University of Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme/specialisation: Information Technology - Robotics and Signal Processing	Spring semester, 2020 Open
Author: Steinar Valle Larsen	
Programme coordinator: Ketil Oppedal Supervisor(s): Ketil Oppedal and Álvaro Fernández Quílez	
Title of master's thesis: Exploring Generative Adversarial Networks to Improve Prostate Segmentation on MRI	
Credits: 30	
Keywords: Generative Adversarial Networks, Convolutional Neural Network, Deep Learning, Deep Convolutional Generative Adversarial Network, Image-to-Image Translation, Biomedical Image Segmentation	Number of pages: 61 + supplemental material/other: 30 + ThesisCode.7z Stavanger, 28 June 2020 date/year

Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

Exploring Generative Adversarial Networks to Improve Prostate Segmentation on MRI

Master's Thesis in Robotics and Signal Processing
by

Steinar Valle Larsen

Internal Supervisors

Ketil Oppedal

Álvaro Fernández Quílez

June 28, 2020

Abstract

Prostate cancer is the second most occurring cancer and the sixth leading cause of cancer death among men worldwide. The number of cases is expected to increase dramatically due to population growth and increased expected lifetime. The magnetic resonance imaging (MRI) examination is an essential and a comfortable tool towards a precise diagnosis at an early stage. The examination method is already used at several hospitals, but its effective use depends on the expertise of clinical personnel.

This thesis will explore how generative adversarial networks can improve prostate segmentation on MRI. Different architecture within the topic of deep learning have proven to be accurate in biomedical image segmentation. However, it depends on a large volume of training data that is hard to obtain due to privacy policy. This thesis investigates the possibilities for generating new anonymized training data to improve biomedical image segmentation.

The final results improve the segmentation score compared to just using the original data. An underperforming segmentation network limits the segmentation results compared to other networks using the same data, but present the potential for expanding the dataset using generated data and improve the segmentation results.

Acknowledgements

This thesis marks the end of my Master's Degree in Robotics and Signal Processing at the Department of Electrical Engineering and Computer Science at the University of Stavanger.

I would like to give a special thanks to my head supervisor Ketil Oppedal and co-supervisor Álvaro Fernández Quílez, for valuable feedback and guidance throughout my last semester and this thesis work. I also want to thank Theodor Ivesdal and Rune Wetteland for help and advice related to the university's UNIX-system.

I would also like to thank all lectures and co-students for five exciting years filled with memories and new knowledge.

Contents

Abbreviations

2D	**T**wo-**D**imensional
3D	**T**hree-**D**imensional
CNN	**C**onvolutional **N**eural **N**etwork
DCGAN	**D**eep **C**onvolutional **G**enerative **A**dversarial **N**etworks
DL	**D**eep **L**earning
DRE	**D**igital **R**ectal **E**xamination
FE	**F**requency-**E**ncoding
GAN	**G**enerative **A**dversarial **N**etworks
GD	**G**radient **D**escent
GP	**G**eneral **P**ractitioner
ITK	**I**nsight **T**oolkit
MR	**M**agnetic **R**esonance
MRI	**M**agnetic **R**esonance **I**maging
NN	**N**eural **N**etworks
NumPy	**Num**erical **Py**thon
PE	**P**hase-**E**ncoding
PI-RADS	**P**rostate **I**maging **R**eporting and **D**ata **S**ystem
PROMISE12	**P**rostate **MR** **I**mage **S**egmentation 20**12**
PSA	**P**rostate-**S**pecific **A**ntigen
Pix2Pix	**Image-to-Image** Translation
SGD	**S**tochastic **G**radient **D**escent
TE	**E**cho **T**ime
TR	**R**epetition **T**ime
TRUS	**Tr**ansrectal **U**ltrasound **S**can
X-ray	**X**-radiation

Chapter 1

Introduction

1.1 Motivation

Prostate cancer is the second most occurring cancer and the six leading cause of cancer death among men, with 1.276.106 new cases and 358.989 deaths worldwide in 2018 [1]. This number is estimated to increase to approximately 2,3 million new cases by 2040, due to population growth and increased expected lifetime [2].

The diagnosis relies on several tests and evaluations, which are both time-consuming and expensive [3]. The first test is clinical and performed by a general practitioner (GP) without the expertise and proper tools. The clinical examination can create uncertainty as the available techniques used at the GP's office are often misleading and imprecise.

The current examination methods are not complication-free. One of the side effects experienced by several patients has been obtaining an infection after a guided biopsy procedure at the hospital. Increased use of a Magnetic Resonance Imaging (MRI) can make the examination process more comfortable for the patients and improve the efficiency of future examination with precise localization of the prostate gland [4].

Different architectures within the topic of Deep Learning (DL) have proven to be accurate on various vision recognition tasks, such as image classification and object detection [5]. A reliable automated segmentation tool could improve the existing MRI examination by freeing up time for experts while retaining the quality of the diagnosis and allow the GP to refer more patients.

A large volume of data is necessary to achieve precise results and train a successful machine learning algorithm [6]. Access to a high amount of biomedical cases is often hard to obtain in medical imaging due to privacy policy. The existing approach to extend a dataset called augmentation has improved the result of medical segmentation, but the new data is often highly correlated to the original. A method that extends the dataset with uncorrelated and anonymized data could potentially have a significant improvement to DL-based medical segmentation.

1.2 Problem Definition

This thesis's primary goal is to improve the robustness of image segmentation performed on T_2-weighted MRIs of the prostate. The main challenges of using biomedical data with DL are privacy policy, resulting in limited access to a large volume of data. To the best of my knowledge, this thesis introduces a new method to extend the dataset used to train a Convolutional Neural Network (CNN) for biomedical image segmentation performed on MRIs of the prostate.

1.2.1 Objectives

- To generate new anonymized training data using two Generative Adversarial Networks (GAN) architectures named Deep Convolutional Generative Adversarial Networks (DCGAN) and Image-to-Image translation (Pix2Pix).
- Use the CNN U-Net to segment the whole gland of the prostate from two-dimensional (2D) MRIs and compare the result using standard augmentation and the proposed method using GAN.

1.2.2 Proposed Method Overview

The augmentation does not require DL architectures, while the GAN based method uses two. The first generates segmentation masks, which are used to generate new uncorrelated and anonymized MRIs. The third and final DL architecture named U-Net evaluates the segmentation result using the original, augmented, and generated data. Figure 1.1 shows an overview of the proposed method.

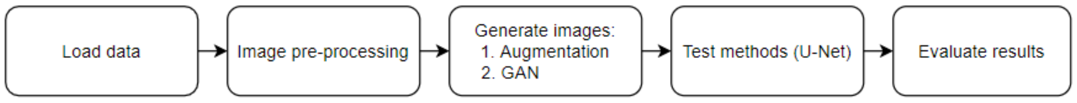


Figure 1.1: This figure illustrates a simple overview of this thesis methodology. The dataset is loaded, pre-processed, and extended with two different methods, augmentation and GAN. A segmentation model named U-Net tests the proposed method and evaluate the result.

1.3 Related Work

This thesis uses three different DL architectures to generate images and test the result. The DL-based image generator utilizes two existing GAN architectures. The model that generates segmentation masks is named Deep Convolutional Generative Adversarial Networks (DCGAN), first introduced by Alec Radford et al. in the paper *"Unsupervised representation learning with deep convolutional generative adversarial networks"* [7]. Their goal was to combine the existing Convolutional Neural Network (CNN) and Generative Adversarial Networks (GAN) with unsupervised learning. Yann LeCUN, et al. first proposed CNN in the paper *"Backpropagation Applied to Handwritten Zip Code Recognition"* [8] and Ian Goodfellow, et al. first proposed GAN in the paper *"Generative adversarial nets"* [9].

The second architecture utilized to generate images is named Image-to-Image Translation (Pix2Pix), first proposed by Phillip Isola et al. in the paper *"Image-to-image translation with conditional adversarial networks"* [10]. Their network learns the mapping from input to output images and a loss function to train this mapping. Their paper demonstrates that this approach can generate photos from label maps, colorize images, and reconstruct objects from edge maps.

To the best of my knowledge, there is no published work on generating MRIs and corresponding segmentation masks of the prostate gland using DCGAN and Pix2Pix. The paper *"Medical Image Synthesis for Data Augmentation and Anonymization using Generative Adversarial Networks"* [11], written by Hoo-Chang Shin et al., generates synthetic MRIs of the brain using Pix2Pix GAN. They uses real segmentation masks with multiple classes, instead of generating new synthetic.

The U-Net segmentation network is used in this thesis to evaluate the proposed method. Olaf Ronneberger et al. first proposed U-Net in the paper *"U-Net: Convolutional Networks for Biomedical Image Segmentation"* [12]. This method was designed with a training strategy that took advantage of data augmentation to use the available data more efficiently.

This thesis uses the Prostate MR Image Segmentation 2012 (PROMISE12) challenge dataset [13]. Some previous implementation is published, where this thesis has used Inom Mirzaev's work on *"Fully convolutional neural network with residual connections for automatic segmentation of prostate structures from MR images"* to solve the PROMISE12 challenge as an inspiration for the U-Net implementation and data pre-processing. [14]

1.4 Outline

This thesis starts with a brief introduction and explanation of the motivation for the subject. The remaining part of this thesis is structured into seven different chapters.

- The second chapter is named Medical Background and describes the essential medical theory used in this thesis.
- The third chapter is named Technical Background and contains the most important background theory related to the technology used in this thesis.
- The fourth chapter, named Materials and Image Pre-Processing, describes the dataset used in this thesis and the pre-processing methods used to improve the data.
- The fifth chapter is named Solution Approach and describes the proposed method to generate new data using GAN.
- Chapter six, named Experimental Evaluation, describes the experimental setup, including U-Net, and evaluates the generated data both visually and with the use of U-Net. This chapter does also compares the U-Net performance after training on original, augmented, and original combined with generated data.
- Chapter seven presents a discussion of the results and limitations of this thesis.
- Chapter eight is the last one and presents the conclusion of this thesis work.

Chapter 2

Medical Background

2.1 Prostate Cancer

Prostate cancer is the second most occurring cancer and the six leading cause of death caused by cancer among men, with 1.276.106 new cases and 358.989 deaths worldwide in 2018 [1]. This number is estimated to increase to approximately 2,3 million new cases by 2040, due to population growth and increased expected lifetime [2].

A process called cell division regenerates all tissue in the human body. Old cells die, and new cells are born. The growth pattern can, in some cases, change and cause overproduction, known as a tumor. These tumors can be both benign and malignant. When cell division gets out of control and immature cells grow and propagate, cancer is created. Cancer cells can grow into neighbor tissues and expand to different parts of the human body. [15]

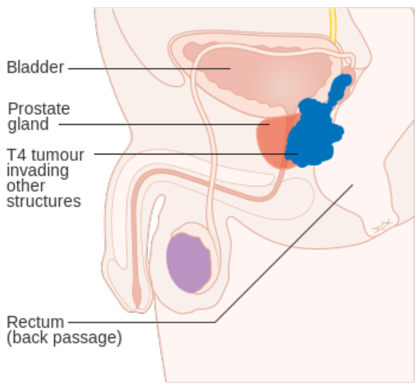


Figure 2.1: Diagram showing stage T4 prostate cancer.
The figure is reprinted in unaltered form from Wikimedia commons, File: Diagram showing stage T4 prostate cancer CRUK 454.svg, licensed under CC BY-SA 4.0

The location of the prostate is underneath the bladder. The first part of the urethra goes through the prostate. The gland's main task in the prostate is to provide liquid in order to keep the semen flow. The prostate will continue to grow as men grow old, and eventually, most men can experience symptoms of a larger prostate. A growing prostate is not necessarily cancerous but can give similar symptoms as a tumor. Prostate cancer symptoms are often frequent urination, trouble starting and stopping while urinating, and blood in the urine. [15]

2.2 Prostate Cancer Examination Methods

There are already some existing diagnostic methods to determine if a patient has prostate cancer. The following examination routines are often part of the entire diagnostical process if the patient experiences symptoms such as frequent urination.

2.2.1 Digital Rectal Exam

This is an examination where the GP inserts a gloved, lubricated finger into the rectum. The GP will be able to feel the rear part of the prostate and determine its size and shape. A GP performs Digital Rectal Exam (DRE) before referring the patient to an expert. Illustration of DRE in figure 2.2. [16]

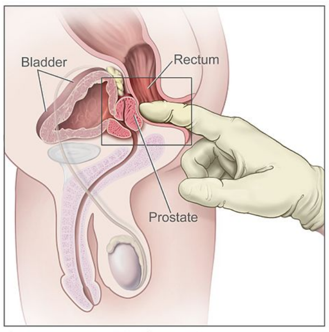


Figure 2.2: Digital rectal examination.
The figure is reprinted in unaltered form from Wikimedia commons, File: 482px-Digital_rectal_exam.jpg

2.2.2 Prostate-Specific Antigen Test

This is an examination where the Prostate-Specific Antigen (PSA) level in the patient's blood is measured by obtaining a blood sample. The amount of PSA in the blood increases when a patient has prostate cancer, but this will also increase when the patient has a benign growth of the prostate. The test is most useful when the doctor can compare a sample taken before and after the patient got cancer. [15]

2.2.3 Biopsy

If the GP suspects cancer, biopsies will be the next step. The most common prostate biopsy method is a transrectal ultrasound scan (TRUS). However, the risk of infection is proven to be between 5% and 7%, even with preventive measures [17]. This examination is performed at the hospital by specialists, where a needle gets inserted into the prostate eight to ten times in order to obtain tissue from several parts [15]. Figure 2.3 visualizes the process of TRUS. The patient's prognosis is obtained using the Gleason score grading system on a sample from the prostate biopsy. A pathologist evaluates the Gleason score, measuring the staging for prostate cancer. A high Gleason score indicates aggressive cancer with a worse prognosis. [18]

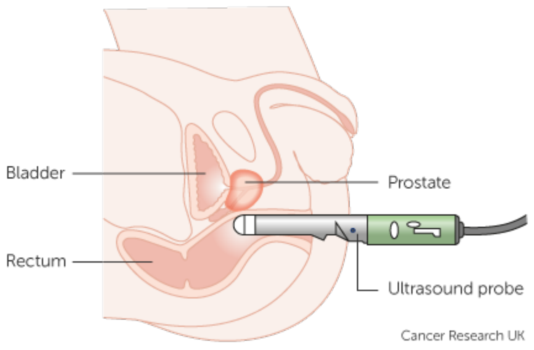


Figure 2.3: Transrectal ultrasound scan examination.
The figure is reprinted in unaltered form from Cancer Research UK's webpage [19]

2.2.4 Magnetic Resonance Imaging

MRI examinations locate the tumor and are often taken before a biopsy, as they improve the precision. Some articles indicate that MRI can assist in determining if the tumor is aggressive or benign [20]. Good cooperation between MRI-experts and surgeons can improve the current treatment and have a significant impact on the prognosis of the disease at an early stage. [15]

MRIs are evaluated by a radiologist using the Prostate Imaging Reporting and Data System (PI-RADS) scoring system [21]. PI-RADS was designed to promote global standardization of prostate multiparametric magnetic resonance imaging (mpMRI) examination. Standardization aims to improve the detection of clinically significant cancer and locate benign diseases to diminish unnecessary biopsy. [21]

Chapter 3

Technical Background

3.1 Magnetic Resonance Imaging

This chapter starts by explaining some terminology within the topic of Magnetic Resonance (MR) technology and it continues by explaining the process involved in MRI.

3.1.1 Basic Terminology

An MR-examination depicts digital images of internal organs with the use of a strong magnetic field and radiofrequency. MRIs are obtained using a pulse sequence involving adjustable timing values, termed Repetition Time (TR) and Echo Time (TE). TR is the time between similar events on a recurrent series of pulses and echoes. The TE describes the time separating the center location of the RF pulse and the corresponding echo. [22]

MRI uses the natural properties of hydrogen in water or lipids to capture images. Two of the most fundamental parameters are characteristic times, termed spin-lattice relaxation time (T_1) and spin-spin relaxation time (T_2). This thesis uses T_2-weighted images, where tissues with long T_2 specific time are brighter as the signal intensity becomes strong. T_1 is the opposite, where tissues with long T_1 specific time result in a weak signal. T_2-weighted images are more time consuming to produce than T_1-weighted images, as long TR and TE are required. The brightest intensity of a T_2-weighted image shows fluids, and the mid-grey values correspond to water- and fat-based tissues. Regions of abnormal fluids have the most substantial intensity compared to the darker ones for healthy tissues. [22]

3.1.2 Analog-to-Digital Converter

The analog MRI signal is transformed into a digital matrix and visualized as an image. Each pixel in the image corresponds to a value originating from the matrix. The original analog MR signal is continuous, where each point in time corresponds to a value. The signal is transformed from continuous to discrete using an analog-to-digital converter. A digital image is always discrete, as each pixel corresponds to one sample. A discrete signal is represented by samples in the time domain and can be less accurate. [22]

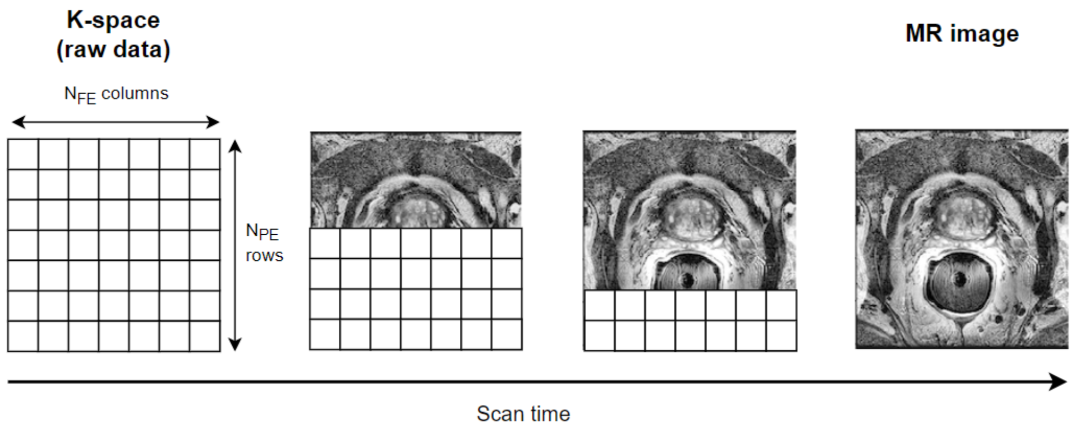


Figure 3.1: Illustration of the process behind capturing a MRI

The pixels in the digital image are sorted into rows and columns in an image array. For MRIs, the columns are often referred to as the frequency-encoding (FE) matrix, and the rows are referred to as the phase-encoding (PE) matrix, as illustrated in figure 3.1. MRIs does also have a third dimension, which is the slice thickness. Each MRI contains several slices that correspond to multiple 2D images in depth. An empty array is created before the scanner starts to fill in the information. The empty array is called raw data space and often termed as k-space. The scanner fills one row per sequence until the entire array corresponds to the MRI, as illustrated in figure 3.1. [22]

3.2 Neural Network

The first Neural Network (NN) design was published by Warren McCulloch and Walter Pitts in 1943 [23], who developed a NN based on algorithms named threshold logic. The name originates from the neural architecture of a human brain. A NN is a structure of processing elements (often referred to as neurons or nodes) connected with unidirectional signal channels, termed connections. Each neuron calculates and distributes a value via connections to the next layer of neurons. The neuron's output corresponds to different types of mathematical functions, but all processing done by the neuron must only depend on the current input values and values stored in the local memory of a neuron [24].

