing on 128x128 images. In order to identify possibilities of future work, we decided as a last effort to scale back experiments to meet our memory requirements and run some of them on 256x256 images. It is worth noting that these experiments provided some of the best results seen so far in this thesis, which for future work would make a thorough investigation for larger images even more interesting.

Due to time limitations, these experiments were not run with any cross validations. We performed an experiment with an ordinary Pix2Pix with no grids, the default architecture provided by tensorflow, a learning rate of 0.002, batch size of 16 and rotation augmented data, where we ran 1000 epochs. Kvasir-SEG was used for training and validation, while CVCClinicDB was used for testing. Best results were achieved by epoch 512, which can be seen in Table 4.17.

Train (Kvasir-SEG)		Validation (Kvasir-SEG)		Test (CVCClinicDB)	
Dice	IoU	Dice	IoU	Dice	IoU
0.98	0.97	0.74	0.64	0.59	0.49

Table 4.17: Single Pix2Pix experiment on 256x256 images using a learning rate of 0.002, batch size of 16 and 512 epochs.

4.4 U-NET Baseline Experiments

We want to investigate the basis of our motivation for the Grid-GAN, which hinge on the hypothesis that the adversarial learning introduced by the model could provide an advantage for learning to perform the segmentations. The theory behind this hypothesis was described in detail in the motivation section of Grid-GANs (section 3.2.1). We investigate this hypothesis by providing corresponding U-NET experiments for our Grid-GAN variants. The corresponding architectures for the generator of the Grid-GAN (which is a U-NET architecture) will be used for our U-NET experiments. The experiments will be for the most part be focused on the Grid-GAN approaches that provided the most promising results.

4.4.1 U-NET

The authors of the Kvasir-SEG dataset performed experiments with a basic U-NET, which gave a performance of 0.7147 in Dice and 0.43 in meanIoU [44]. These experiments were based on images of size 256x256 while our experiments have for the most part been based on 128x128 images. We will in spite of this use the results as part of our baseline when comparing with other models while also performing our own experiments based on the parameters from this model.

The following experiment is performed with the same parameters as for the optimized model given in this paper, but we adapt the experiment to 128x128 images in order to compare with our models. We use our chosen architecture described in Section 4.2.3 and a learning rate of 0.0001, a batch size of 16 and 120 epochs. We also performed the experiment with rotation augmented data, which has proven to be best among the data augmentation techniques we have investigated. Results can be seen in Table 4.18.

Train (Kvasir-SEG)						
Dice			IoU			
Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg	
0.96	0.95	0.95	0.93	0.93	0.93	

Validation (Kvasir-SEG)						
Dice			IoU			
Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg	
0.62	0.63	0.62	0.51	0.52	0.52	

Test (CVCClinicDB)						
Dice			IoU			
Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg	
0.48	0.45	0.47	0.39	0.36	0.37	

Table 4.18: 2-fold cross validation of ordinary U-NET without any grids, but with rotation augmented data. Performed using the same parameters from the Kvasir-SEG dataset paper with a learning rate of 0.0001, 120 epochs and a batch size of 16.

4.4.2 Grid U-NET with Rotation Augmented Data

These Grid U-NET experiments are baseline experiments for the Grid-GAN experiments seen in Section 4.3.3. We tried to optimize parameters on the basis the parameters used for the Grid-GAN experiments. Below are two experiments, one performed with a learning rate of 0.0004, batch size of 4 and 98 epochs (given in Table 4.19) and another experiment performed with a learning rate of 0.003, 226 epochs and a batch size of 16 (given in Table 4.20).

The first U-NET experiment (Table 4.19) performed the best of the two experiments, though it gave worse segmentation performance than for the corresponding Grid-GAN model. This may be due to our initial hypothesis which stated that the GAN models might perform better due to their ability to learn their loss function, but we acknowledge that these results can also be due to a vast parameter space that was not sufficiently explored.

Train (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.99	0.99	0.99	0.99	0.99	0.99
4	0.99	0.99	0.99	0.99	0.98	0.99
8	0.99	0.99	0.99	0.98	0.98	0.98
16	0.98	0.98	0.98	0.97	0.97	0.97
128	0.95	0.94	0.95	0.91	0.9	0.91
All grids:	0.98	0.98	0.98	0.97	0.97	0.97

Validation (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.81	0.82	0.82	0.74	0.75	0.74
4	0.7	0.71	0.7	0.59	0.6	0.6
8	0.64	0.66	0.65	0.53	0.56	0.55
16	0.61	0.64	0.62	0.51	0.54	0.52
128	0.58	0.6	0.59	0.48	0.51	0.49
All grids:	0.67	0.69	0.68	0.57	0.59	0.58

Test (CVCClinicDB)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.73	0.73	0.73	0.64	0.65	0.65
4	0.57	0.57	0.57	0.48	0.47	0.47
8	0.48	0.5	0.49	0.39	0.41	0.4
16	0.45	0.46	0.46	0.36	0.37	0.37
128	0.45	0.46	0.45	0.36	0.37	0.37
All grids:	0.54	0.54	0.54	0.45	0.45	0.45

Table 4.19: 2-fold cross validation of Grid U-NET ran with rotation augmentation. Performed with a learning rate of 0.0004, 98 epochs and a batch size of 4.

Train (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.99	0.99	0.99	0.99	0.99	0.99
4	0.99	0.99	0.99	0.99	0.99	0.99
8	0.99	0.99	0.99	0.99	0.99	0.99
16	0.99	0.99	0.99	0.99	0.99	0.99
128	0.96	0.96	0.96	0.93	0.94	0.93
All grids:	0.99	0.99	0.99	0.98	0.98	0.98

Validation (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.81	0.82	0.81	0.73	0.75	0.74
4	0.66	0.71	0.69	0.56	0.61	0.59
8	0.61	0.65	0.63	0.51	0.55	0.53
16	0.57	0.63	0.6	0.48	0.53	0.51
128	0.56	0.6	0.58	0.47	0.51	0.49
All grids:	0.64	0.68	0.66	0.55	0.59	0.57

Test (CVCClinicDB)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.73	0.71	0.72	0.65	0.63	0.64
4	0.52	0.54	0.53	0.43	0.45	0.44
8	0.48	0.44	0.46	0.4	0.36	0.38
16	0.42	0.41	0.42	0.34	0.34	0.34
128	0.43	0.4	0.42	0.35	0.33	0.34
All grids:	0.52	0.5	0.51	0.43	0.42	0.43

Table 4.20: 2-fold cross validation of Grid U-NET ran with rotation augmentation. Performed with a learning rate of 0.003, 226 epochs and a batch size of 16.