Figure 3.2 illustrates a simple feedforward network example with two inputs values ($\mathbf{x} = [x_1, x_2]$), one hidden layer with four neurons, and two outputs values ($\hat{y}$). The input value $\mathbf{x}$ contains the initial information that propagates to each neuron in the hidden layers and, at the end it predicts the output ($\hat{y}$).

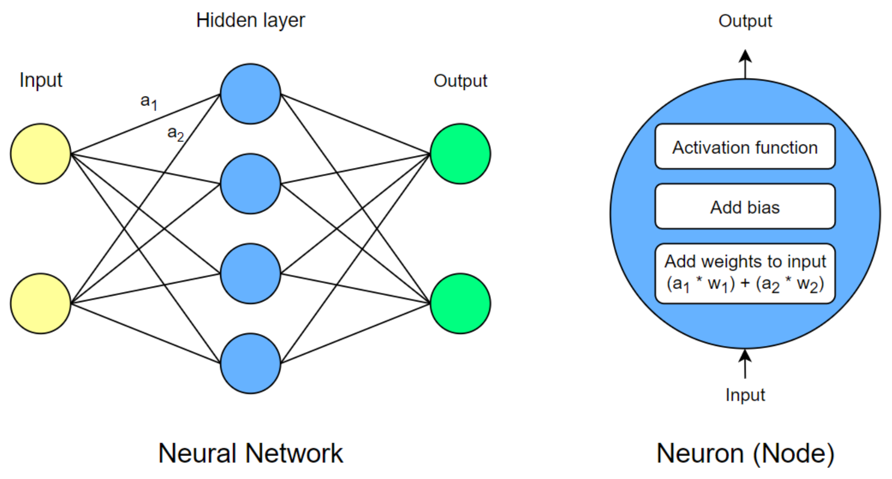


Figure 3.2: Illustration of a neural network on the left side with two input values (x), one hidden layer including 4 neurons, and two output values. Neurons, input, and output values are combined with connections. The right side of the figure illustrates a neuron and the including mathematical functions.

Figure 3.2 does also illustrates a neuron with the corresponding mathematical functions. A neuron has N number of input values received over N connections. The example in figure 3.2 has two inputs, equal to x_1 and x_2, for each neuron in the hidden layer. The mathematical layer within a neuron adds a specific weight ($w = [w_1, ..., w_N]$), learned by the algorithm itself, to the input vector ($x = [x_1, ..., x_N]$). The next mathematical function adds a parameter, termed bias (b), to adjust the output value. The neuron process this information and passes the result into an activation function $g(z)$ [25]. The result after the activation function is equal to the neuron output, where $\hat{y}$ is equal to the final hidden layer output, as shown in equation 3.1.

$$\hat{y} = g(z) = g\left(\sum_i w_i x_i + b\right) \tag{3.1}$$

Equation 3.1 is the calculation of the last hidden layer, where g is the activation function and z corresponds to the neuron input and local parameters. The input variable x in equation 3.1 resembles the previous layer's output value or the network input values if the calculation applies to the first hidden layer.

3.2.1 Backpropagation and Gradients

A NN architecture learns by minimizing the loss value L in equation 3.2 and improves the prediction by adjusting the local weights w in each neuron. The goal of the learning process is to find weights that predict an output equal or close to the expected value for each input. With fixed and finite training data, including both input and expected output, the calculation of weights is a measure of the variation between the predicted and expected output.

$$L(w) = Loss(\hat{y}, y) \tag{3.2}$$

For general feedforward NNs, the method popularized by David E. Rumelhart et al., named backpropagation, allows the information from the loss to propagate backward through the network to compute the gradient [26]. Backpropagation is a method to compute the gradient, while optimizer algorithms learn by using this gradient [27, page 279 - 307]. In learning algorithms, the method to compute the gradient utilizes the chain rule for derivatives that compute the gradient of the loss function $L(w)$ with respect to the weights w, as calculated in equation 3.3. [26]

$$\frac{\partial L(w)}{\partial w_{ij}} = \frac{\partial L(w)}{\partial x_j} \cdot \frac{\partial x_j}{\partial w_{ij}} \tag{3.3}$$

In equation 3.3, the parameter x_j corresponds to the j-th input, and w_{ji} is the weight for the i-th output towards the j-th input, as backpropagation transfer information backward. [26]

3.2.2 Optimizer Algorithms

Optimization is common within DL algorithms and relates to minimizing or maximizing a function $f(x)$ by adjusting x. In this thesis, optimizers improve the learning process by minimizing the loss function. These optimizers are gradient-based and use the gradient calculated by algorithms, such as backpropagation, to learn features that minimize the loss in a process, termed gradient descent (GD). [27, page 279 - 307]

Figure 3.3 shows a function $f(x)$ including one global minimum and two local minimum. The global minimum obtains the lowest value of $f(x)$, and the local minimum is a point where $f(x)$ is lower than all neighboring points on both sides. The goal of optimization is to find the global minimum, but one prominent problem is that the local minimum often seems global and reduces the model's performance.

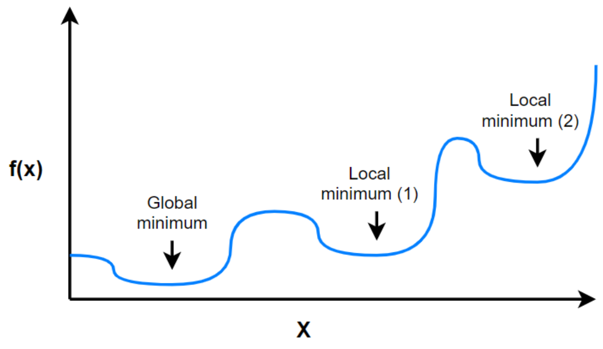


Figure 3.3: Function $f(x)$ including one global minimum and two local minimum.

The algorithm used in this thesis is the Adam optimizer algorithm, which combines a basic algorithms named stochastic gradient descent (SGD) with Momentum [28], and an algorithm with an adaptive learning rate[1], named RMSProp [29].

Stochastic Gradient Descent with Momentum

The original SGD is an iterative algorithm that aims to optimize a loss function under a specific criteria. SGD starts with an initial guess and generates a sequence of values for the learning rate. The algorithm replaces the original gradient with an approximation, which means that it is a stochastic estimate of the gradient descent optimization. The algorithm is a popular optimization algorithm within the topic of machine learning, but it can sometimes be slow. The SGD with momentum accelerates learning in situations with high curvature, noisy gradients, or small, consistent gradients. The algorithm accumulates an exponentially decreasing moving average of previous gradients and proceeds in that direction [27, page 290 - 296].

$$w := w - \eta \Delta L_i(w) + \alpha \Delta w \tag{3.4}$$

Equation 3.4 calculates new values for the weights using SGD with momentum. Where w is the weight, and L_i correspond to the loss for the i-th observation. The variable η is the current learning rate, and α is an exponential decay factor between zero and one. [30]

[1]The learning rate determines the step size for an optimization algorithm that seeks to minimize a loss function.

RMSProp

The RMSProp algorithm has an adaptive learning rate, which keeps a moving average of the magnitudes of previous gradients for each weight. The magnitude of the gradient does change differently for each weight and can vary during the learning process, making algorithms with adaptive learning rates preferable. The main advantage of using RMSprop compared to earlier proposed optimizers with adaptive learning rate, is the opportunity to use mini-batches. A mini-batch is a small batch of the training dataset that optimizers are using to update different types of parameters, mainly model weights. [29]

$$MeanSqueare(w,t) = 0.9 * MeanSquare(w,t-1) + 0.1\left(\frac{\partial C_i(w)}{\partial w^{(t)}}\right)^2 \tag{3.5}$$

In equation 3.5 the parameter w corresponds to the weight and t resembles the time. The factor 0.9 and 0.1 is just weighted factors for the average squared gradient and the current gradient. [29]

$$w := w - \frac{\eta}{\sqrt{MeanSquare(w,t)}}\Delta C(w) \tag{3.6}$$

Adam

The name Adam originates from the phrase *"adaptive moments"* as the algorithm is a variant of the combination of RMSProp with adaptive learning rate and the SGD with momentum. The momentum in adam optimizer is included directly as an estimate of the first-order moment of the gradient. Adam includes a bias to the estimates of the first-order moments and the second-order moments, considering their initialization at the origin. [27, page 302 - 306]

The pseudo-code to explain the Adam optimizer algorithm 3.1 is reprinted in unaltered form from the original paper *"ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION"*, written by Kingma and Ba in 2014. [31].

3.2.3 Activation Function

The activation function g(z), determines the output of a neuron and can be both linear and non-linear. The value z corresponds to the local values calculated in each neuron (see section 3.2). To learn parameters for complex non-linear models, the activation function for each neuron in the hidden layers must be non-linear. This section introduces some basic intuitions motivating some commonly used activation functions.

Algorithm 3.1 *Adam*, our proposed algorithm for stochastic optimization. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power of t

```
Input: $\alpha$: Stepsize
Input: $\beta_1, \beta_2 \in [0, 1]$: Exponential decay rates for the moment estimates
Input: $f(x)$: Stochastic objective function with parameters $\theta$
Input: $\theta_0$: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize $1^{st}$ moment vector)
  $v_0 \leftarrow 0$ (Initialize $2^{nd}$ moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while $\theta_t$ not converged do
    $t \leftarrow t + 1$
    $g_t \leftarrow \Delta_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t stochastic objective at timestep $t$)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\hat{m}_t \leftarrow m_t/(1 - \beta_1^t$ (Compute bias-corrected first moment estimate)
    $\hat{v}_t \leftarrow v_t/(1 - \beta_2^t)$ (Compute biased corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return $\theta_t$ (Resulting parameters)
```

Rectified Linear Unit

Rectified Linear Units (ReLU) are units (often referred to as neurons) with the Rectified Linear activation functions. The mathematical description of ReLU is shown in table 3.1. The ReLU is similar to a linear function but returns zero for half of its domain. As a result, the first derivative remains large and consistent. The first derivative of ReLU, often referred to as the gradient, is zero for values above zero and one for values below or equal to zero. [27, page 187 - 192]

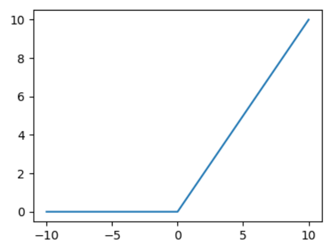
Plot	Activation function
	$g(z) = \begin{cases} z & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$

Table 3.1: ReLU activation function.

Leaky Rectified Linear Unit

Leaky Rectified Linear (LeakyReLU) activation was first presented by Maas et al. [32]. Table 3.2 shows the mathematical description of the function, where the difference between this and the original version is the fixed parameter a. The original paper recommends a to be 100, but it can also be a number in the range $[1, \infty]$. [33]

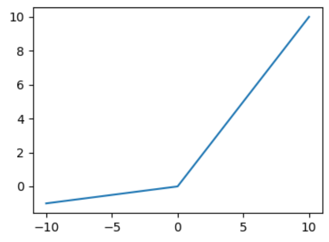
Plot	Activation function
	$g(z) = \begin{cases} z & \text{if } z \geq 0 \\ \frac{z}{a} & \text{if } z < 0 \end{cases}$

Table 3.2: LeakyReLU activation function.

Sigmoid

The sigmoid activation function exists in the range $[0, 1]$, as shown in table 3.3. This thesis uses the Sigmoid on some of the output units to predict a binary variable. Unlike ReLU and LeakyReLU, the sigmoidal units saturates most of their domain. Large positive z saturates to a high output value, and large negative z saturates to a low output value, but the output is sensitive when z is close to zero. The Sigmoid activation is often just used in the output layer where a loss function can compensate for saturation, as it makes gradient-based learning hard within the hidden layers. [27, page 178 - 192]

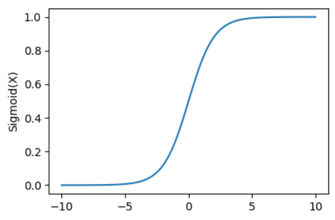
Plot	Activation function
	$g(z) = \sigma(z) = \frac{1}{1+e^{-z}}$

Table 3.3: Sigmoid activation function.

Hyperbolic Tangent

The Hyperbolic Tangent (TanH) is similar to the Sigmoid function, as shown in table 3.4. Unlike the sigmoid function, the range is $[-1, 1]$. The main advantage of TanH against the Sigmoid function is that TanH is not that sensitive to z values close to zero. Negative

z saturates to a negative output, and positive z saturates to a positive output. Input values z close to zero results in an output near zero. [27, page 178 - 192]

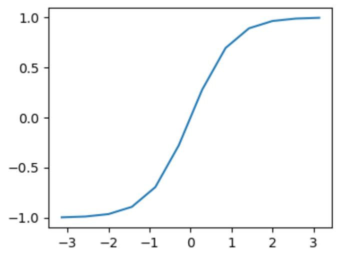
Plot	Activation function
	$g(z) = 2\sigma(2z) - 1 = \frac{2}{1+e^{-2z}} - 1$

Table 3.4: TanH activation function.

3.2.4 Supervised, Semi-supervised and Unsupervised Learning

A machine learning problem needs input information to learn. The term supervised learning corresponds to the process where the input x has an expected output y, termed label, and tries to produce a predicted output $\hat{y}$ equal to y. An unsupervised problem uses unlabeled data to train. It seeks to automatically discover and learn patterns and regularities in the input data to produce the predicted output $\hat{y}$. A Semi-supervised problem is a combination of supervised and unsupervised learning, that learns from both labeled and unlabeled data. [34] [35, page 3 - 5]

3.3 Convolutional Neural Networks

Deep CNNs have outperformed and dominated the task of visual recognition in recent years [36] [37]. The technology has existed for a long time [8], but small datasets and limited access to computer power narrowed the potential of the technology.

"Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers."

— Page 326, *Deep learning*, 2016 [27]

CNN consists of one or more models that take an input image, pass it through convolutional and other image processing layers, and get an output. This chapter explains some of the main topics within the technology of CNN.

3.3.1 Convolution Layer

A standard convolutional layer starts with an input image of size $n*m$, where n represents the number of pixels in width and m in height. The convolutional calculation between the input and output depends on two main parameters, the number of padding (p) and stride (s). The padding adds a border around the image with a specific value. A commonly used padding technique is zero padding, which adds a border of pixels with the value of zero around the input array. Padding is used when the kernel size and input size does not add up. The parameter named strides decides the number of n or m to shift the kernel when moving across the image. A kernel extract features, like edges and corners, from a region termed receptive field, and produces an output termed feature map. The spatial dimension of the output images after convolutional layers is often less than the input, but can also be equal if the stride is equal to 1. [27, page 327 - 329]

Figure 3.4 visualizes the standard process achieved by a convolutional layer, where the input illustrates an image with m and n equal to 6 pixels.

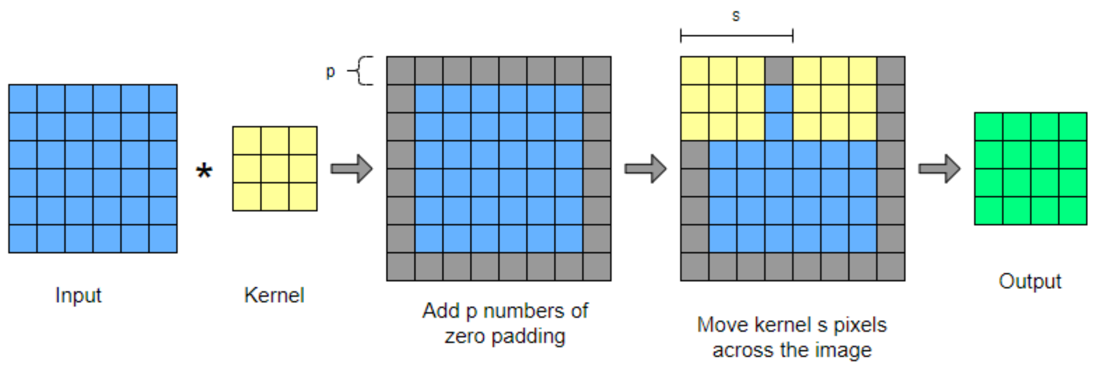

Input
*
Kernel
p
Add p numbers of zero padding
s
Move kernel s pixels across the image
Output

Figure 3.4: Illustration of the process behind a convolution layer

3.3.2 Transposed Convolution Layer

Transposed convolution is an implementation of trainable upsampling. Figure 3.5 visualizes the process of transposed convolution. Similar to the standard convolution layer, transposed layer is also defined by strides (s) and padding (p). The input layer does often have a smaller spatial dimension, then the output. The first step is to add zeros between each number in the input, which increases the image size to $(2*m-1, 2*n-1)$. The next step is zero padding, that adds p layers of pixels equal to zero, surrounding the image array. Standard convolution with strides (s) processes the new image with zeros between and around the original pixels and returns the output array. A common mistake is to define transposed convolution as the opposite of standard convolution.

This interpretation is wrong as the transposed convolution does only reverse the spatial dimensions but not the values of standard convolution. [38]

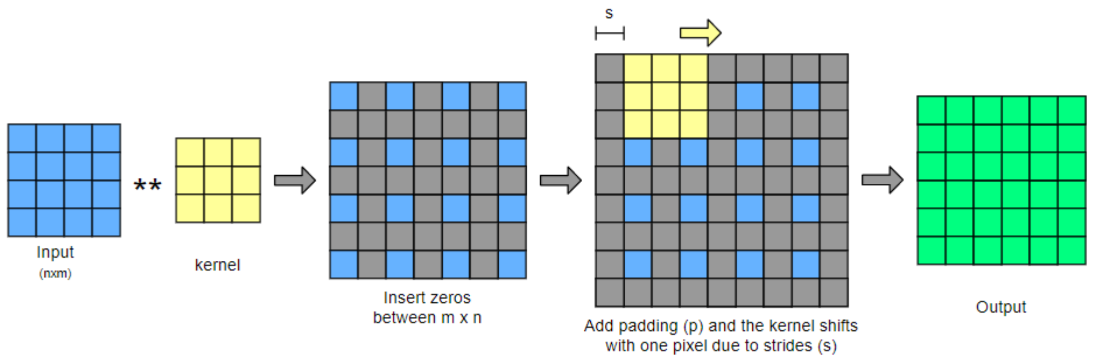

s
Input
(nxm)
**
kernel
Insert zeros
between m x n
Add padding (p) and the kernel shifts
with one pixel due to strides (s)
Output

Figure 3.5: Illustration of the process behind a transposed convolution layer

3.3.3 Max Pooling

The max pooling layer downsamples the image as well as it keeps the largest (and most important) values. The function downsamples the image to halve the input in both spatial dimensions, if the pooling size is equal to 2x2. The concept of max pooling with the same parameters, as used in this thesis, is visualized in figure 3.6. [27, page 335 - 339]

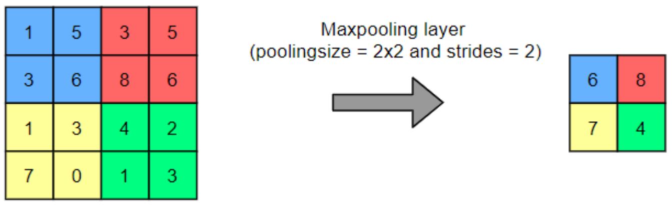


Figure 3.6: Illustrate the max pooling process with pooling size of 2x2.

3.3.4 Flatten and Fully Connected Layers

The layer named flatten is usually at the end of the model. The input of a flatten layer is equal to a 2D array, such as an image. The flatten layer transforms this array into a one-dimensional linear vector to connect with the fully connected layer. Fully connected layers originate from the original NN structure and are often implemented at the end of a convolutional network to make a decision. Figure 3.7 illustrates a standard process where an array is transformed with a flatten layer and combined with a fully connected layer. [39]

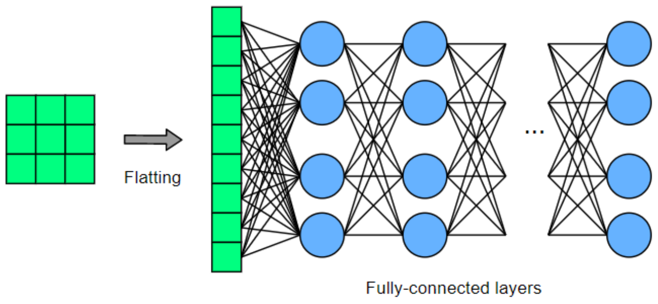


Figure 3.7: Illustration of flatten and fully connected layers

3.3.5 Dense Layer

A Dense layer includes neurons from a NN and is commonly used in classification models. A NN is a structure of several nodes, where each node includes three layers, weights, bias, and an activation function, as explained in chapter 3.2. A dense layer is a part of the Keras library [40] and consists of two or more nodes. This layer is useful as one node can only draw one decision boundary[2]. [41]

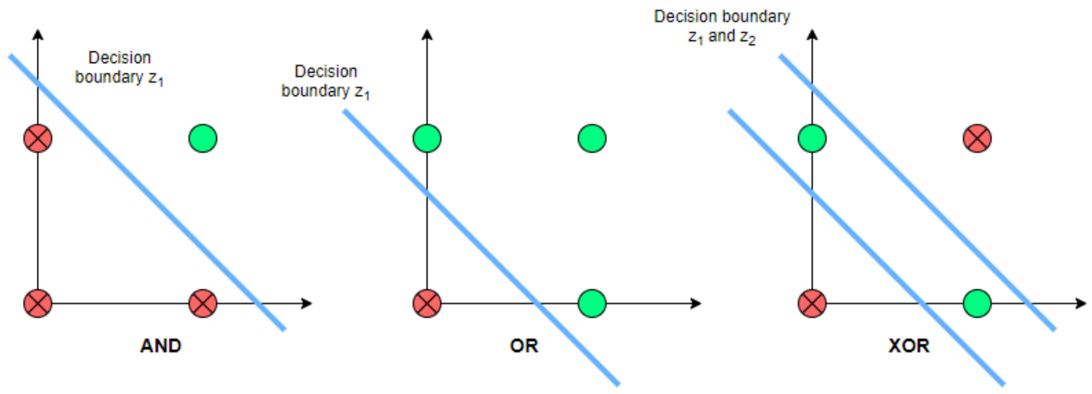


Figure 3.8: Shows classification problems where two is possible to separate with one decision boundary and one needs a dense layer as it is impossible to classify using one decision boundary.

Figure 3.8 illustrates three examples where two are possible to separate with one decision boundary, and one is not. The illustration visualizes an AND gate, OR gate, and an XOR gate. Since one node draws one decision boundary, two nodes are required to draw the decision boundaries for an XOR gate. The number of neurons (N) in a dense layer is equal to the number of classes in the output. For classification on values reaching from 0 to 9, N is equal to ten. [41]

[2]A decision boundary is a linear function separating samples from a vector space into two classes.

3.3.6 Operations performed in CNN

This section describes some typical operations performed in CNN to boost performance.

Batch normalization

All three models in this thesis are CNNs where several layers are stacked on top of each other. These networks cause the distribution to change throughout the training as each layer's input is affected by the previous. Small parameter variations in the previous layer increase as the network grows, which can cause a considerable variation at the end feature map distribution. Equation 3.7 describes the calculation of batch normalization, which performs normalization on the previous layers and takes advantage of working with batches to make the network more stable. The input of the current layer is equal to x^k, and the batch normalization's output is $\hat{x}^k$. The batch mean is subtracted from the past layer output and divided by the batch standard deviation. [42]

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}} \tag{3.7}$$

Dropout

NNs include non-linear hidden layers that learn complicated relations between the input and the expected value. The result can be affected by sampling noise generated, if training data is deficient. This problem is called overfitting, in machine learning terms. One of the most common techniques to avoid overfitting is termed dropout, first proposed by Srivastava, Nitish, et al." Dropout: a simple way to prevent neural networks from overfitting", JMLR 2014. [43]

The term dropout applies to drop neurons, both hidden and visible, in a NN. The dropout approach is temporarily excluding the neurons and the associated connections from the network. Which neurons to drop is randomly calculated, where each neuron is retained with the fixed independent probability of p. [43]

3.4 Augmentation

Image augmentation is a well-established method to expand the training data set artificially. Data augmentation is not made to expand the dataset to more examples, but rather to augment the images to new versions of themselves for each epoch[3] or batch[4]. The model that learns features from the images will read the augmented versions as new images, known as artificial expanding. Several different implementations are accessible online. This thesis makes use of the Keras built-in function for image augmentation [44]. Some of the image augmentation types used in this thesis are listed below. [45]

- Image shift: Moves all the pixels of an image in one direction. This augmentation method removes a region of the image and leaves a black region on the other side.
- Rotation: Rotates the image clockwise in the range of [0, 360] degrees, starting from the original angle. The rotation augmenter rotates some pixels out of the image and leaves a black region.
- Horizontal flip: Reverses all rows of pixels.
- Vertical flip: Reverses all columns of pixels.
- Zoom: Applies random zooming in the range of [min, max] on the image. Zoom parameter equal to 1 corresponds to the original image. Zoom larger than one moves the object further away, while a zoom less than one moves the object closer.

3.5 Generative Adversarial Network

The GAN architecture was first introduced in 2014 by Ian Goodfellow et al. in the paper titled *Generative Adversarial Networks* [9]. The baseline for GAN is a game-theoretic scenario with an architecture of two competing models, a generator that generates images similar to the training data and a discriminator that classifies the input images from the dataset as true and the generator as false. Generative modeling, in general, is an unsupervised learning process without a label to correct the prediction. GAN solves the generative process by framing the task as supervised learning, where the discriminator acts as the label. The two next subsections explain two GAN architectures that generate images is this thesis. [35, page 3 - 11] [46]

3.5.1 Deep Convolutional Generative Adversarial Networks

In recent years, DCGAN has become a standard implementation of GAN. The structure was first applied in the paper *"Unsupervised Representation Learning with Deep*

[3]One epoch implies that the dataset is transferred through a neural network once.
[4]Batch size is the number of training samples simultaneously passed through a neural network.

Convolutional Generative Adversarial Networks", written by Alec Radford et al. in 2016 [7]. DCGAN was developed to promote the unsupervised learning in combination with convolutional networks.

DCGAN is similar to the original GAN architecture and consists of two CNNs, one generator, and one discriminator. The main difference is that convolutional stride replaces max pooling, transposed convolution is used instead of upsampling, and fully connected layers are removed. Figure 3.9 illustrates the structure of the generator from the original paper for DCGAN.

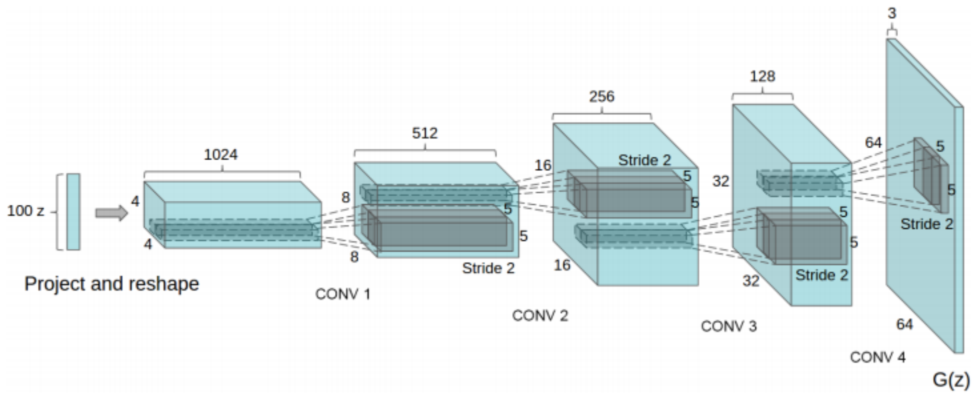


Figure 3.9: Illustration of the original DCGAN generator implementation. *The figure is reprinted in unaltered form from the paper written by Alec Radford et al. named "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks"* [7]

3.5.2 Image-to-Image Translation

Image-to-image translation, also known as Pix2Pix, was first published in the paper *"Image-to-Image Translation with Conditional Adversarial Networks"* written by Phillip Isola et al. in 2017 [10]. The article investigates conditional adversarial networks as a general-purpose solution to image-to-image translation problems. Pix2Pix is used in this thesis to generate new MRIs based on the DCGAN generated segmentation masks.

The Pix2Pix architecture learns the mapping between input and output image, but also a loss function to train this mapping. As a result, the same generic approach can be applied to problems that traditionally would require complex loss functions. The article demonstrates that image-to-image translation is useful to generate images from label maps, to reconstruct objects from edge maps, and to colorize images based on segmentation masks. [10]

The Pix2Pix architecture is also built with two competing models, one generative and one discriminative model. A high-level illustration of the cooperation between the generator and discriminator is visualized in figure 3.10

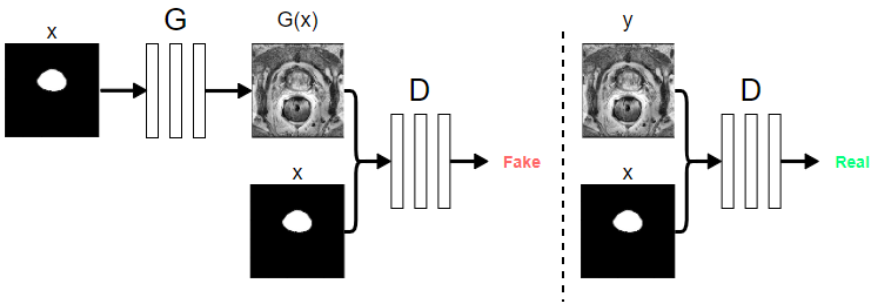


Figure 3.10: Training a conditional GAN to map segmentation masks to MRIs. The discriminator, D, learns to classify between synthetic and real MRIs. The generator, G, learns to fool the discriminator. [10]

The Pix2Pix generator uses a U-Net based architecture, which is the same technology as explained in chapter 3.6. The input image is equal to a feature map, for example, a segmentation mask. This image is compressed into a low dimensional vector representation. The generator tries to upsample this image to be similar to the expected value, and fool the discriminator. The expected value can, for example, be the corresponding MRI slice to a segmentation input. [35, page 466 - 516]

The Pix2Pix discriminator, often referred to as PatchGAN discriminator, classifies a (N x N) patch of the image. The advantage of PatchGAN discriminators, compared to classifying the entire image at once, is fast computing and better classifying of small details. [35, page 466 - 516]

3.6 U-Net segmentation

A binary segmentation means that the segmentation consists of two classes, background, and object. The segmented part of the image array is equal to one, and the background is equal to zero.

U-net has become a common tool to perform image segmentation. The structure was first applied in 2015 to process biomedical images [12]. An illustration of the original structure is visualized in figure 3.11. The name U-net originates from the shape of the network. The left part of the architecture consists of a contracting path. The contracting path corresponds to a general convolutional network, which captures context. A symmetric expanding path is added to the right side, which enables precise localization.

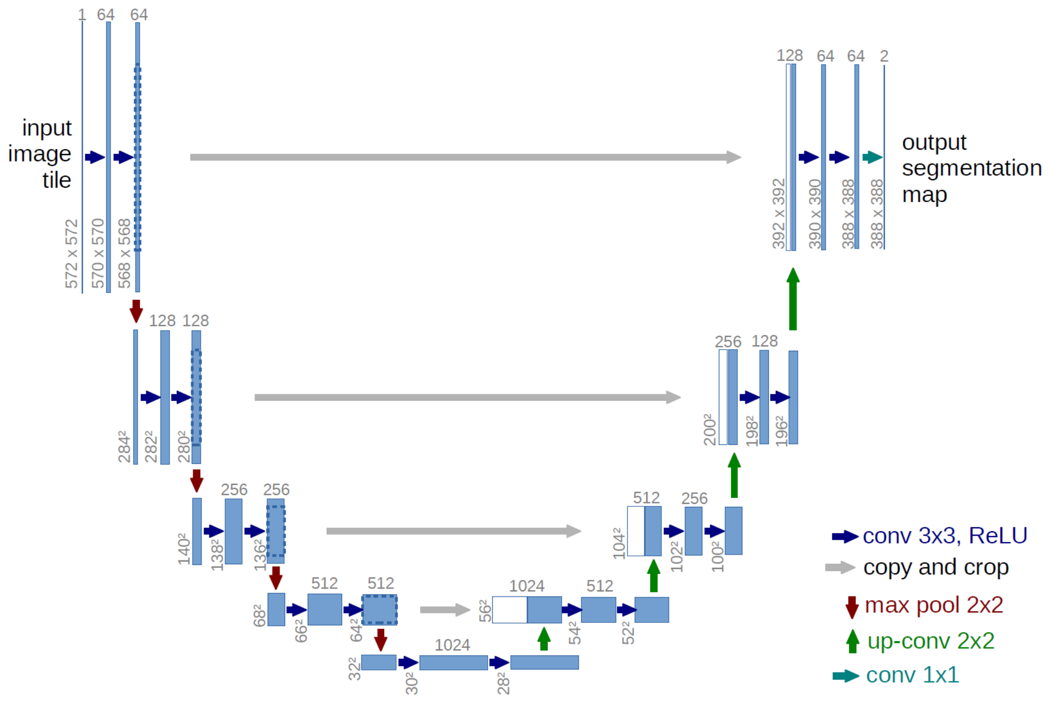


Figure 3.11: Illustration of the original U-Net generator implementation. *The figure is reprinted in unaltered form from the paper written by Olaf Ronneberge named "U-net: Convolutional networks for biomedical image segmentation"*[12]

Each process, in both the contracting and expanding path, forms two convolutional layers, illustrated as three rectangles. The input is a grayscale image with height and width equal to 572 pixels and a depth of one channel. The depth of the input image increases after a convolutional layer. In this structure, it extends from 1 to 64 after the first process. The size of the last image in the initial process shrinks due to padding issues related to the size of the kernels. The red arrow is the pooling process and halves the dimension of the height and width in the images. The convolutional and pooling process are repeated three times until the dimensions are 28x28x1024 (height, width and depth, respectively determined).