4.4.3 Grid U-NET with Contrast Enhanced Data

We performed a corresponding Grid U-NET experiment based on the Grid-GAN experiments with contrast enhanced data seen in Section 4.3.4. This Grid U-NET experiment can be seen in Table 4.21. It was performed with 283 epochs, a learning rate of 0.002 and a batch size of 16. The experiment did not show a substantial gain in segmentation performance, which we also observed for the corresponding Grid-GAN experiment. We believe the results for both the U-NET and the Grid-GAN in this case can be explained by the same factors. It is possible that the model already learns to implicit perform a contrast enhancement as a part of learning the segmentation, or that the results are due to a parameter space which has not been sufficiently explored.

Train (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.99	0.99	0.99	0.99	0.99	0.99
4	0.99	0.99	0.99	0.99	0.99	0.99
8	0.99	0.99	0.99	0.99	0.99	0.99
16	0.99	0.99	0.99	0.99	0.99	0.99
128	0.98	0.96	0.97	0.96	0.94	0.95
All grids:	0.99	0.99	0.99	0.99	0.98	0.98

Validation (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.76	0.78	0.77	0.66	0.69	0.68
4	0.59	0.61	0.6	0.46	0.48	0.47
8	0.47	0.5	0.49	0.37	0.39	0.38
16	0.4	0.45	0.42	0.31	0.35	0.33
128	0.38	0.43	0.41	0.3	0.34	0.32
All grids:	0.52	0.55	0.54	0.42	0.45	0.43

Test (CVCClinicDB)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.67	0.71	0.69	0.57	0.61	0.59
4	0.45	0.48	0.46	0.35	0.37	0.36
8	0.37	0.39	0.38	0.28	0.29	0.29
16	0.31	0.33	0.32	0.23	0.25	0.24
128	0.29	0.31	0.3	0.22	0.24	0.23
All grids:	0.42	0.44	0.43	0.33	0.35	0.34

Table 4.21: 2-fold cross validation of Grid U-NET ran on contrast enhanced images, A learning rate of 0.002 was used as well as a batch size of 16. The experiment was run with 283 epochs.

4.4.4 Grid U-NET with Loss Modifications

The loss inherent to the U-NET architecture is a cross entropy loss. In this experiment, we extend it by adding the same mean cross entropy grid loss used in the Grid-GAN variant presented in Section 4.3.6. We also perform the experiment with rotation augmentation, which have shown promising results in earlier experiments. Our final experiment can be seen in Table 4.22, which was performed using a learning rate of 0.0002, 85 epochs and a batch size of 4. This gave marginally better results than the original model without the mean cross entropy grid loss. Based on these results, it would be difficult to conclude that this loss actually improved segmentation performance. We believe more experiments are needed in order to make a definite conclusion, which would have to be left for future work.

Train (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.99	0.99	0.99	0.99	0.99	0.99
4	0.99	0.99	0.99	0.98	0.98	0.98
8	0.98	0.98	0.98	0.98	0.97	0.97
16	0.98	0.97	0.97	0.96	0.96	0.96
128	0.94	0.93	0.94	0.9	0.89	0.89
All grids:	0.98	0.97	0.97	0.96	0.96	0.96

Validation (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.82	0.81	0.82	0.75	0.74	0.74
4	0.68	0.71	0.69	0.57	0.6	0.59
8	0.64	0.65	0.64	0.53	0.55	0.54
16	0.62	0.62	0.62	0.52	0.52	0.52
128	0.58	0.6	0.59	0.48	0.5	0.49
All grids:	0.67	0.68	0.67	0.57	0.58	0.58

Test (CVCClinicDB)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.76	0.68	0.72	0.68	0.6	0.64
4	0.59	0.51	0.55	0.49	0.42	0.46
8	0.54	0.43	0.49	0.44	0.35	0.4
16	0.52	0.42	0.47	0.41	0.33	0.37
128	0.5	0.42	0.46	0.4	0.34	0.37
All grids:	0.58	0.49	0.54	0.48	0.41	0.45

Table 4.22: 2-fold cross validation of Grid U-NET with a mean cross entropy grid loss. Performed using a learning rate of 0.0002, a batch size of 4 and 85 epochs.

4.4.5 Grid U-NET with Upsampling Skip Connections

This experiment is essentially the generator of the Grid-GAN seen in Section 4.3.7 which is run and trained as a U-NET with a learning rate of 0.0002, batch size of 4, 131 epochs and rotation augmented data. The results can be seen in Table 4.23. It performed a bit worse than the corresponding Grid-GAN model and had very similar performance to previous experiments with only marginal differences.

Train (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.99	0.99	0.99	0.99	0.99	0.99
4	0.99	0.99	0.99	0.98	0.98	0.98
8	0.98	0.98	0.98	0.98	0.97	0.97
16	0.98	0.97	0.97	0.96	0.95	0.96
128	0.94	0.93	0.94	0.9	0.89	0.9
All grids:	0.98	0.97	0.97	0.96	0.96	0.96

Validation (Kvasir-SEG)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.81	0.81	0.81	0.73	0.73	0.73
4	0.68	0.7	0.69	0.58	0.6	0.59
8	0.64	0.66	0.65	0.54	0.56	0.55
16	0.62	0.63	0.62	0.51	0.53	0.52
128	0.57	0.59	0.58	0.48	0.5	0.49
All grids:	0.67	0.68	0.67	0.57	0.58	0.58

Test (CVCClinicDB)						
Grid size	Dice			IoU		
	Fold 1	Fold 2	Avg	Fold 1	Fold 2	Avg
2	0.75	0.69	0.72	0.67	0.62	0.64
4	0.59	0.56	0.57	0.49	0.47	0.48
8	0.55	0.48	0.51	0.44	0.4	0.42
16	0.52	0.44	0.48	0.42	0.36	0.39
128	0.49	0.42	0.46	0.4	0.35	0.37
All grids:	0.58	0.52	0.55	0.48	0.44	0.46

Table 4.23: 2-fold cross validation of Grid U-NET with upsampling skip connections performed with a learning rate of 0.0002, batch size of 4 and 131 epochs.