In the expanding path, the size of the image is upsampled to its original size using transposed convolutional layers. After the first layer, the image resized to 56x56x512. This image gets concatenated with the corresponding image on the contracting path. The two concatenated images have a combined size of 56x56x1024 and connected to receive a more accurate prediction. Same as in the contracting path, the transposed convolutional process is repeated three times. The last process has an extra filter with a kernel size of 1x1. The final kernel reshapes the image to the required size.

3.7 Metrics and Loss Function

A loss function, often referred to as a cost or error function, seeks to minimize a loss value to optimize a machine learning model. Metrics are used to judge the model's performance by measuring the similarity between the predicted and the expected output. This thesis uses binary segmentation, which corresponds to a segmentation task with two classes. An area with class labeled as one is *"true"* and defines a region of interest. An area with class label zero is *"false"* or negative, which corresponds to the background. All matrices and loss functions are explained based on binary segmentation tasks.

3.7.1 Pixel Accuracy

Pixel accuracy is perhaps one of the most straightforward matrices to estimate the model's performance. It compares each pixel in both the expected and predicted output and returns the percentage of correct pixel classification. The disadvantage of pixel accuracy appears when classes should be weighted differently. The illustration in figure 3.12 is a binary segmentation, which means that it contains two classes. The first class is the background visualized as black pixels, and the second class is the segmentation illustrated with white pixels. The left image corresponds to the expected segmentation, and the right image is the predicted segmentation. The number of pixels in each image is 100, and the expected segmentation has four white pixels and 96 black pixels. The predicted image has 100 black pixels.

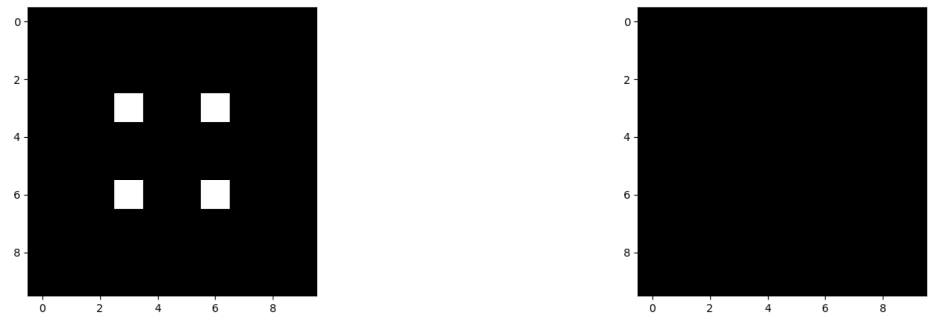


Figure 3.12: The left figure is the expected segmentation and the right figure is the predicted segmentation

The prediction will then be 96% correct, using pixel accuracy. This calculation is correct but useless to measure the model's performance, as 96% appears as a valid prediction.

3.7.2 Dice Coefficient

The dice coefficient is a weighted measure of the model's performance and commonly used in image segmentation tasks. Equation 3.8 describes the dice coefficient, where TP, FP, and FN correspond to true positive, false positive, and false negative pixels, respectively. TP is a region where the prediction and expected output are *true*, FP is a region where the prediction is *true*, and the expected output is *false*, FN corresponds to the region where the expected value is *true* and the prediction is *false*, and TN corresponds to the region where both are *false*.

$$\text{Dice coefficient} = \frac{2TP}{2TP + FP + FN} \tag{3.8}$$

The Dice coefficient exists in the range $[0, 1]$, where zero corresponds to a wrongly predicted segmentation, and one indicates that the prediction is a perfect reconstruction of the expected segmentation. [47]

3.7.3 Relative Absolute Volume Difference

The Relative Absolute Volume (RAV) difference is a metrics that measures the relative absolute volume difference between the expected segmentation and the predicted segmentation. RAV equal to zero corresponds to a perfect reconstruction of the relative absolute volume, but the segmentation could still be wrong as long as the volume is equal. [48]

3.7.4 Mean Surface Distance

The Mean Surface Distance (MSD) is a metrics that measures the distance between the predicted segmentation boundary and the expected boundary. Equation 3.9 describes the total surface distance d between the expected surface S and the predicted surface $\hat{S}$. The minimum of euclidean norm gives the distance between a point p on the expected surface S and the predicted surface $\hat{S}$. The goal when measuring surface distance on image segmentation is to get this value as low as possible. [49]

$$d(p, \hat{S}) = \min_{\hat{p} \in \hat{S}} ||p - \hat{p}|| \tag{3.9}$$

3.7.5 Hausdorff Distance

The Hausdorff Distance (HD) is a metrics that describes the longest distance possible to travel between one point on the expected output boundary towards the closest point on the predicted output boundary. This thesis uses the 95th percentile of the HD, which is more used on biomedical segmentations as it skips small outliers. [50]

3.7.6 Dice Loss

The dice loss is a value to optimize a machine learning model, that is equal to 1− Dice coefficient. This thesis uses a version that returns a single scalar for each image. The dice loss is defined in equation 3.10, where "I" corresponds to the expected output and "$\hat{I}$" is equal to the predicted output. [47]

$$DL(I, \hat{I}) = 1 - \frac{2\sum I_{h,w}\hat{I}_{h,w}}{\sum I_{h,w} + \sum \hat{I}_{h,w}} \tag{3.10}$$

3.7.7 Binary Crossentropy

The Binary cross-entropy is a loss value that optimizes a machine learning model. This loss is often used in correlation with models that use a sigmoid activation function at the output layer. The binary cross-entropy is defined in equation 3.11. The output range for binary cross-entropy is equal to $[0, \infty]$ [47]

$$CE(p, \hat{p}) = -(p\,log(\hat{p}) + (1-p)\,log(1-\hat{p}) \tag{3.11}$$

3.7.8 Mean Absolute Error

The Mean Absolute Error (MAE or L1) is a loss value that optimizes a machine learning model. This loss is often used for regression models. It calculates the sum of the absolute difference between the expected y and predicted output $\hat{y}$. The output range for MAE is equal to $[0, \infty]$ [47]

$$MAE = \frac{\sum_{i=1}^{n} |y_i - y_i^p|}{n} \tag{3.12}$$

3.8 Software

The technical part of this thesis is implemented with the programming language named Python. Python is a high-level, general-purpose programming language. A high-level programming language makes the process of developing simpler with the use of natural language and leaving hardware configurations to automate systems. A general-purpose programming language is used to develop a wide range of software applications [51].

Python does also use external libraries with additional premade functions. This chapter will describe some of the main libraries added to the implementation of this thesis.

3.8.1 Tensorflow

Tensorflow is an interface and implementation to express and execute machine learning algorithms. The library can be used to implement a wide variety of algorithms for a deep NN, like training and inference algorithms. [52]

3.8.2 Keras

The DL application programming interface (API) used in this thesis is Keras. Keras library uses Tensorflow to enable fast experimentation and implementation of DL ideas. [40]

3.8.3 Insight Toolkit

The Insight Toolkit (ITK) is an open-source library written in the programming language named C++, but supports multiple language bindings. The library is primarily used for medical image analysis, but can also be applied to other images. In this thesis, the library is used to read the medical file extension .mhd for the MRI images. The library is also used to implement the image preprocessing functions. [53] [54]

3.8.4 Numerical Python

The Numerical Python (NumPy) library is developed for Python programming language and supports high-level scientific computing and data analysis for numbers and multi-dimensional arrays. This library has been used within several functions in this thesis to generate random integers, generate arrays, and execute mathematical functions. NumPy is also used by several other libraries like Tensorflow, which utilizes the library to build the Tensor objects and more. [55]

Chapter 4

Dataset and Image Pre-Processing

4.1 Dataset

The dataset used in thesis work originates from the Prostate MR Image Segmentation 2012 (PROMISE12) challenge, with a submission deadline of June 29th, 2012. The dataset and submission for evaluation are still available in June 2020. Figure 4.1 shows a random MRI slice with the corresponding segmentation mask, obtained from the PROMISE12 dataset. [56][57]

The PROMISE12 challenge aims to improve interactive and (semi-)automatic segmentation algorithms that segment the prostate in transversal T_2-weighted MRIs. The system should be generalized and be able to segment the prostate from multiple centers and vendors data. [56]

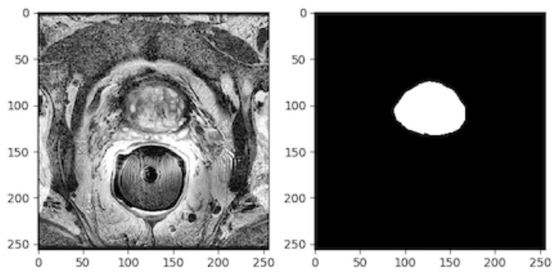


Figure 4.1: One transversal T_2-weighted MRI and the corresponding segmentation mask from the PROMISE12 dataset

The dataset consists of 50 cases for training and 29 cases to test the implementation. Each training case has one transversal T2-weighted MRI of one anonymous patient's prostate and a corresponding label. Each test case does only have the transversal T2-weighted MRI, as the label is hidden for the developer. The data originates from multiple medical centers with multiple vendors' equipment, where the data has different acquisition protocols (e.g., different resolutions), visualized in table 4.1. The label describing each MRI (cases) contains information of the actual location, size, and shape of the prostate. [56]

MRIs correspond to a stack of 2D images that represent a three-dimensional (3D) image when they are connected (see chapter 3.1). As the data has different acquisition protocols, the height, width, and depth are varying, making the data representative for medical clinics worldwide. The MRI and the corresponding label are stored in Meta (or MHD/RAW) format. This format saves an image as a file with the extension `.mhd`.

Number of cases (50)	22	1	25	2
Width (Pixels)	512	384	320	256
Height (Pixels)	512	384	320	256
Number of slices (1377)	760	28	548	41

Table 4.1: Table showing the number of MRI slices and the correlated image size.

4.2 Image Pre-Processing

This section presents and discusses all the pre-processing techniques used in this thesis. The implementation to load and pre-process the PROMISE12 data is inspired by Inom Mirzaev's work on the PROMISE12 challenge [14]. The list below mentions the pre-processing steps involved in this thesis.

- Rearrange data
- Reshape data
- Separating slices and organize data
- Store organized data
- Filtering

4.2.1 Rearrange Data

The first step rearranges images to sort them by patients and to create a validation set. Each case is named `Case[num]` where `num` increase from `00` to `50`. Since the available test data provided by PROMISE12 has hidden labels, a validation set is manually selected from the training set. The training set has 45 cases after transferring five cases, numbered 5, 15, 25, 35, and 45 to the validation set. The validation set validates the precision of the model after each epoch of training.

4.2.2 Reshape Data

A CNN must train on images with equal size. The same network can be used to train images of different sizes, but not at the same time. It is common to design the CNN to fit the shape for the desired dataset to get the best result. The dataset used in this thesis has images with height and width, ranging from `256x256` to `514x514`. Two commonly used methods to align the image shape are padding or reshaping. This thesis reshapes all images to have height and width equal to `256x256` for both the U-Net and the generative models.

4.2.3 Separating Slices and Organize Data

The chosen architecture to test the proposed method is the 2D segmentation network named U-Net. Each 2D slice in all the original 50 cases (both training and validation set) is separated and represents an individual case to fit the U-Net architecture, resulting in 1.377 (see table 4.1) new cases. The original 50 cases include the whole prostate gland, but not on the first and last slices, resulting in some slices without the gland. The total amount of slices including the prostate gland is 697.

4.2.4 Save Organized Data

DL networks depend on fast computation, even with a small dataset. The NumPy library provides that advantage, and saves the data as a four-dimensional NumPy array in a file with extension `.npy`. The first index of the array is the number of cases. The number of cases in each category is described in table 4.2. The two next indexes correspond to the image's width and height, and the last index describes the depth (number of channels) of the image. The channel is equal to one for all cases in this thesis as the images are grayscale. The final shape of the NumPy array is `(number of slices, height, width, channels)`.

Data	**Number of slices**	**Shape**
All slices (Training set)	1250	(256, 256, 1)
All slices (validation set)	127	(256, 256, 1)
Cases including the prostate gland (training set)	697	(256, 256, 1)
Cases including the prostate gland (validation set)	81	(256, 256, 1)

Table 4.2: The number of slices with and without the prostate gland in the training set and validation set.

4.2.5 Filtering the Data

The original data is dark and hard to extract features from, as visualized in figure 4.2. The MRIs are pre-processed using three filters named normalization, outlier removal, and contrast equalizer.

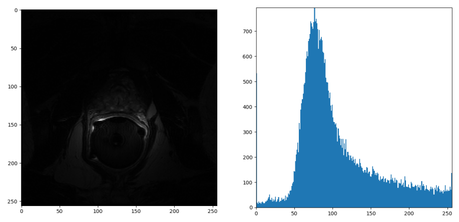


Figure 4.2: Visualizes a random example from the original dataset without any filters and the corresponding histogram

A grayscale image contains pixel values ranging from 0 to 255, where 0 corresponds to black and 255 to white. Normalization divides the pixel values by 255, resulting in images with pixel values ranging from 0 to 1. Normalization does also normal distribute the pixel histogram, making the mean of the original pixel values the most represented shade. Figure 4.3 is the same example as shown in figure 4.2 with the normalization filter.

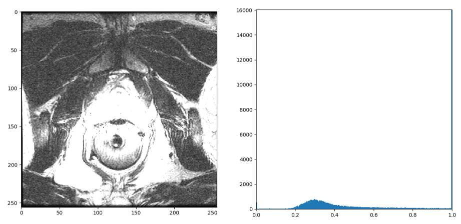


Figure 4.3: Visualizes the same example as shown in figure 4.2, including the normalization filter.

The calculation of linear normalization on an image (I) is described in equation 4.1, where *Min* and *Max* is the old minimum and maximum pixel values. The value *newMin* and *newMax* correspond to the desired minimum and maximum values.

$$I_N = (I - Min)\frac{newMax - newMin}{Max - Min} + newMin \tag{4.1}$$

The outlier removal filter has two parameters, pixel intensity maximum, and minimum. The minimum replaces the 1% darkest pixel value with values similar to the neighbors. The maximum is not defined and include all the brightest pixel values. Figure 4.4 visualizes the same example as shown in figure 4.2 with just the outlier removal filter.

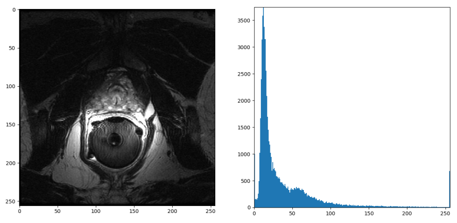


Figure 4.4: Visualizes the same example as shown in figure 4.2, including the outlier removal filter.

The contrast equalizer used in this thesis is the *Contrast Limited Adaptive Histogram Equalization (CLAHE)*. CLAHE is an algorithm to perform local contrast enhancement that calculates several histograms, which correlates to a distinct part of the image and redistributes the pixel values. This method enhances local details without overamplifying the noise, even in dark or bright regions [58] [59]. This thesis defines one parameter named `clip_limit` to 0.05. The `clip_limit` parameter is a contrast factor to prevent oversaturation and must be in the range $[0, 1]$, where a higher value results in stronger contrast. Figure 4.5 visualizes the same example as shown in figure 4.2 with just the CLAHE filter.

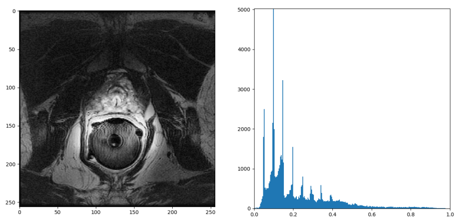


Figure 4.5: Visualizes the same example as shown in figure 4.2, including the CLAHE filter.

Figure 4.6 visualizes the same example as shown in figure 4.2 printed with all filters from the `.npy` file.

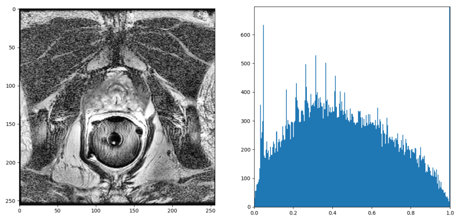


Figure 4.6: An MRI slice processed with normalization, outlier removal, and CLAHE filter.

4.3 Each Models Input and Output

This thesis explores several CNNs with different input and output data. All networks in this thesis use the resized and separated PROMISE12 data pre-processed with normalization, outlier removal, and CLAHE. The generative networks are only training on data that includes the prostate gland, while the evaluation network U-Net uses all data. The input data for U-Net and the generative networks are scaled differently to fit the model's output activation. The U-Net data is scaled in the range $[0, 1]$ to fit the Sigmoid activation function while the data used in the generative networks are scaled in the range $[-1, 1]$ to fit the TanH activation function.

DCGAN is an unsupervised network using only segmentation masks as input. The Pix2Pix generator is only using segmentation masks as input, while the discriminator uses both the segmentation mask and a MRI to predict the likelihood and classify the prediction. U-Net is a supervised network, including both the MRIs and segmentation masks. The MRI correspond to the U-Net input and the segmentation mask corresponds to the expected prediction. The generative networks use hard labels to describe whether the data fed into the discriminator is generated or real. Hard labels are an array with the same shape as the input image, where all numbers in the array are either one or zero. Generative networks, in general, do also flip their labels, depending on the type of model training. The labels defined for the discriminator training process have class labels equal to one for real images, and class labels equal to zero for generated images. These labels are flipped when the generator trains, making the class label equal to one for generated images and zero for real images. Table 4.3 visualizes an overview of the data used to update weights for each model.

Model	Input	Output	Evaluation
DCGAN discriminator	Real and generated segm. masks	Classification	Label 1: real segm. mask Label 0: fake segm. mask
DCGAN generator	Gaussian distributed random values	Fake segm. masks	Discriminator feedback. Label 1: fake Label 0: not fake
Pix2Pix discriminator	Real segm. mask Real and fake MRIs	Classification	Label 1: real MRIs Label 0: fake MRIs
Pix2Pix generator	Real segm. mask	Fake MRIs	Discriminator feedback and L1 loss. Label 1: fake Label 0: real
U-Net segmentation	Real and fake MRIs	Predicted segm. mask	Dice loss to compare predicted and expected segm. mask

Table 4.3: Input and output values for each model utilized in this thesis.

Chapter 5

Solution Approach

5.1 Introduction

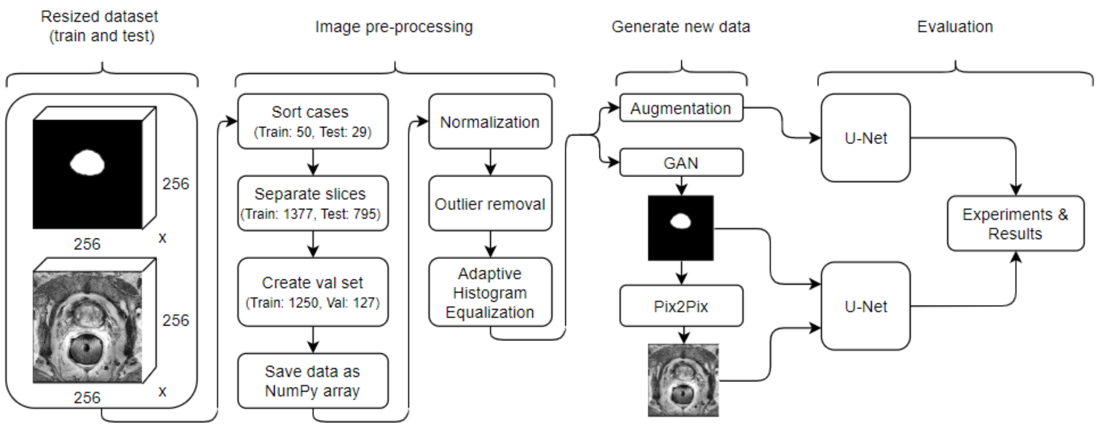

Resized dataset (train and test)
256
256
x
256
256
x
Image pre-processing
Sort cases (Train: 50, Test: 29)
Separate slices (Train: 1377, Test: 795)
Create val set (Train: 1250, Val: 127)
Save data as NumPy array
Normalization
Outlier removal
Adaptive Histogram Equalization
Generate new data
Augmentation
GAN
Pix2Pix
Evaluation
U-Net
U-Net
Experiments & Results

Figure 5.1: Overview of the proposed approach.

5.1.1 Existing Approaches

The paper *Medical Image Synthesis for Data Augmentation and Anonymization using Generative Adversarial Networks* written by Hoo-Chang Shin et al., uses Pix2Pix to generate MRIs of the brain. This paper uses real segmentation masks and is not generating new ones. The segmentation masks used in this paper have multiple classes and a black background surrounding the brain.

A master thesis with the title *Data augmentation in deep learning using generative adversarial networks* written by Thomas Neff for the Graz University of Technology uses one generative network named Wasserstein GAN (WGAN) [60]. This thesis generates both X-radiation (x-rays)[1] images of the lung and the corresponding segmentation mask using only the WGAN. Thomas Neff was able to generate image-segmentation pairs instead of just x-rays, but the model and evaluation setup showed improvement possibilities. [62]

5.2 Proposed Method using GAN to Expand the Dataset

The testing network named U-Net requires both a source image (MRI) and a target image (segmentation mask) to learn features and to segment the whole gland of the prostate from new cases. In this thesis, a two-step approach has been developed to extend the training set, utilizing two GAN architectures that build on top of each other. The first architecture is DCGAN that generates segmentation masks, and the second is Pix2Pix that generates synthetic MRIs based on these masks. Both networks contain a generator and a discriminator, designed as two models that update weights separately. DCGAN and Pix2Pix do only train on data including the patient's prostate gland, which narrows the dataset to 697 samples (See table 4.3 for an explanation of input and output associated with each model).

5.2.1 DCGAN Methodology

DCGAN is chosen to generate new segmentation masks, as the training process has proven to be more stable than the original GAN [7] [9].

DCGAN trains for 1500 epochs, where each uses about 100 seconds on the GPU named Nvidia Tesla P100 with 16GB. The GAN generator uses the discriminator to measure performance and no evaluation matrices, making it hard to know when the training process should stop, and when the samples are realistic. The network train for 1500 epochs, but share visual examples and model weights when the discriminator observes improvement. To run a DCGAN for 1500 epochs is an abnormally large amount, only reasonable as the model trains on a small dataset (see table 4.2). The batch size does only contain 32 samples, even when the original paper recommends 128. This batch size is smaller than the recommendation, as the dataset is rather small, with 697 MRIs compared to the dataset used in the original paper with 3 million images. All models train using Adam optimizer with a learning rate equal to 0.0002 and β_1 equal to 0.5, as recommended in the original paper.

[1]X-rays are high-energy electromagnetic radiation that can be used to take a picture of the body's internal structure. [61]

Parameter	**Value**
Epochs	1500
Batch size	32
Optimizer	Adam(Learning rate = 0.0002, $\beta_1 = 0.5$)

Table 5.1: DCGAN hyperparameters

Model Design

The DCGAN implementation is inspired by Jason Brownlee's work on DCGAN for grayscale handwritten digits [35, page 95 - 131] and the original DCGAN architecture [7]. Jason Brownlee's work on DCGAN generates images with 28x28 pixels, and the original paper generates images with 64x64 pixels. This thesis is based on these previous implementations, but the architecture is modified to generate images with 256x256 pixels and to stabilize the training process.

The DCGAN discriminator is an image classification model that learns the difference between real and synthetic segmentation masks. A visual illustration of the model is shown in appendix A figure A.1. The implementation of the discriminator is equal to the original but extended with two convolutional layers. The output is a single neuron with the Sigmoid activation function, and the loss function is a binary cross-entropy, as recommended.

- **Discriminator input:** Segmentation masks (real or synthetic) with 256x256 pixels and one channel.
- **Discriminator output:** Binary classification, it determines if the input is real or synthetic.

The DCGAN generator expands the dataset with synthetic segmentation masks. The model input corresponds to a 100 elements vector of Gaussian distributed random numbers. A visual illustration of the model is shown in appendix A figure A.2.

- **Generator input:** A vector containing 100 elements of Gaussian distributed random numbers.
- **Generator output:** Segmentation masks with 256x256 pixels and one channel.

The two first layers of the generator, dense and reshape, transform the input vector into a 2D array. The dense layer has 262144 neurons representing several parallel and different low-resolution versions of the output image. The calculation of neurons is described in equation 5.1.

$$neurons = 4096 * \frac{256}{2^5} * \frac{256}{2^5} \tag{5.1}$$

The activation of these neurons is reshaped to 4096 different 8x8 pixels feature maps. The original paper follows the same approach but with different dimensions. DCGAN uses a transposed convolutional layer to learn weights while upsampling the input. This report's implementation adds two transposed convolutional layers with a stride equal to (2, 2) and a kernel size of (4, 4). The kernel size is a factor of the stride to avoid a checkerboard pattern after upsampling, and the stride quadruples the input size. The original paper recommends ReLU activation for all generator layers except for the output, while this thesis uses the softer activation named LeakyReLU, as recommended in Soumith Chintala's talk on NIPS 2016 [63, after 11 minutes]. The output convolutional layer merges all feature maps to one final and uses a TanH activation function to ensure output values in the range $[-1, 1]$.

The generator's weights are updated based on the discriminator's feedback in a new logical model. This model provides a 100 elements vector of Gaussian distributed random numbers to the generator's input and feeds generated segmentation masks to the discriminator. The discriminator classifies each sample and returns a score used to update the generator weights. The logical model combining the generator and discriminator is compiled with the binary cross-entropy loss as recommended.

5.2.2 Pix2Pix Methodology

Pix2Pix is chosen to generate MRIs with a gland, described by the generated segmentation mask, due to the promising results on generating images from labels [10] [46]. The network trains on both labels and MRIs obtained from the PROMISE12 dataset, but only cases including the gland.

The network is trained for 200 epochs as recommended in the original paper, where each uses up to 300 seconds on the GPU, named Nvidia Tesla P100 with 16GB. The batch size is equal to 1, as recommended. The generator is a modified version of the U-net architecture with a contracting path (encoder) and an expanding path (decoder). A batch size of 1 can be inappropriate for the generator's encoder-decoder transition, as the batch normalization zeros the activation on the bottleneck layer [10]. Removing the batch normalization for the bottleneck layer solves this problem [35]. Like DCGAN, all models in the Pix2Pix network train using Adam optimizer with an initial learning rate of 0.0002 and β_1 equal to 0.5, as recommended [10].

Parameter	**Value**
Epochs	200
Batch size	1
Optimizer	Adam(Learning rate = 0.0002, $\beta_1 = 0.5$)

Table 5.2: Pix2Pix hyperparameters

Model Design

The Pix2Pix implementation is inspired by Jason Brownlee's work *"How to Develop a Pix2Pix End-to-End"* [35, page 466 - 516] and uses the same model architecture as the original Pix2Pix paper [10]. The implementation is modified to stabilize the network and to fit the PROMISE12 dataset. Jason Brownlee's work uses the Keras DL framework as this thesis, while the original implementation uses the Torch. Both of these previous implementations are designed for colored images, instead of grayscale as this thesis.

The Pix2Pix discriminator is a PatchGAN model that learns different features from synthetic and real MRIs by evaluating 70x70 pixels patches of the image. The classification is an average of all patches in each MRI. A visual illustration of the model is shown in appendix A figure A.3

- **Discriminator input:** Segmentation masks (real or synthetic) and corresponding MRI (real or synthetic)
- **Discriminator output:** Binary classification, it determines if the input is real or synthetic.

The discriminator input concatenates a segmentation mask and the corresponding real or synthetic MRI in the channel dimension. The model has six convolutional layers with filters doubling for each layer. The first four convolutional layers have a stride equal to $(2, 2)$, instead of max pooling; and the last two have a stride equal to (1, 1). Strided convolutional layers are used instead of pooling processes to allow learning of weights while downsampling. All convolutional layers use the LeakyReLU activation except the last one, which uses the Sigmoid to classify the MRI as real or synthetic. The discriminator model is compiled with the binary cross-entropy loss and Adam optimizer, as recommended in the original paper.

The generator used in this thesis does also follows the same design as the original paper. A visual illustration of the model is shown in appendix A figure A.4. A GAN generator should generate images that are similar to the training set but uncorrelated. DCGAN introduces randomness by starting with a 100 elements vector of Gaussian distributed random numbers, while the U-Net generator's input is a segmentation mask. Dropout layers are applied to replace the missing randomness in the input.

- **Generator input:** Segmentation mask with 256x256 pixels and one channel.
- **Generator output:** Generated MRI of the input segmentation mask.

The U-Net generator has an encoder and decoder path (see chapter 3.6). The encoder has seven convolutional layers with strides equal to $(2, 2)$. All convolutional layers, except the first layer in the encoder path, are followed by the LeakyReLU activation. The decoder has seven transposed convolutional layers with stride equal to $(2, 2)$. The decoder layers are concatenated with the encoder layer at a corresponding depth of the U-shape and processed with a LeakyReLU activation (see figure 3.11 for the original U-Net architecture, explaining the concatenation). The output is a transposed convolutional layer that merges all feature maps to a final one and uses a TanH activation function to ensure output values in the range $[-1, 1]$. The generator's weights are updated based on the discriminator's feedback in a new logical model, similar to the DCGAN implementation. This model transfers a segmentation mask from the dataset to the generator's input and feeds generated MRI to the discriminator. The discriminator classifies a 70x70 patch of the input, averages all patches, and classifies each MRI. The discriminator returns a score used together with MAE loss to update the generator weights. The total loss is calculated as equation 5.2, where L1 acts like a regularizer weighted with λ.

$$loss = adversarial\_loss + MAE * \lambda \tag{5.2}$$

Some additional features introduced in Soumith Chintala's talk on NIPS 2016 [63] have been implemented to make the network more stable. The first feature separates real and synthetic training data for the discriminator. According to Soumith Chintala's talk, this will improve the result compared to training the discriminator on both real and synthetic data at the same time. Two label features called label smoothing and noisy labels were implemented in order to avoid the discriminator to over-perform. Label smoothing changes the label equal to 1 for real samples to randomly be in the range of $[0.7, 1]$. Usually, labels are always correct when training the discriminator, but noisy labels fool the discriminator by changing some of the labels for fake images to be real. These features make it easier for the generator to improve results and generate more realistic examples.

Chapter 6

Experimental Evaluation

This chapter presents the experimental setup, the experimental result, and performance scores using U-Net. This chapter does also includes visual examples and an evaluation of the result generated by DCGAN and Pix2Pix.

6.1 Experimental Setup

The U-Net model used to test the proposed approach is reprinted from the original paper written by Olaf Ronneberge et al. for Biomedical Image Segmentation [12], but extended with two convolutional layers in both the contracting and expanding path to improve the result. The code to train the U-net model and the model itself is implemented from scratch, but the code to print the U-Net performance score is inspired by Inom Mirzaev's work on the PROMISE12 prostate MRI segmentation challenge [14]. See chapter 3.6 for a detailed explanation of the U-Net architecture.

The loss function used to optimize the U-Net model is Dice loss, and the metric Dice coefficient is used to evaluate the model performance during training. The implementation of dice loss and dice coefficient is reprinted in unaltered form from Lars Nieradzik's implementation [47]. The network trains for 200 epochs with a batch size of 32, and the model uses Adam optimizer with an initial learning rate of 0.001 and a learning rate scheduler, that reduces the learning rate with the factor of 0.1 if the validation loss does not decrease after ten epochs. The minimum achievable learning rate is $1 * 10^{-09}$. The network monitors the validation loss and saves the best weights only.

The proposed method to improve prostate segmentation on MRI changes the U-Net training data. The model is also tested using the original data as a baseline and augmented data to compare the proposed method with an existing approach. U-Net trains on each

Parameter	**Value**
Epochs	200
Batch size	32
Optimizer	Adam(Learning rate = 0.001)

Table 6.1: U-Net hyperparameters

group of input data separately and saves the weights with the lowest loss. The loss graph for each test is included in appendix B.

The weights with the lowest loss for each training data group are loaded to the U-Net model and tested on the validation set, using the same metrics as the official PROMISE12 grand challenge [56]. The validation set is represented as five volumetric MRIs, where all MRI slices for each patient are merged in the original order. All performance metrics, except the dice score, are a measure of the volumetric performance instead of the score for each slice individually. This data represents a realistic scenario for real-world cases. The implementation for all evaluation metrics is obtained from online sources and cited after each function. The metrics are the Dice coefficient (DSC) [47], the relative absolute volume difference (RAV) [14], the mean surface distance (MSD) [48], and the Hausdorff distance (HD) [48].

6.2 Experimental Results

This section focuses on the experimental results and the correlated score for original, augmented, and synthetic data. It includes visual examples of augmented and generated images and the performance using the U-Net model.

6.2.1 Result of Standard Image Augmentation

This thesis uses the Keras built-in function for image augmentation [44]. The augmentation methods applied to analyze the segmentation result are shift, rotation, flip, and zoom. The shift augmentation moves all pixels in the range of zero to 10% of the total image height and width. The rotation parameter rotates the image 10 degrees from the original angle. A horizontal and vertical flip is applied, and the zooming range is defined as $[1, 1.2]$. These augmentations are the same as Inom Mirzaev used on the PROMISE12 challenge implementation, with the exception that their work includes a method called elastic transformation. Figure 6.1 visualizes five augmented examples from the PROMISE12 dataset.

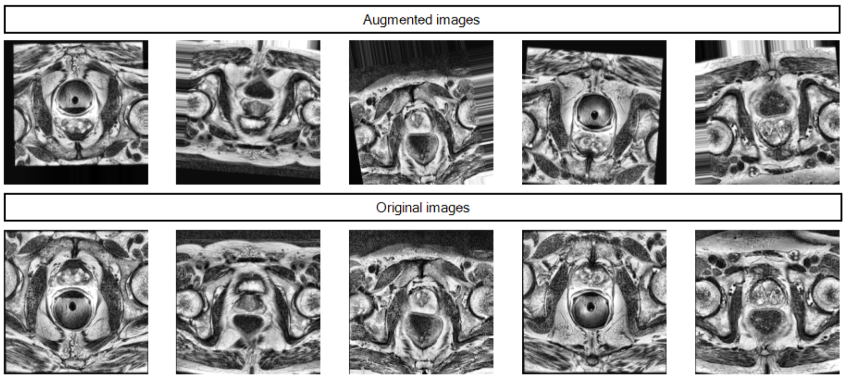


Figure 6.1: Illustration of augmented images (upper row) and the related original images (under each augmented image)

Tabel 6.2 describes the performance score for the U-Net architecture trained on original data, the data with all augmentations applied, and each augmentation separately. The original data has a dice score of 0.693, which is a week representation of the expected segmentation mask. A RAV difference of 88.604, HD of 8.160 mm, and MSD of 23.041 do also describe a week representation of the expected segmentation mask. The data augmented with all techniques have a dice score of 0.668, which is slightly less than the original data. The RAV and MSD have improved significantly to 27.358 and 13.138, respectively.