4.5 Summary

This chapter relates to our second and third research objectives. First, we apply the grid segmentation framework to U-NET and Pix2Pix and investigate various data augmentation techniques to improve segmentation performance. We later modify the architecture and the loss of the models in attempts to make the grid segmentation networks better adapted to the grid segmentation framework, which according to our theories in Section 4.3.7 and 4.3.6 could promote better segmentation performance.

Our top three best performing grid segmentation models had the ability to perform at least as well as our corresponding U-NET and Pix2Pix experiments without grids. This can be observed for all of our three best performing Grid-GAN models seen in Table 4.24, and the same observations can be observed for corresponding Grid U-NET experiments seen in Table 4.25. The Grid-GAN provided slightly better segmentation results than our best Pix2Pix experiment, but we acknowledge that this may be due to a parameter space that was not fully investigated.

For 128x128 images, Grid-GAN based networks provided our best results which were also the networks where most of our research was focused. None of our baseline Grid U-NET experiments outperformed the corresponding Grid-GAN experiments. The Grid-GAN variant with upsampling skip connections and rotation augmentation provided the best results, though with a very small margin as can be seen in Table 4.24.

- **USSC:** Grid-GAN with upsampling skip connections and rotation augmented data
- **MCE Loss:** Grid-GAN with mean cross entropy grid loss and rotation augmented data
- **Normal:** Grid-GAN with only rotation augmented data
- **No Grids:** Normal Pix2Pix with only rotation augmented data

Top three Grid-GAN models compared with our best Pix2Pix experiment

Grid size	USSC		MCE Loss		Normal		No Grids	
	Dice	IoU	Dice	IoU	Dice	IoU	Dice	IoU
2	0.77	0.68	0.76	0.68	0.76	0.67	-	-
4	0.65	0.54	0.63	0.53	0.62	0.52	-	-
8	0.59	0.48	0.59	0.48	0.57	0.46	-	-
16	0.54	0.43	0.53	0.43	0.52	0.42	-	-
128	0.53	0.43	0.53	0.42	0.51	0.41	0.49	0.4
All grids:	0.61	0.51	0.61	0.51	0.59	0.5	-	-

Table 4.24: Average test performance results from cross validation experiments of our best varieties of Grid-GANs, ran on 128x128 images. The test dataset was obtained from a separate dataset called CVCclinicDB, while training was performed with the Kvasir-SEG dataset.

The best results of all our experiments were seen with Pix2Pix on 256x256 images, but we unfortunately did not have the computational resources nor the time to fully investigate this approach for our grid segmentation networks, which had to be left for future work.

As can be seen in Table 4.25, all of our top three Grid U-NET experiments performed somewhat worse than corresponding Grid-GAN experiments.

- **USSC**: Grid U-NET with upsampling skip connections and rotation augmented data
- **MCE Loss**: Grid U-NET with mean cross entropy grid loss and rotation augmented data
- **Normal**: Grid U-NET with only rotation augmented data
- **No Grids**: Normal U-NET with only rotation augmented data

Grid size	USSC		MCE Loss		Normal		No Grids	
	Dice	IoU	Dice	IoU	Dice	IoU	Dice	IoU
2	0.72	0.64	0.72	0.64	0.73	0.65	-	-
4	0.57	0.48	0.55	0.46	0.57	0.47	-	-
8	0.51	0.42	0.49	0.4	0.49	0.4	-	-
16	0.48	0.39	0.47	0.37	0.46	0.37	-	-
128	0.46	0.37	0.46	0.37	0.45	0.37	0.47	0.37
All grids:	0.55	0.46	0.54	0.45	0.54	0.45	-	-

Table 4.25: Average test performance results from cross validation experiments of our best varieties of Grid U-NETs, ran on 128x128 images. The test dataset was obtained from a separate dataset called CVCClinicDB.

Chapter 5

Conclusion

This chapter summarizes the work presented in this thesis and presents perspectives relevant for future work.

5.1 Summary and Contributions

Colorectal cancer is one of the most common causes of cancer related deaths, where early detection is crucial for survival. Screening is manually performed by doctors, which includes both the detection and localization of polyps, commonly known as precursors to gastrointestinal cancer. A major problem for the manual screening of patients is human errors where polyps could for instance not be detected by the doctor. Similar problems are observed when manually determining attributes of the polyp, such as size and borders, which further could be important to remove the polyp or determine malignancy. These problems associated with manual screening motivate the use of automated segmentation, which is an area of research where machine learning have excelled over the recent years. The U-NET and its various variants have provided state-of-the-art segmentation for medical applications since their introduction. To address the problems with current manual screening of the gastrointestinal tract, we proposed in Section 1.2 the development of new variants of neural networks which we will compare with current-state-of-the-art neural networks for segmentation.

The main objective of the developed networks was to answer our research question:

Can we improve the segmentation performance of a segmentation model by learning to segment within several degrees of segmentation precisions?

Our research question was based on the hypothesis that features used to create less precise segmentations could aid the training of pixel-level segmentations. We obtained different degrees of segmentation precisions by learning to segment within grids, while the grid size was specified as an additional input to the segmentation model and acted as a parameter for controlling the segmentation precision. To answer our research question, we developed three objectives to help guide us to a solution. Each objective builds on the one that came before it, and through the completion of each objective, we are better suited to make a final conclusion.

Objective 1: *Research and develop a framework for learning a grid segmentation mapping that takes an image and a grid size and segments the image within the specified grid size. The segmentation mapping should also be able to perform a normal pixel-level segmentation in addition to these grid segmentations, which would correspond to a grid size equal to the image size. The resulting framework should be able to work with several state-of-the-art neural networks.*

This objective is supported by our development of the grid segmentation framework and the corresponding pipeline. The grid segmentation framework enables machine learning algorithms to perform segmentations with degrees of precisions, which is realized by learning to segment within different grids. The framework included an approach for the encoding of grids as well as methods for generating the various segmentations within specific grids. Our framework was successfully implemented and worked for several state-of-the-art neural networks.