Val. data	**Mean DSC**	**std. DSC**	**Mean RAV**	**Mean HD**	**Mean MSD**
Org. data	0.693	0.077	84.804	8.160	23.041
Vert. & horiz flip	0.694	0.005	99.298	19.156	35.517
Rot.	0.711	0.121	54.970	9.556	5.063
Shift	0.732	0.164	27.349	16.148	1.946
Zoom	0.682	0.043	87.343	8.317	27.383
All	0.668	0.146	27.358	15.329	13.138

Table 6.2: The U-Net segmentation performance score for original data, data with each augmentation separately, and the data with all augmentations applied

The approach proposed in the paper of Inom Mirzaev, *"Fully convolutional neural network with residual connections for automatic segmentation of prostate structures from MR images"* [14] uses augmentation to create 150k new versions of the data for each epoch. This approach has been tested in this thesis and is described in table 6.2. This method improves the dice score, RAV, and especially MSD, but the HD has increased with 15.08 mm. The result published in the paper of Inom Mirzaev, had significantly better results than these ones, with a dice score of 0.885. [14]

Val. data	**Mean DSC**	**std. DSC**	**Mean RAV**	**Mean HD**	**Mean MSD**
Org. data	0.693	0.077	84.804	8.160	23.041
All augmentation and data augmented to 150k sampels.	0.726	0.120	38.746	23.240	0.947

Table 6.3: The performance score for the dataset expanded to 150k samples using just standard image augmentation

6.2.2 Generative Results using DCGAN and Pix2Pix

GANs are becoming a common approach for developing generative models using DL, but these networks do not have a loss function to minimize nor an expected output to evaluate the performance using metrics. This thesis uses two approaches to evaluate the GAN Generator, a manual evaluation and the use of the segmentation network U-Net.

DCGAN Results

Firstly, the segmentation masks were generated using DCGAN. Out of the 2000 generated examples, 1348 of them were deemed usable and 652 not. Some of the unusable segmentation masks had more than one gland and shape that differs from the original samples in the PROMISE12 dataset. Figure 6.2 visualizes a random sample of the wrongly predicted segmentation masks, which were manually removed.

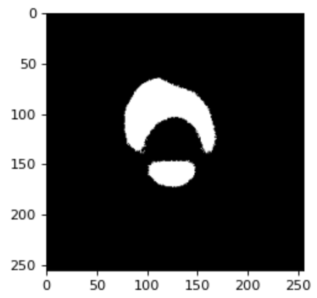


Figure 6.2: Most of the predicted samples have a realistic shape, but some are unrealistic and have more than one prostate gland and a shape that divides from the original samples in the PROMISE12 dataset.

The synthetic segmentation masks are saved in a separate file than the original and are utilized to generate synthetic MRIs with Pix2Pix. Figure 6.3 visualizes four correctly predicted examples.

Figure 6.4 shows the DCGAN generator loss. The best weight was saved at epoch 55, but the samples printed after 1500 epochs seem more realistic from a human perspective. See appendix C for samples at epoch 24, 33, and 55.

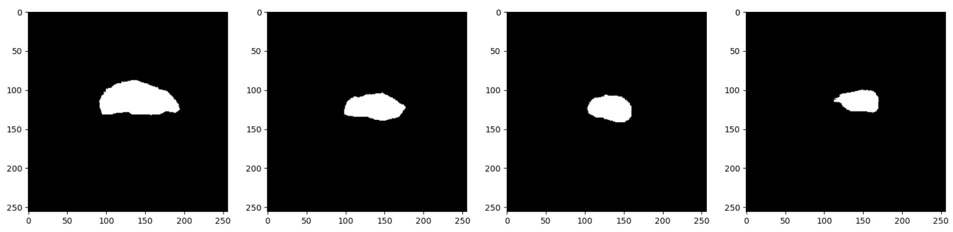


Figure 6.3: DCGAN generated segmentation mask after 1500 epochs

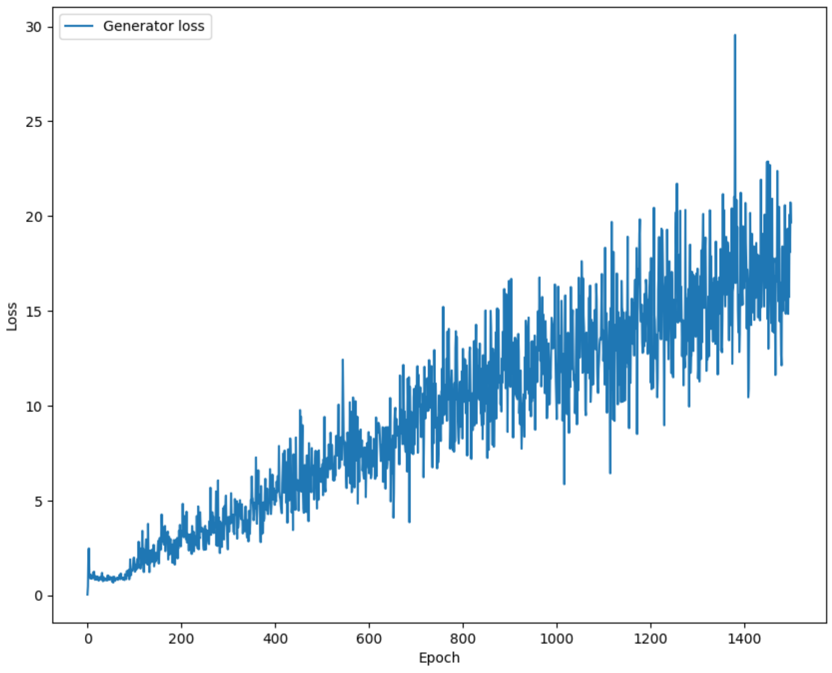


Figure 6.4: DCGAN generator loss for 1500 epochs

Pix2Pix Result

The Pix2Pix architecture generates synthetic MRIs based on the generated segmentation masks from DCGAN. A total of 1348 MRIs are stored as a separate file in the same order as the corresponding synthetic segmentation masks. Although some examples are not completely realistic from a human perspective, all are used to train the U-Net model. Figure 6.5 shows four examples where the bottom row represents the generated MRIs, and the upper row represents the corresponding generated segmentation mask. Figure 6.5 represents samples printed at epoch 162, which got the lowest loss during training (see figure 6.6). Generated samples from epoch 10, 50, and 200 are included in appendix D. The region surrounding the prostate gland has fewer details than the real MRIs, as the model tries to get the best possible score for all images; which results in averaging the pixel intensity of all training data with a similar segmentation mask. This issue is not necessarily harming the U-Net segmentation if the region of the gland seems real.

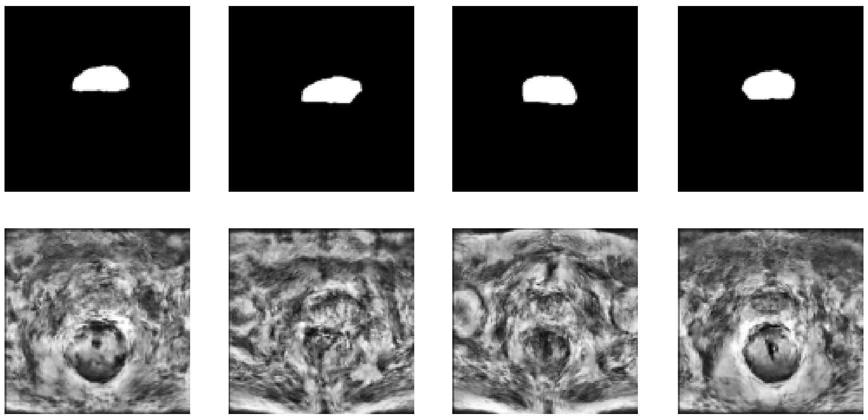


Figure 6.5: Pix2Pix generated MRIs and the corresponding input segmentation mask, generated by DCGAN. The MRIs were generated after 162 epochs and had the lowest generator loss during training.

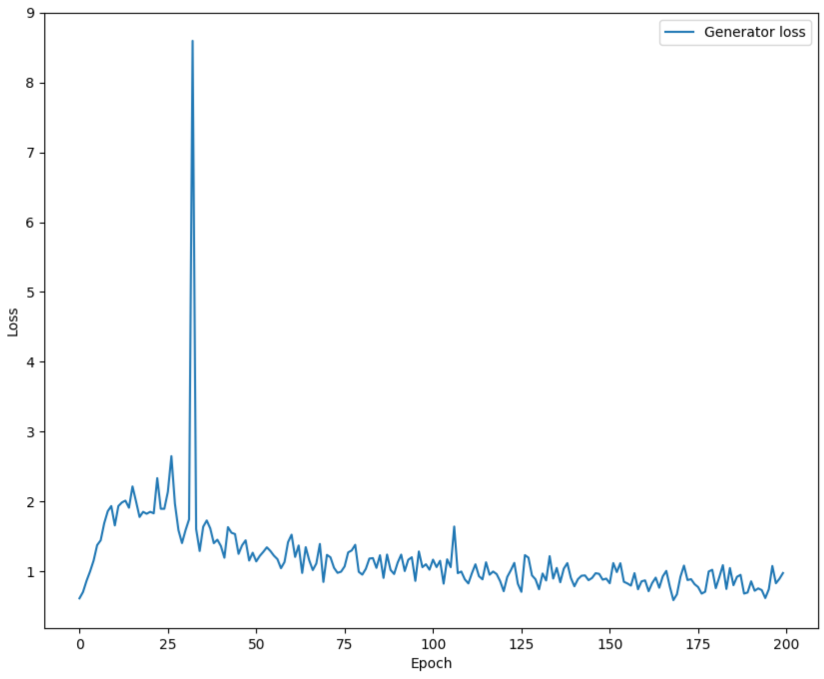


Figure 6.6: The Pix2Pix generator loss for 162 epochs

Test Result

The synthetic data are concatenated with the original and tested using the U-Net architecture. Table 6.4 describes the result for synthetic segmentation masks generated at epoch 1500 and synthetic MRIs generated at epoch 162. The test result for segmentation masks generated at epoch 1500 and MRIs generated at epoch 50 is included in appendix E. Tabel 6.4 shows that the dice score without any augmentation has improved to 0.731, compared to only using the original data. The RAV has decreased significantly to 30.885, but it is still not considered as a good result. HD has increased with 6.879 mm, while the mean MSD decreases to the promising result of 1.286 mm. The generated and original data combined does not seem to take advantage of standard image augmentation, as only the rotation augmentation improve the dice score.

Val. data	**Mean DSC**	**std. DSC**	**Mean RAV**	**Mean HD**	**Mean MSD**
Org. and & gen. data	0.731	0.108	30.885	15.039	1.286
Vert. & horiz flip	0.710	0.1593	29.322	12.985	1.611
Rot.	0.735	0.137	33.069	12.879	1.793
Shift	0.717	0.104	26.267	13.181	1.929
Zoom	0.690	0.142	26.191	11.161	2.458
All	0.673	0.167	29.417	17.366	2.047

Table 6.4: The performance score for real and generated data combined. The score is calculated for synthetic segmentation masks printed after **1500 epochs** and synthetic MRIs printed after **162 epochs**.

Figure 6.7 shows how the U-Net architecture predicts the gland region from random samples of the validation set, using weights trained on original data extended with synthetic data. This figure shows an example where the predicted region of the prostate gland is nearly a perfect reconstruction of the expected segmentation mask. More examples are included in appendix G.

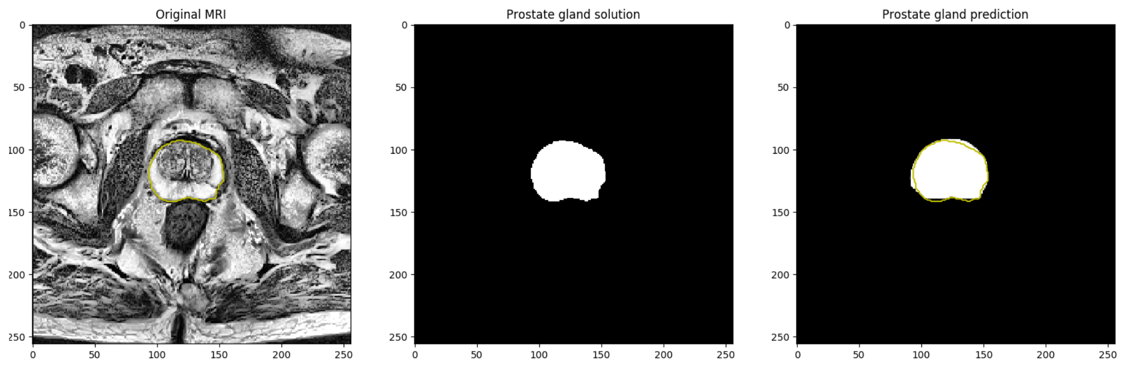


Figure 6.7: The U-Net segmentation result on a random MRI from the validation set after training on real and generated data combined.

Chapter 7

Discussion

This chapter presents a discussion of the proposed approach, the achieved results, a comparison with related work, the evaluation process, and the limitations.

7.1 Standard Augmentation

Data augmentation for deep NN presents a useful and straightforward method to extend the dataset artificially. Since augmentation rearranges the pixels enough for the model not to recognize the original MRI, it is important to preserve key features. Augmentation will be counterproductive if the training data is too dissimilar from the test data, making it essential to analyze the training data to find the best augmentations that preserve the key features of a real MRI. The analysis process is time consuming and it requires previous knowledge about the data.

As mentioned before, selecting the best augmentation techniques is time-consuming. Hereby, this thesis builds upon the work of Inom Mirzaev's work on *"Fully convolutional neural network with residual connections for automatic segmentation of prostate structures from MR images"* [14], as his results are promising.

7.2 Generative Approach

The quality of the generated segmentation masks is considered good, even though only 1348 of the total 2000 were deemed usable. Some examples had two regions of the prostate gland or a shape that differs from the original dataset. The unusable data could be realistic for a radiologist, but this thesis has only compared the generated samples with the PROMISE12 data. The generated MRIs have similarities to real MRIs with a region containing the prostate gland and similar pixel intensity. However, the MRIs have improvement possibilities as they are easy to classify for a human.

The result of training a biomedical segmentation model on the combination of original and synthetic data is promising, as the performance is higher when combining than when using original data only. The generated data does not take advantage of using data augmentation as the performance is lower for all augmentations, except for rotation. The reason might be that the synthetic data's main features are the location and shape of the gland.

7.3 Evaluation Process

This thesis's primary focus has been to develop a method to extend the dataset with synthetic data using GAN. The combination of two GAN architectures stacked on top of each other proved to be sensitive to instability, and required much testing. More time was spent on the two generative models and less on the segmentation model. The overall performance score is not great if compared to the PROMISE12 scoreboard [64], where the leading team has a dice score of 0.913 [65]. However, this thesis's goal is not to beat the above mentioned scoreboard, but rather to increase the performance level with the help of GAN generated data. The segmentation model is the same for the original data, augmented data, and synthetic data combined with the original data, and should be suitable for comparing the results.

7.4 Comparisons With Related Work

To the best of my knowledge, there is no published work on generating MRIs and corresponding segmentation masks of the prostate using DCGAN and Pix2Pix.

The paper *"Medical Image Synthesis for Data Augmentation and Anonymization using Generative Adversarial Networks"* written by Hoo-Chang Shin et al., shows a promising result on generating synthetic MRIs of the brain from real labels as an anonymization tool using Pix2Pix. Their approach depends on detailed multiclass segmentation masks, where the final MRI has a black background. The PROMISE12 segmentation masks have only two classes, describing the location and shape of the prostate gland and the background.

Although there is no published work on the generative approach, others have developed an automated segmentation system for the prostate gland using the same dataset. The leading team on the PROMISE12 scoreboard [64] proposed a new network to segment the prostate gland in MRIs. The network is named HD-Net and described in the paper *"HD-Net: Hybrid Discriminative Network for Prostate Segmentation in MR Images"* written by Haozhe Jia et al. in 2019 [65]. Their work on HD-Net consists of a 3D segmentation decoder that captures volumetric features and a 2D boundary decoder pointing the focus on the semantically discriminative intra-slice features [65]. The second team with a slightly lower score than the leading one, Fabian Isensee et al., published their work in the 2020 paper *"Automated Design of Deep Learning Methods for Biomedical Image Segmentation"* [66]. They used a DL network named nnU-Net, which enables 3D segmentation and adapts automatically to many biomedical datasets without requiring a specialized design for each dataset. The HD-Net and nnU-Net architecture are far more advanced than the original U-Net architecture. These networks are deeper and extract really specific features from different parts of the prostate. The achieved result for the leading team, and the second team are presented together with this thesis result in table 7.1. As mentioned earlier, the main reason to present these other results together with the current thesis' ones is not to perform better than them, but to have an idea of the results explored in this thesis work compared with others.

	Mean DSC	**Mean RAV**	**Mean HD**	**Mean MSD**
This thesis using synthetic data (Validation set)	0.731	30.885	15.039	1.286
HD-Net (Official test set)	0.913	5.0133	3.933	1.361
nnU-Net (Official test set)	0.919	3.304	3.950	1.243

Table 7.1: Comparisons of segmentation result from previously published papers and this thesis' proposed approach.

7.5 Limitations

This section will discuss some limitations of this project.

7.5.1 Data

The PROMISE12 dataset contains 1377 slices from 50 patients, where 697 slices include the prostate gland. For a CNN and especially GAN, this dataset is considered small and will limit the performance. The data is cross-center and from multiple vendors' equipment, which can be considered a strength while working with the segmentation part of this thesis and will improve the generalization of the U-Net model. For GAN, this is expected to limit the performance as wide distribution of dissimilar data is harder to recreate. A large volume of biomedical data can be challenging to obtain, but a slight increase in the dataset could improve all models' performance. The use of center-specific data could potentially improve GAN performance and result in a better classification score for the synthetic data combined with the original one.