Objective 2: *Apply the grid segmentation framework to state-of-the-art neural networks for segmentation. Investigate how the amount and quality of the data affects the learning of the grid segmentations. Compare the results with corresponding state-of-the-art neural networks without grids.*

We applied the grid segmentation framework to Pix2Pix [41] and U-NET models [94]. All results considered, our models showed particularly good results for segmentations of lower precisions at the cost of less precision, while segmentations of higher precisions proved more difficult. Normal pixel-level segmentations proved to be the hardest task, where we obtained at least as good performance as current state-of-the-art neural networks. The precisions obtained for lower segmentations often allowed for good approximations to be obtained when higher precisions failed. We also investigated how the amount and quality of the data affected our models by using rotation augmented data and contrast enhanced data. We obtained our best results with rotation augmentation, which offered the most significant improvements. We only observed small improvements for contrast enhancement. These results were also confirmed by our baseline experiments.

Even though several of our Grid-GANs did exceed the segmentation performance of our Pix2Pix experiments, the difference in performance was marginal and went from 0.49 to 0.53 in Dice at best. Similar observations were made for our baseline experiments with Grid U-NET, which did not give better performance than corresponding U-NET models. The lack of improvement of segmentation performance may be due to several reasons. GAN models are notorious for long training time and it may be that the parameter space was not sufficiently explored. Another very likely alternative is that there was no improvement, because the conceptual idea for the effective learning of features when learning grids (described in Section 3.2.1) did not conform with how the networks learned in practice.

Both our Pix2Pix and Grid-GAN models performed better than corresponding U-NET and Grid U-NET architectures in our experiments, which may be due to our initial hypothesis that the Grid-GAN and Pix2Pix would perform better due to their ability to learn their loss function during training. The results may also be explained by our choice to particularly focus on the Grid-GAN models, which lead to fewer experiments being performed for the Grid U-NETs and thus fewer parts of the parameter space being explored.

Objective 3: *Investigate how the learning of these networks could be fully adapted and modified to the grid segmentation framework in order to enhance segmentation performance. Again, compare these results with corresponding state-of-the-art neural networks without grids.*

This objective is supported by the theories we developed for how our grid segmentation networks could learn in a manner that could improve segmentation performance and the following modifications we performed to accommodate the learning process. We developed a theory that the receptive fields of the different layers could represent different grids, in which adding skip connections between all upsampling layers to the last layer could allow grid information to be propagated directly to the last layer. We also incorporated the grid size into the loss of our models in order to try to push the models towards better learning of grids. Generally, our experiments with these models obtained nearly the same results as previous experiments.

For the variant with upsampling skip connections, this lack of improvement may be due to several reasons. Besides the reasons already mentioned in objective 2, it may be that there is no improvement because the U-NET architecture already propagates the grid segmentation information efficiently through the layers to produce the final segmentation. Given our theory for how the segmentations could be learned, the Grid U-NET could learn to disregard the higher level grid details from the skip connections coming from the down-sampling part, depending on the input grid size. In other words, the grid information is simply upsampled during the subsequent propagations through the layers. This would imply that the learning of the Grid-GAN already is optimal and could explain why there is no improvement for this type of modification.

For the loss modification, we saw that it generally required a weight of low value to be able to learn well. Given how little the model improved and the low weight that was required, it is possible that the grid loss function was unlearned during the learning of the loss function, which could explain why there was no gain in segmentation performance. In addition to all the scenarios described for the various models, the lack of observed gain in performance could be due to a parameter space that was not sufficiently explored. GANs, which were the main focus of this thesis, were really time consuming to train.

We will now summarize the answer to our research question, which was largely addressed in the answer to the research objectives above. Our research question was based on the hypothesis that learning several degrees of precision in the segmentation maps would enhance segmentation performance. This idea was motivated by the hypothesis that features pertaining to segmentations of less precision could build upon features of higher precision, including the performance of normal pixel-level segmentations. Although some of our Grid-GAN experiments showed marginally better results than Pix2Pix, our baseline experiments with Grid U-NET and U-NETs did not. Based on these results, we were not able to confirm the hypothesis that learning several degrees of precision in the segmentation maps would enhance segmentation performance. We described several possible reasons, which could be that the effective learning of features that we hypothesized would be possible in Section 4.3.7 and 3.2.1 did not occur in practice. There is also the possibility that the parameter space was not sufficiently explored due to the long training of the Grid-GANs and the limited time for our thesis. Regardless of cause, further research of this topic would probably benefit from looking at how our models actually learn to capture grid information, in order to further

accommodate the learning according to our hypothesis. This could for instance be researched by the use of saliency maps.

Despite this conclusion, the grid segmentation networks showed particularly promising results for their ability to segment with less precision and approximate where a given finding is situated. If the network fails to perform a normal pixel-level segmentation of a particular image, a lower grid size could be given to the network and often mark an area of interest where the particular finding is roughly situated. This is in stark contrast to even the best neural networks, which might not fail as gracefully and only provide a few sporadic markings of the object. These networks would not have the same ability to approximate the location of a given finding when a normal segmentation fails, but the grid segmentation networks we developed do have this ability.

5.2 Future Work

Over the course of this thesis, we demonstrated how machine learning algorithms could learn to segment within several degrees of segmentation precisions. We showed how it is possible to obtain quite good performance for lower degrees of segmentation precisions and a pixel-level segmentation performance that was at least as good as experiments with current state-of-the-art networks. However, there is still potential for improvements, including further research of our hypothesis regarding that the lower degrees of segmentation precisions could aid pixel-level segmentation precisions. Based on the research we conducted during the work of this thesis, we have several proposals for future work, many of which we believe could be very interesting to research.

- **Performing experiments with larger images**

This is perhaps the most promising alternative for future work. We unfortunately did not have the time or the computational resources available to fully investigate performance for our grid segmentation networks on images larger than 128x128. The few restricted experiments we performed gave the best results seen so far in this thesis, which makes this particularly interesting to investigate, especially if applied for the Grid-GAN, which provided the best results in general for 128x128 images.

- **CycleGAN implementation of Grid-GAN**

It would be interesting to investigate if unsupervised learning by CycleGAN would be able to find an underlying representation in the data that makes the network learn a better internal representation of grids. CycleGAN has shown promising results for image segmentation [94].

- **Investigating alternative grid encodings**

Our approach have mainly been focused on conditioning on a grid size that is encoded as an image which is given as an input together with the input image. This is in accordance with how conditional GANs have traditionally been implemented, but it can be a bit unfortunate

in our case. The grid encoding is in essence a large vector that encodes a very limited set of possible grid sizes, which is an inefficient representation that could impact the learning.

An alternative approach could be to implement a conditioning similar to the approach seen in Dual Conditional GANs [77]. The condition (grid size) could be encoded to a small vector that is concatenated with the result of the downsampling part which is then directly fed to the first upsampling layer in the generator/U-NET architecture. This alternative approach for conditioning requires fewer parameters and could facilitate learning. It has already been successfully implemented and utilized for several image based tasks.

- **Investigating alternative mappings**

We mainly investigated one particular mapping during the work of this thesis but there are other mappings which would be interesting to investigate as well. One approach that we did not have time to investigate sufficiently was a mapping where each grid cell corresponds to a single pixel, which differs from our initial approach where a grid cell is in essence a collection of pixels. The mapping can be illustrated in Figure 5.1.

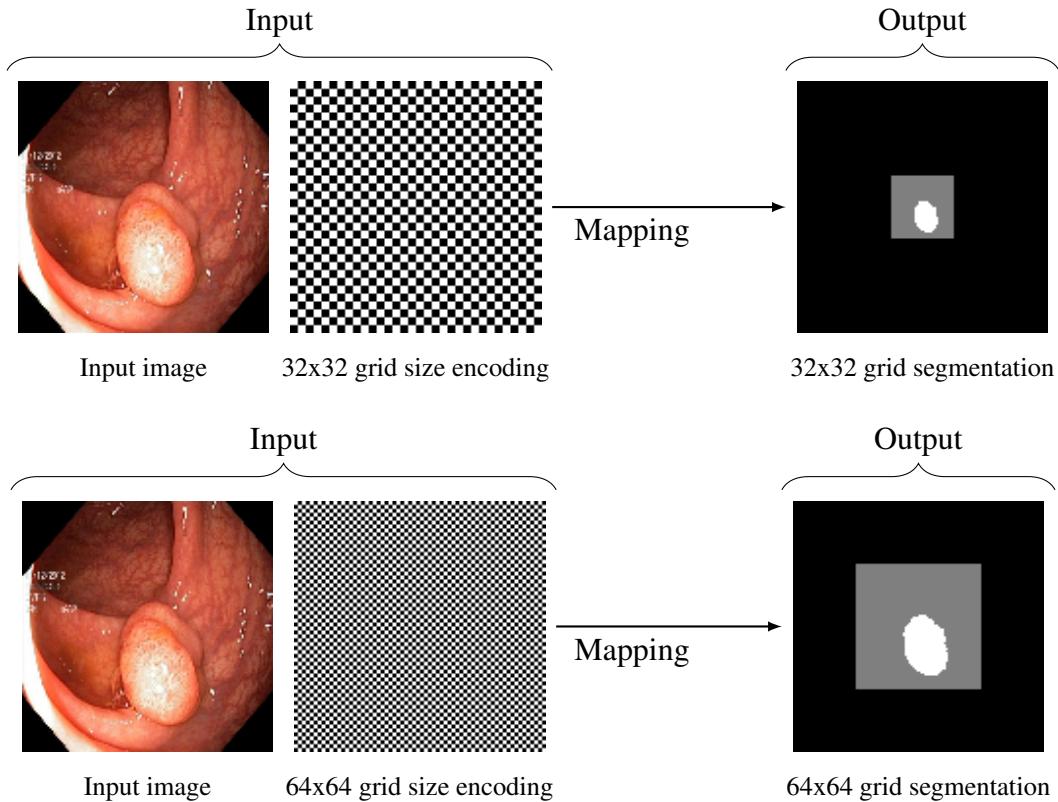


Figure 5.1: Segmentation examples with grid size 32 and 64 for the Grid-GAN variant where each grid cell is a pixel value. The black color in the output indicates that it is outside of the segmentation. Gray is no finding. White is a finding.

- **Investigating alternative grid losses**

We experimented with some grid losses in this thesis with the goal of pushing the learning of the loss in the direction of learning grids better. We only had time to investigate a few variants of these losses and it would be interesting to investigate if other losses could have given a better effect for the learning of the grid segmentations.

- **Investigating alternative architectures and parameters**

GANs are notoriously time consuming to train, which have restricted our ability to investigate the parameter space and different architectures given the computing power we have available for the master thesis. Further exploring the parameter space and different architectures with different amounts of layers and neurons is set as future work due to this limitation.

- **Run experiments on the Kvasir-Capsule dataset**

During the work of this thesis, we helped develop the Kvasir-Capsule dataset, but we unfortunately did not have time to perform any experiments with it. We ultimately chose to rather prioritize segmentation based datasets of one single class for our experiments, which we considered to be the best approach for a proof of concept for our models. The Kvasir-Capsule dataset could nevertheless be interesting to investigate for future work, as it contains several other categories of gastrointestinal diseases besides polyps.

Bibliography

- [1] Deep-Insights AI. “Covid-19 classifier.” In: (Apr. 25, 2020). Fetched 07/18/2020. URL: <https://labs.deep-insights.ai/>.
- [2] Ali Alammari et al. “Classification of Ulcerative Colitis Severity in Colonoscopy Videos using CNN.” In: *Proceedings of the 9th International Conference on Information Management and Engineering*. ACM. 2017, pp. 139–144.
- [3] L. A. Alexandre, N. Nobre, and J. Casteleiro. “Color and Position versus Texture Features for Endoscopic Polyp Detection.” In: *2008 International Conference on BioMedical Engineering and Informatics*. Vol. 2. 2008, pp. 38–42.
- [4] Mohammad Ali Armin. “Automated visibility map from colonoscopy video to support clinical diagnosis and improve the quality of colonoscopy.” In: (2016), p. 29.
- [5] Indriani Astono et al. “Optimisation of 2D U-Net Model Components for Automatic Prostate Segmentation on MRI.” In: *Applied Sciences* 10 (Apr. 2020), p. 2601. DOI: 10.3390/app10072601.
- [6] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. *Learning long-term dependencies with gradient descent is difficult*. Vol. 5. Fetched 06/12/2019. IEEE Transactions on Neural Networks, Mar. 1994, pp. 157–166.
- [7] 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. ISSN: 0895-6111. DOI: <https://doi.org/10.1016/j.compmedimag.2015.02.007>. URL: <http://www.sciencedirect.com/science/article/pii/S0895611115000567>.
- [8] Roisin Bevan and Matthew D Rutter. “Colorectal Cancer Screening—Who, How, and When?” In: (Sept. 5, 2017). Fetched 06/12/2019. URL: <https://pdfs.semanticscholar.org/70be/3576447b47f727da4df5d6dae3aa7ac86154.pdf>.
- [9] Leonardo Bottacci et al. “Artificial neural networks applied to outcome prediction for colorectal cancer patients in separate institutions.” In: *The Lancet* 350.9076 (1997), pp. 469–472. ISSN: 0140-6736. DOI: [https://doi.org/10.1016/S0140-6736\(96\)11196-X](https://doi.org/10.1016/S0140-6736(96)11196-X). URL: <http://www.sciencedirect.com/science/article/pii/S014067369611196X>.
- [10] Peter Boyle and Michael Langman. “ABC of colorectal cancer: Epidemiology.” In: (2000), pp. 1–2. URL: <https://doi.org/10.1136/bmj.321.7264.805>.
- [11] Dmitrii Bychkov et al. “Deep learning based tissue analysis predicts outcome in colorectal cancer.” In: *Scientific Reports* 8.1 (2018), pp. 1–11. ISSN: 20452322. DOI: 10.1038/s41598-018-21758-3. URL: <http://dx.doi.org/10.1038/s41598-018-21758-3>.