7.5.2 Augmentation

The generative approach was tested using a U-Net segmentation model and compared with the original and augmented data. The original data represent the baseline, and the augmented data is introduced to compare the proposed method with an existing one. The data augmentation was implemented with inspiration from a previous publication, without knowledge regarding the scientific conclusions behind the chosen augmentations. The potential of radiologist-based opinions and thus following their recommendations for augmentation techniques could have improved the final results for this thesis.

7.5.3 Network Performance

Generative models like GAN are considered challenging to train, compared to classification and object detection models. Within the time constraints of this project, three different DL networks with five different models have been implemented. Two of these networks are generative and contain two models each. GAN is highly sensitive to the choice of hyperparameters, and much time has been used to explore different solutions for each network to avoid instability, oscillating parameters, week improvement, and over-performing discriminators.

Out of the 2000 synthetic segmentation masks generated by DCGAN, 652 were deemed unusable. The selection of usable and not usable was decided by comparing the generated samples with the PROMISE12 data. The unusable data could be realistic for a radiologist, especially the synthetic segmentation masks with one region of the prostate gland and a shape, different from the PROMISE12 data. The potential of a radiologist-based selection of usable synthetic segmentation masks could improve the quality of the generated MRIs.

Chapter 8

Conclusion and Future Directions

This thesis explores the potential of extending the dataset with synthetic MRIs and corresponding segmentation masks using GAN with the ambition to train a reliable DL-based segmentation system. The proposed method uses two GAN architectures abbreviated as DCGAN and Pix2Pix that introduce a new method to extend the dataset with new anonymized, uncorrelated, and realistic biomedical data. This method challenges the current one using standard image augmentation, which has proven to require precise analysis of the data and an expert to determine if the augmentation fits the desired dataset.

A set of experiments was conducted to find the optimal model architecture, data augmentation, and hyperparameters to generate the best possible data and segment the prostate gland correctly. Limited time and five different models to adjust represent the main challenge of this thesis. The data was supposed to be officially tested by PROMISE12, but time was a limitation. Instead, the validation set was used to test the performance, which may have affected the segmentation network's final result.

The proposed method using GAN to expand the dataset is sensitive to instability, but it presents results slightly better than just the original data. The quality of the generated MRIs can be improved, but they represent an image that has the potential to train a segmentation network. Close cooperation between a radiologist and the DL developer could improve both the generative and the segmentation network.

The final segmentation dice score for the generated and original data was on 0.731, compared to 0.693, using just the original data. This thesis's result indicates that GAN can extend the training set with new synthetic data and improve the segmentation robustness for prostate cancer diagnosis.

8.1 Future Directions

GAN has become a popular topic within DL and is still improving. Using the PROMISE12 dataset to train two GAN architectures has proven to be challenging as the dataset was small and contained data from cross-center and multiple vendors' equipment. Exploring the possibilities of using center-specific data and increasing the size of the dataset could potentially improve the quality of the generated data.

The augmentations used in this thesis to compare the proposed method with an existing one, uses the same parameters as a previous publication without any knowledge about the scientific conclusions behind the chosen augmentations. A future experiment could be to use a radiologist based opinion and follow their recommendations for augmentation techniques.

The amount of slices generated in this thesis work, is approximately the same as the number of slices in the original dataset. Further experiments where the generated data is extended with more examples or diminished to fewer could potentially improve the segmentation results.

Out of the 2000 synthetic segmentation masks generated by DCGAN, 652 were deemed unusable. A further experiment could be to test the potential of a radiologist-based selection of a usable synthetic segmentation mask in order to improve the quality of the generated MRIs.

Further development of the U-Net architecture could potentially improve the performance score adopting the same data used in this thesis. Further experiments applying different hyperparameters and model designs could improve the stability and the generative performance even more.

List of Figures

List of Tables

Appendix A

DCGAN and Pix2Pix model design

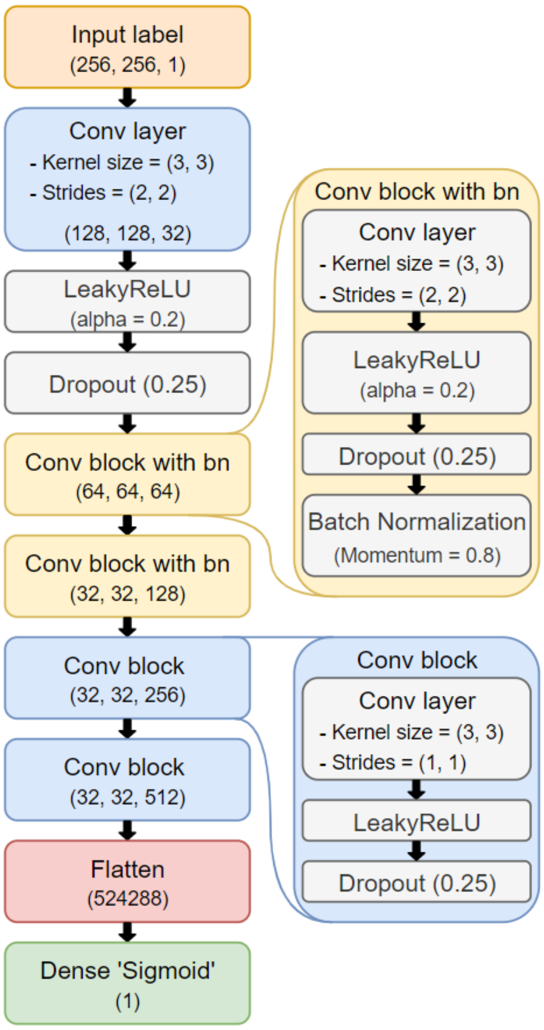


Figure A.1: DCGAN discriminator

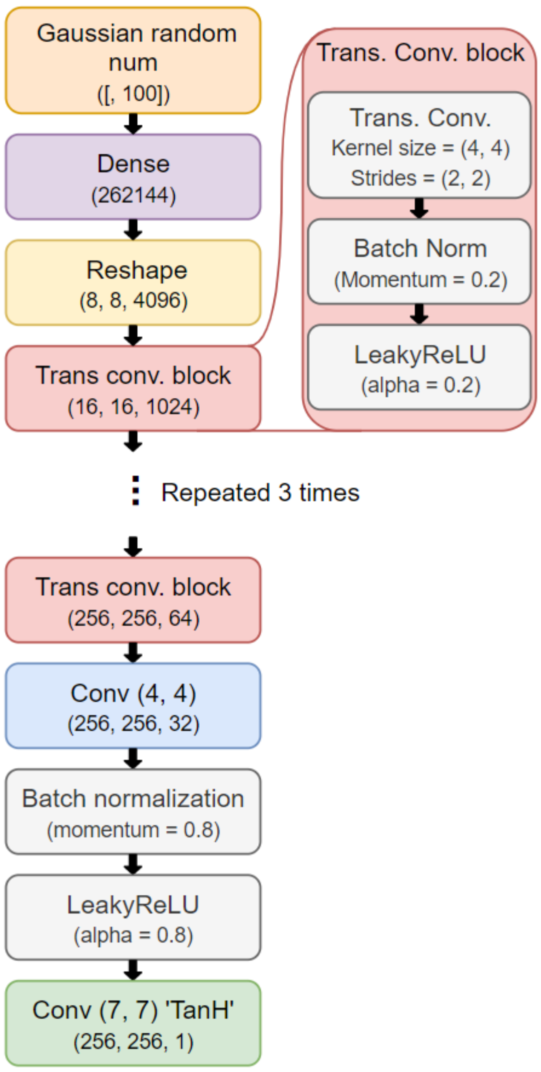


Figure A.2: DCGAN generator

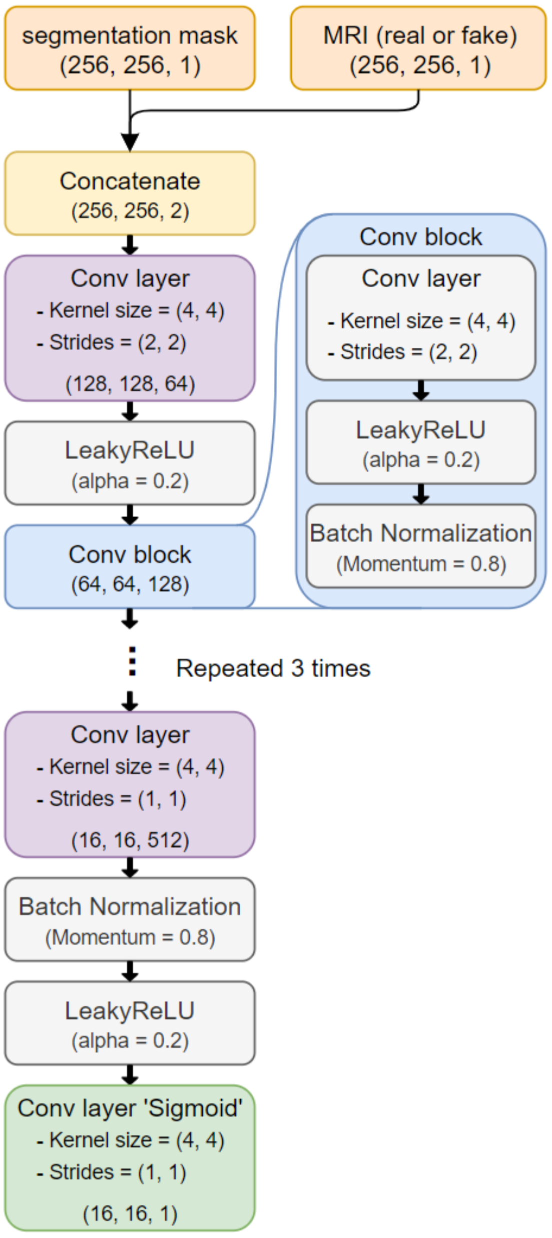


Figure A.3: Pix2Pix discriminator

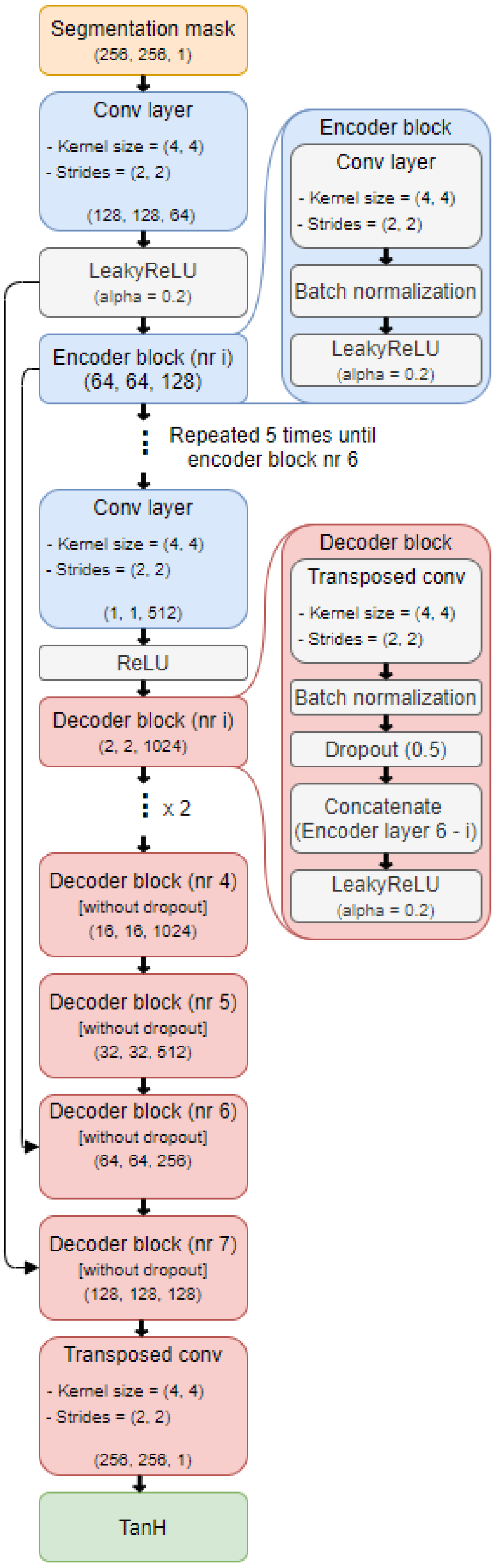


Figure A.4: Pix2Pix generator

Appendix B

U-Net Loss

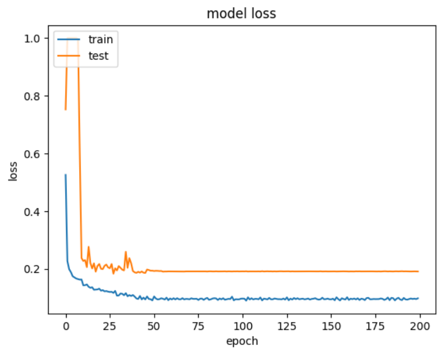


Figure B.1: Generated and original data. No augmentation.

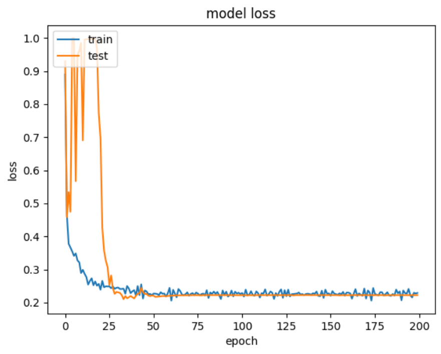


Figure B.2: Original data only. No augmentation.

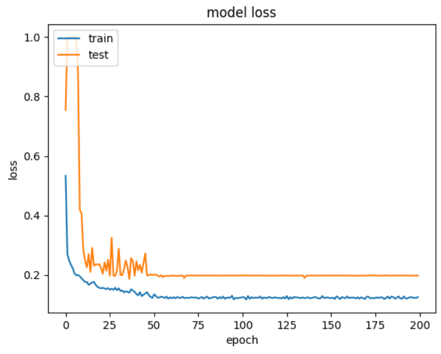


Figure B.3: Generated and original data. Flip augmentation.

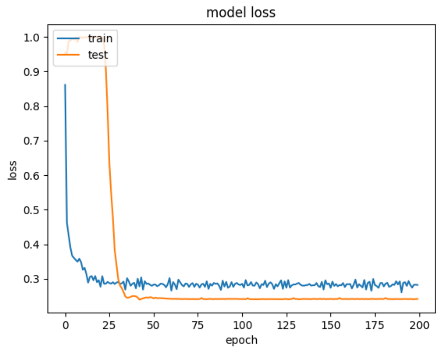


Figure B.4: Original data only. Flip augmentation.

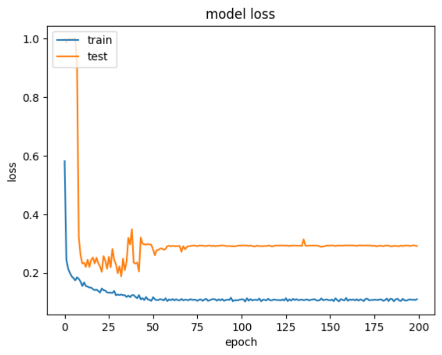


Figure B.5: Generated and original data. Rotation augmentation.

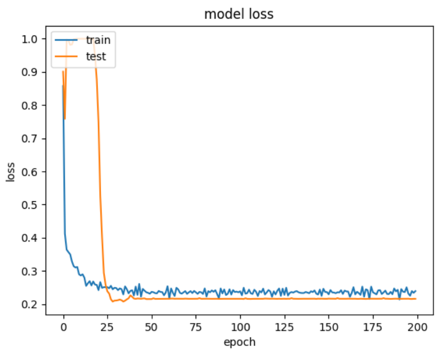


Figure B.6: Original data only. Rotation augmentation.

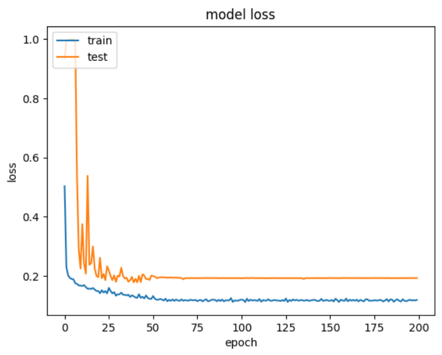


Figure B.7: Generated and original data. Shift augmentation.

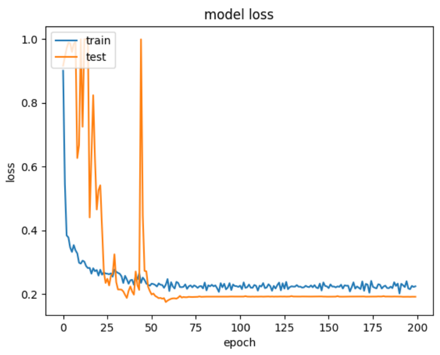


Figure B.8: Original data only. shift augmentation.

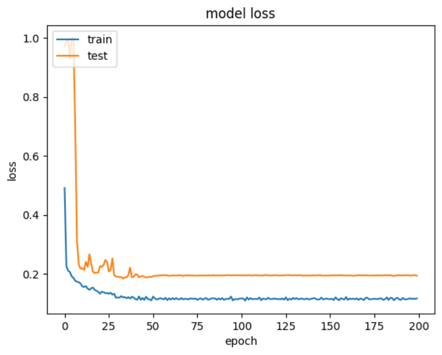


Figure B.9: Generated and original data. Zoom augmentation.

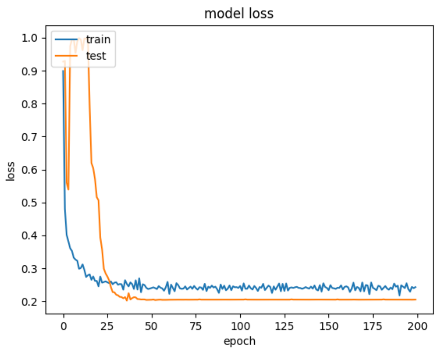


Figure B.10: Original data only. Zoom augmentation.