- [12] H. Chen, Q. Xiao, and X. Yin. “Generating Music Algorithm with Deep Convolutional Generative Adversarial Networks.” In: *2019 IEEE 2nd International Conference on Electronics Technology (ICET)*. 2019, pp. 576–580.
- [13] Junyoung Chung et al. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. 2014. arXiv: 1412.3555 [cs.NE].
- [14] Taco S. Cohen et al. *Spherical CNNs*. 2018. arXiv: 1801.10130 [cs.LG].
- [15] D. E. Comer et al. “Computing as a Discipline.” In: *Commun. ACM* 32.1 (Jan. 1989), pp. 9–23. ISSN: 0001-0782. DOI: 10.1145/63238.63239. URL: <https://doi.org/10.1145/63238.63239>.
- [16] Guido Costamagna et al. “A prospective trial comparing small bowel radiographs and video capsule endoscopy for suspected small bowel disease.” In: *Gastroenterology* 123.4 (2002), pp. 999–1005.
- [17] G. Cybenko. “Approximation by superpositions of a sigmoidal function.” In: *Mathematics of Control, Signals, and Systems (MCSS)* 2.4 (Dec. 1, 1989), pp. 303–314. ISSN: 0932-4194. DOI: 10.1007/BF02551274. URL: <http://dx.doi.org/10.1007/BF02551274>.
- [18] Emmanuel Gbenga Dada et al. “Machine learning for email spam filtering: review, approaches and open research problems.” In: *Helijon* 5.6 (2019), e01802. ISSN: 2405-8440. DOI: <https://doi.org/10.1016/j.heliyon.2019.e01802>. URL: <http://www.sciencedirect.com/science/article/pii/S2405844018353404>.
- [19] Robin Delaine-Smith et al. “Transglutaminase-2 Mediates the Biomechanical Properties of the Colorectal Cancer Tissue Microenvironment that Contribute to Disease Progression.” In: *Cancers* 11 (May 2019), p. 701. DOI: 10.3390/cancers11050701.
- [20] Jeff Donahue et al. “Long-term Recurrent Convolutional Networks for Visual Recognition and Description.” In: (Nov. 30, 2015). Fetched 06/13/2019. URL: <https://arxiv.org/pdf/1411.4389.pdf>.
- [21] Xavier Dray et al. “CAD-CAP: une base de données française à vocation internationale, pour le développement et la validation d’outils de diagnostic assisté par ordinateur en vidéocapsule endoscopique du grêle.” In: *Actes des JFHOD 2018*. Ed. by Thieme. Vol. 50. Endoscopy 03. Paris, France, Mar. 2018, p. 316. URL: <https://hal.archives-ouvertes.fr/hal-01808657>.
- [22] Babak Ehteshami Bejnordi et al. “Diagnostic Assessment of Deep Learning Algorithms for Detection of Lymph Node Metastases in Women With Breast Cancer.” In: *JAMA* 318.22 (Dec. 2017), pp. 2199–2210. ISSN: 0098-7484. DOI: 10.1001/jama.2017.14585.
- [23] Robert A Enns et al. “Clinical practice guidelines for the use of video capsule endoscopy.” In: *Gastroenterology* 152.3 (2017), pp. 497–514.
- [24] Theodoros Evgeniou and Massimiliano Pontil. “Support Vector Machines: Theory and Applications.” In: vol. 2049. Jan. 2001, pp. 249–257. DOI: 10.1007/3-540-44673-7_12.
- [25] Jacques Ferlay et al. “Cancer incidence and mortality worldwide: sources, methods and major patterns in GLOBOCAN 2012.” In: (Sept. 13, 2014). Fetched 06/12/2019. URL: <https://www.ncbi.nlm.nih.gov/pubmed/25220842/>.
- [26] S. I. Gallant. “Perceptron-based learning algorithms.” In: *IEEE Transactions on Neural Networks* 1.2 (1990), pp. 179–191.

- [27] Lianli Gao et al. *Learning in High-Dimensional Multimedia Data: The State of the Art*. 2017. arXiv: 1707.02683 [cs.CV].
- [28] Xabier García-Albéniz et al. “Effectiveness of screening colonoscopy to prevent colorectal cancer among Medicare beneficiaries aged 70–79 years: a prospective observational study.” In: (Sept. 27, 2016). Fetched 06/12/2019. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5417337/>.
- [29] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].
- [30] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. Chap. 10. URL: <http://www.deeplearningbook.org>.
- [31] Eric Greenstein. “Japanese-to-English Machine Translation Using Recurrent Neural Networks.” In: 2015.
- [32] Beverley Greenwood-Van Meerveld, Anthony C Johnson, and David Grundy. “Gastrointestinal physiology and function.” In: *Gastrointestinal Pharmacology*. Springer, 2017, pp. 1–16.
- [33] European Colorectal Cancer Screening Guidelines Working Group. “European guidelines for quality assurance in colorectal cancer screening and diagnosis: Overview and introduction to the full Supplement publication.” In: (June 26, 2012). Fetched 06/12/2019. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4482205/>.
- [34] A. Gupta and A. Joshi. “Speech Recognition Using Artificial Neural Network.” In: *2018 International Conference on Communication and Signal Processing (ICCSP)*. 2018, pp. 0068–0071.
- [35] H.A. Haenssle et al. “Man against machine: diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists.” In: *Annals of Oncology* 29.8 (2018). Immune-related pathologic response criteria, pp. 1836–1842. ISSN: 0923-7534. DOI: <https://doi.org/10.1093/annonc/mdy166>. URL: <http://www.sciencedirect.com/science/article/pii/S0923753419341055>.
- [36] Philipp Harzig, Moritz Einfalt, and Rainer Lienhart. “Automatic Disease Detection and Report Generation for Gastrointestinal Tract Examination.” In: *Proceedings of the 27th ACM International Conference on Multimedia*. MM ’19. Nice, France: Association for Computing Machinery, 2019, pp. 2573–2577. ISBN: 9781450368896. DOI: 10.1145/3343031.3356066. URL: <https://doi.org/10.1145/3343031.3356066>.
- [37] David G Hewett, Charles J Kahi, and Douglas K Rex. “Efficacy and effectiveness of colonoscopy: how do we bridge the gap?” In: *Gastrointestinal Endoscopy Clinics* 20.4 (2010), pp. 673–684. DOI: 10.1016/j.giec.2010.07.011.
- [38] Toshiaki Hirasawa et al. “113 application of artificial intelligence using convolutional neural network for detecting gastric cancer in endoscopic images.” In: *Gastrointestinal Endoscopy* 87.6 (2018), AB51.
- [39] Xun Huang et al. *Multimodal Unsupervised Image-to-Image Translation*. 2018. arXiv: 1804.04732 [cs.CV].
- [40] Haruhiro Inoue et al. “The Paris endoscopic classification of superficial neoplastic lesions: esophagus, stomach, and colon: November 30 to December 1, 2002.” In: *Gastrointestinal endoscopy* 58 6 Suppl (2003), pp. 3–43.

- [41] Phillip Isola et al. *Image-to-Image Translation with Conditional Adversarial Networks*. 2016. arXiv: 1611.07004 [cs.CV].
- [42] Debesh Jha et al. *DoubleU-Net: A Deep Convolutional Neural Network for Medical Image Segmentation*. 2020. arXiv: 2006.04868 [eess.IV].
- [43] Debesh Jha et al. “Kvasir-seg: A segmented polyp dataset.” In: *Proceedings of International Conference on Multimedia Modeling (MMM)*. 2020, pp. 451–462.
- [44] Debesh Jha et al. “ResUNet++: An Advanced Architecture for Medical Image Segmentation.” In: *Proceedings of IEEE International Symposium on Multimedia (ISM)*. 2019, pp. 225–2255.
- [45] Tero Karras, Samuli Laine, and Timo Aila. *A Style-Based Generator Architecture for Generative Adversarial Networks*. 2018. arXiv: 1812.04948 [cs.NE].
- [46] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: 1412.6980 [cs.LG].
- [47] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2013. arXiv: 1312.6114 [stat.ML].
- [48] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks.” In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>.
- [49] Sang W. Lee et al. *Advanced Colonoscopy and Endoluminal Surgery*. Springer, 2017, p. 132.
- [50] Si Hyung Lee et al. “Endoscopic experience improves interobserver agreement in the grading of esophagitis by Los Angeles classification: conventional endoscopy and optimal band image system.” In: *Gut and liver* 8.2 (2014), p. 154.
- [51] Haoning Lin, Zhenwei Shi, and Zhengxia Zou. “Maritime Semantic Labeling of Optical Remote Sensing Images with Multi-Scale Fully Convolutional Network.” In: *Remote Sensing* 9 (May 2017), p. 480. DOI: 10.3390/rs9050480.
- [52] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. “Unsupervised Image-to-Image Translation Networks.” In: (Mar. 2017).
- [53] Luke Metz et al. *Unrolled Generative Adversarial Networks*. 2016. arXiv: 1611.02163 [cs.LG].
- [54] Min Min et al. “Computer-aided diagnosis of colorectal polyps using linked color imaging colonoscopy to predict histology.” In: *Scientific reports* 9.1 (2019), p. 2881.
- [55] Mehdi Mirza and Simon Osindero. *Conditional Generative Adversarial Nets*. 2014. arXiv: 1411.1784 [cs.LG].
- [56] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning.” In: *Nature* 518 (Feb. 2015), pp. 529–33. DOI: 10.1038/nature14236.
- [57] Volodymyr Mnih et al. *Playing Atari with Deep Reinforcement Learning*. 2013. arXiv: 1312.5602 [cs.LG].

- [58] Ahmed Mohammed et al. *Sparse coded handcrafted and deep features for colon capsule video summarization*. IEEE, 2017, p. 3.
- [59] Yuichi Mori et al. “Real-time use of artificial intelligence in identification of diminutive polyps during colonoscopy: a prospective study.” In: *Annals of internal medicine* (2018).
- [60] Guobing Pan and Litong Wang. “Swallowable Wireless Capsule Endoscopy: Progress and Technical Challenges.” In: *Gastroenterology research and practice* 2012 (Jan. 2012), p. 841691. DOI: 10.1155/2012/841691.
- [61] Harry A. Pierson and Michael S. Gashler. *Deep Learning in Robotics: A Review of Recent Research*. 2017. arXiv: 1707.07217 [cs.R0].
- [62] Stephen M. Pizer et al. “Adaptive Histogram Equalization and Its Variations.” In: *Comput. Vision Graph. Image Process.* 39.3 (Sept. 1987), pp. 355–368. ISSN: 0734-189X. DOI: 10.1016/S0734-189X(87)80186-X. URL: [http://dx.doi.org/10.1016/S0734-189X\(87\)80186-X](http://dx.doi.org/10.1016/S0734-189X(87)80186-X).
- [63] Konstantin Pogorelov et al. “KVASIR: A Multi-Class Image Dataset for Computer Aided Gastrointestinal Disease Detection.” In: *Proceedings of the 8th ACM on Multimedia Systems Conference. MMSys’17*. Taipei, Taiwan: ACM, 2017, pp. 164–169. ISBN: 978-1-4503-5002-0. DOI: 10.1145/3083187.3083212. URL: <http://doi.acm.org/10.1145/3083187.3083212>.
- [64] K. Pogorelov et al. “Deep Learning and Hand-Crafted Feature Based Approaches for Polyp Detection in Medical Videos.” In: *2018 IEEE 31st International Symposium on Computer-Based Medical Systems (CBMS)*. 2018, pp. 381–386.
- [65] Khansa Rasheed et al. *Machine Learning for Predicting Epileptic Seizures Using EEG Signals: A Review*. 2020. arXiv: 2002.01925 [cs.LG].
- [66] Jeroen Rijn et al. “Polyp Miss Rate Determined by Tandem Colonoscopy: A Systematic Review.” In: *The American journal of gastroenterology* 101 (Feb. 2006), pp. 343–50. DOI: 10.1111/j.1572-0241.2006.00390.x.
- [67] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].
- [68] Elisabeth Rosenthal. “The 2.7 Trillion Medical Bill.” In: (June 1, 2013). Fetched 06/12/2019. URL: <http://goo.gl/CuFyFJ>.
- [69] Hasim Sak, Andrew Senior, and Francoise Beaufays. “long short-term memory based recurrent neural network architectures for large vocabulary speech recognition.” In: (Sept. 27, 2016). Fetched 06/12/2019. URL: <https://arxiv.org/pdf/1402.1128.pdf>.
- [70] Jürgen Schmidhuber. “Deep learning in neural networks: An overview.” In: (2017). Fetched 10/8/2019, p. 95.
- [71] Florian Schroff, Dmitry Kalenichenko, and James Philbin. “FaceNet: A unified embedding for face recognition and clustering.” In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2015). DOI: 10.1109/cvpr.2015.7298682. URL: <http://dx.doi.org/10.1109/CVPR.2015.7298682>.
- [72] Alex Sherstinsky. “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network.” In: (Aug. 2018). Fetched 06/14/2019. URL: <https://arxiv.org/abs/1808.03314>.

- [73] David Silver et al. “Deterministic Policy Gradient Algorithms.” In: *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*. ICML’14. Beijing, China: JMLR.org, 2014, pp. 387–395.
- [74] David Silver et al. *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*. 2017. arXiv: 1712.01815 [cs.AI].
- [75] Kristina Sinaga and Miin-Shen Yang. “Unsupervised K-Means Clustering Algorithm.” In: *IEEE Access* PP (Apr. 2020), pp. 1–1. DOI: 10.1109/ACCESS.2020.2988796.
- [76] Pia H Smedsrød et al. *Kvasir-Capsule, a video capsule endoscopy dataset*. Aug. 2020. DOI: 10.31219/osf.io/gr7bn/. URL: <https://osf.io/gr7bn/>.
- [77] Jingkuan Song et al. “Dual Conditional GANs for Face Aging and Rejuvenation.” In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. IJCAI’18. Stockholm, Sweden: AAAI Press, 2018, pp. 899–905. ISBN: 9780999241127.
- [78] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting.” In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pp. 1929–1958. ISSN: 1532-4435.
- [79] Vajira Thambawita et al. *The Medico-Task 2018: Disease Detection in the Gastrointestinal Tract using Global Features and Deep Learning*. 2018. arXiv: 1810.13278 [cs.LG].
- [80] Sascha C Van Doorn et al. “Polyp morphology: an interobserver evaluation for the Paris classification among international experts.” In: *The American Journal of Gastroenterology* 110.1 (2015), pp. 180–187. DOI: 10.1038/ajg.2014.326.
- [81] Michael B. Wallace. “Endoscopic removal of polyps in the gastrointestinal tract.” English (US). In: *Gastroenterology and Hepatology* 13.6 (June 1, 2017), pp. 371–374. ISSN: 1554-7914.
- [82] Liansheng Wang, Cong Xie, and Yanxing Hu. *IDDF2018-ABS-0260 Deep learning for polyp segmentation*. 2018.
- [83] Liansheng Wang, Cong Xie, and Yanxing Hu. “IDDF2018-ABS-0260 Deep learning for polyp segmentation.” In: *Gut* 67 Suppl 2 (2018), A84–A85. ISSN: 0017-5749. DOI: 10.1136/gutjnl-2018-IDDFabstracts.181. eprint: https://gut.bmjjournals.org/content/67/Suppl_2/A84.full.pdf. URL: https://gut.bmjjournals.org/content/67/Suppl_2/A84.
- [84] Mei Wang and Weihong Deng. *Deep Face Recognition: A Survey*. 2018. arXiv: 1804.06655 [cs.CV].
- [85] Waqas Waqas Nadeem et al. “Brain Tumor Analysis Empowered with Deep Learning: A Review, Taxonomy, and Future Challenges.” In: *Brain Sciences* 10 (Feb. 2020), pp. 1–33. DOI: 10.3390/brainsci10020118.
- [86] Yingce Xia et al. “Dual Learning for Machine Translation.” In: (Nov. 2016).
- [87] K. Xu et al. “DeepRefiner: Multi-layer Android Malware Detection System Applying Deep Neural Networks.” In: *2018 IEEE European Symposium on Security and Privacy (EuroS P)*. 2018, pp. 473–487.
- [88] Shuoheng Yang, Yuxin Wang, and Xiaowen Chu. *A Survey of Deep Learning Techniques for Neural Machine Translation*. 2020. arXiv: 2002.07526 [cs.CL].

- [89] Tom Young et al. *Recent Trends in Deep Learning Based Natural Language Processing*. 2017. arXiv: 1708.02709 [cs.CL].
- [90] Da Zhang. “Recurrent Neural Network for Sequence Processing.” In: (Feb. 28, 2018). Fetched 06/13/2019. URL: <https://sites.cs.ucsb.edu/~yfwang/courses/cs281b/notes/RNN.pdf>.
- [91] Yizhe Zhang, Zhe Gan, and Lawrence Carin. “Generating Text via Adversarial Training.” In: 2016.
- [92] Q. Zhao and M. Q. -. Meng. “Polyp detection in wireless capsule endoscopy images using novel color texture features.” In: *2011 9th World Congress on Intelligent Control and Automation*. 2011, pp. 948–952.
- [93] XueFei Zhou. “Understanding the Convolutional Neural Networks with Gradient Descent and Backpropagation.” In: *Journal of Physics: Conference Series* 1004 (Apr. 2018), p. 012028. DOI: 10.1088/1742-6596/1004/1/012028. URL: <https://doi.org/10.1088/1742-6596/1004/1/012028>.
- [94] Jun-Yan Zhu et al. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. 2017. arXiv: 1703.10593 [cs.CV].