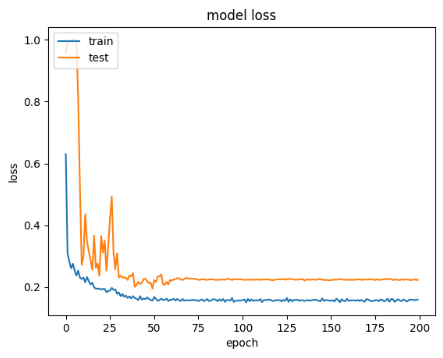


Figure B.11: Generated and original data. All augmentation.

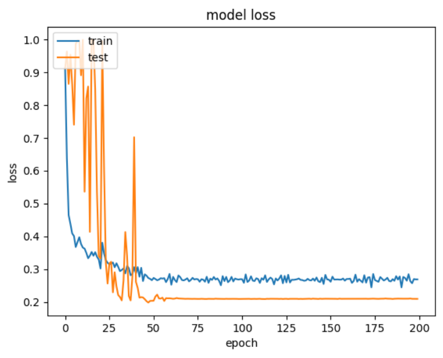


Figure B.12: Original data only. All augmentation.

Appendix C

Generated Segmentation Mask

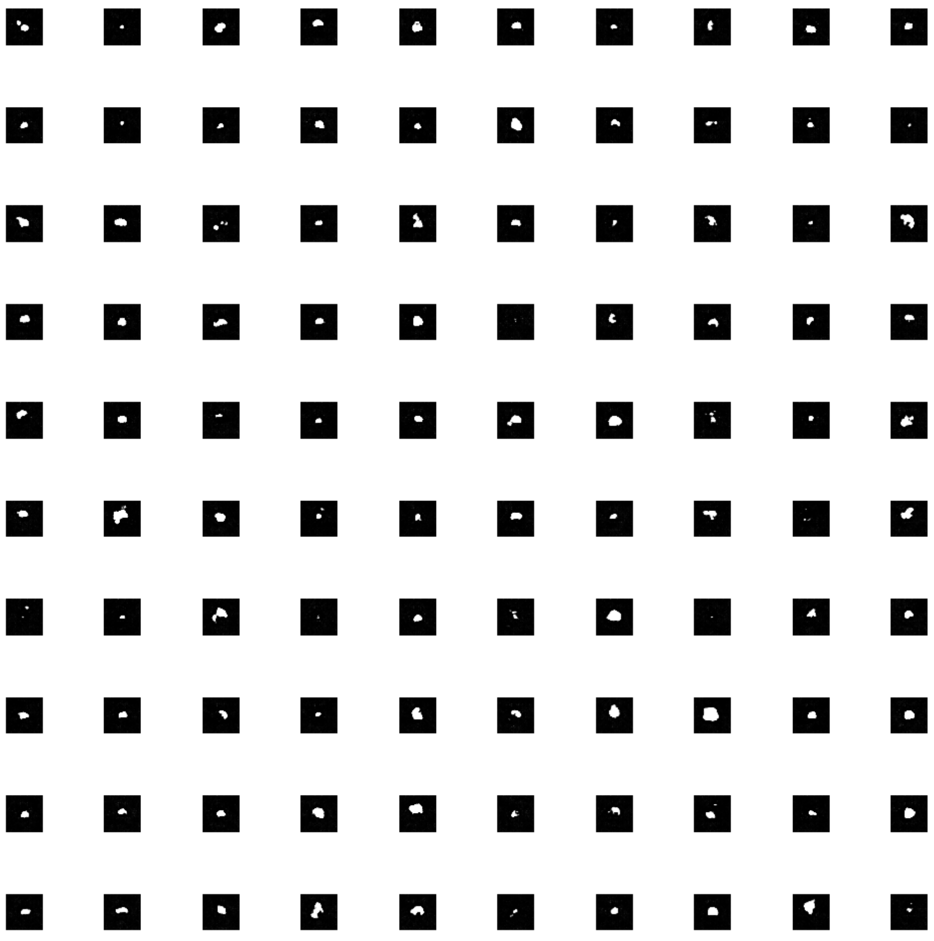


Figure C.1: Segmentation masks generated after **24 epochs**

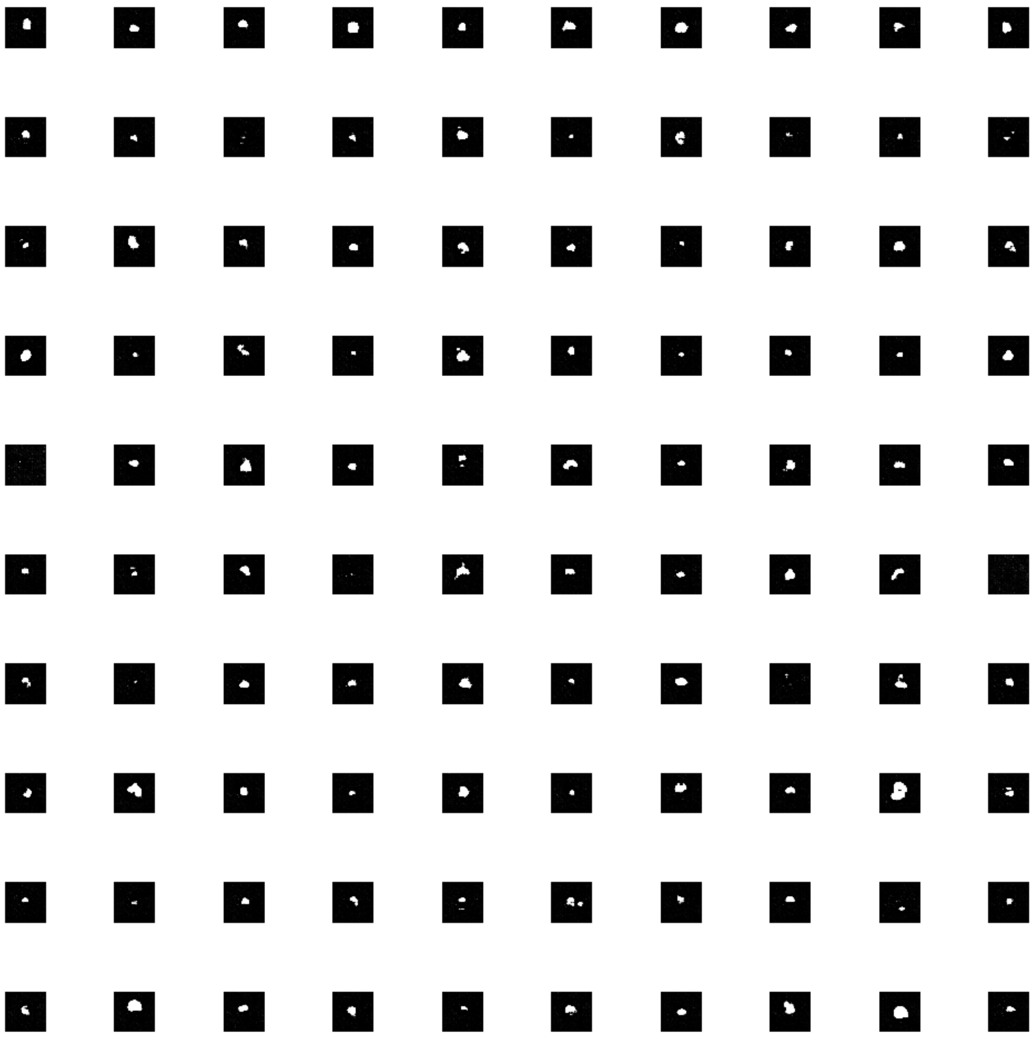


Figure C.2: Segmentation masks generated after **33 epochs**

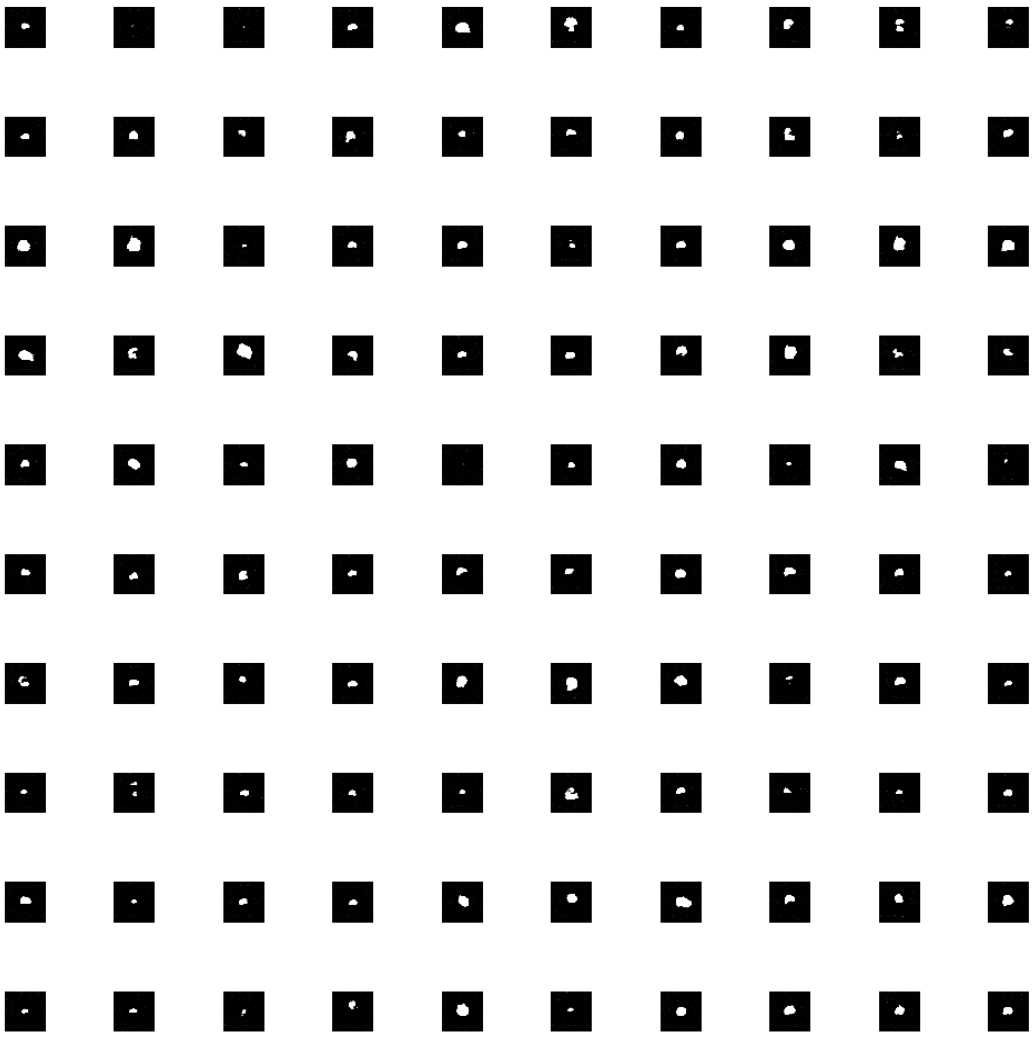


Figure C.3: Segmentation masks generated after **55 epochs**

Appendix D

Generated MRIs and Segmentation Mask

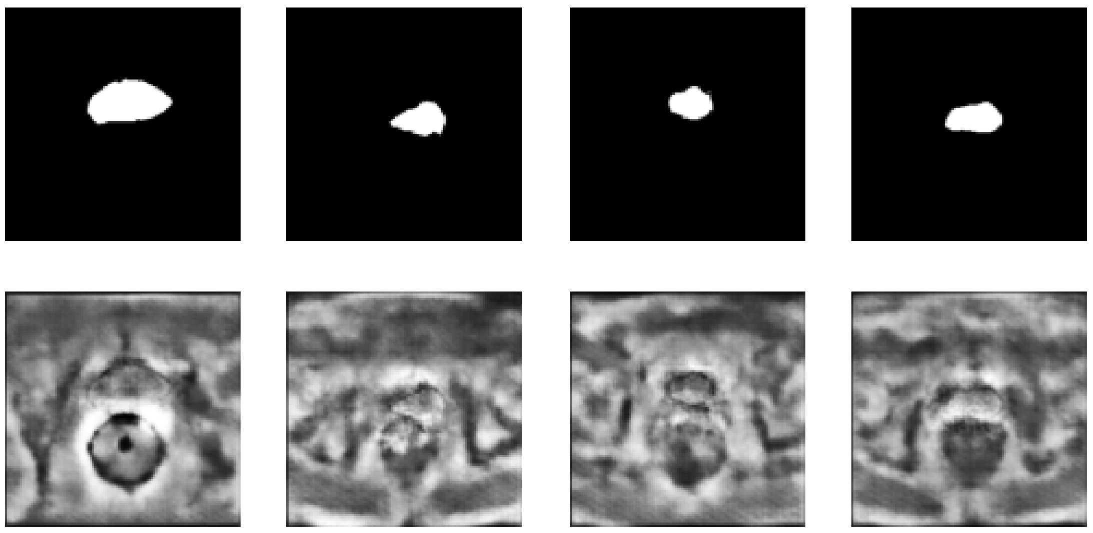


Figure D.1: MRIs generated at **10 epochs** and corresponding segmentation mask generated after **1500 epochs**.

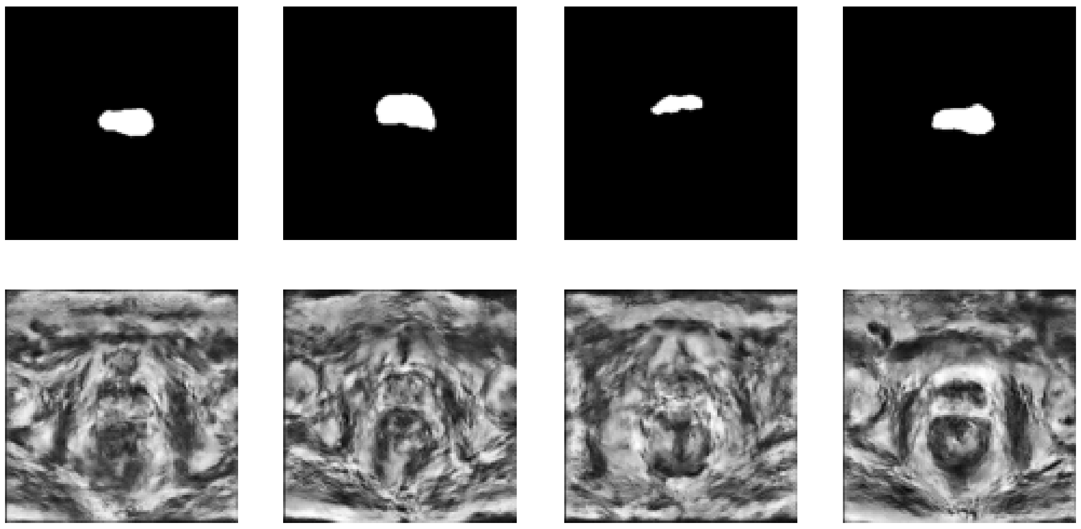


Figure D.2: MRIs generated at **50 epochs** and corresponding segmentation mask generated after **1500 epochs**.

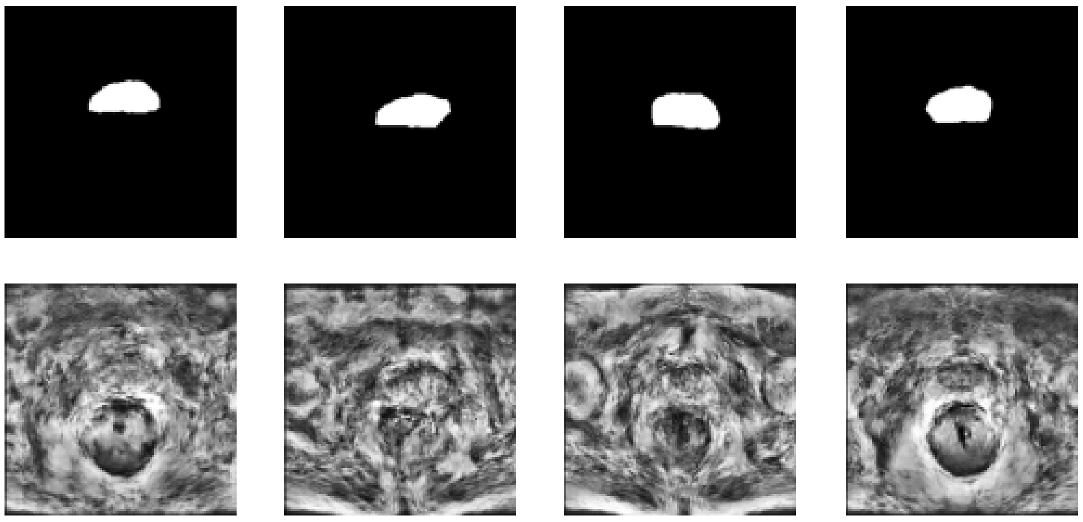


Figure D.3: MRIs generated at **200 epochs** and corresponding segmentation mask generated after **1500 epochs**.

Appendix E

Performance Score for MRIs at 50 Epochs

Val. data	**Mean DSC**	**std. DSC**	**Mean RAV**	**Mean HD**	**Mean MSD**
Org. and & gen. data	0.683	0.111	28.248	10.622	1.026
Vert. & horiz flip	0.672	0.065	53.348	19.588	5.679
Rot.	0.690	0.114	30.299	19.054	1.117
Shift	0.688	0.114	30.299	19.054	1.117
Zoom	0.699	0.007	99.264	15.860	36.488
All	0.654	0.048	64.452	9.327	15.779

Table E.1: Describes the evaluation score for real and generated data combined. The score is calculated for fake segmentation masks printed after **1500 epochs** and fake MRIs printed after **50 epochs**.

Appendix F

Segmentation Result

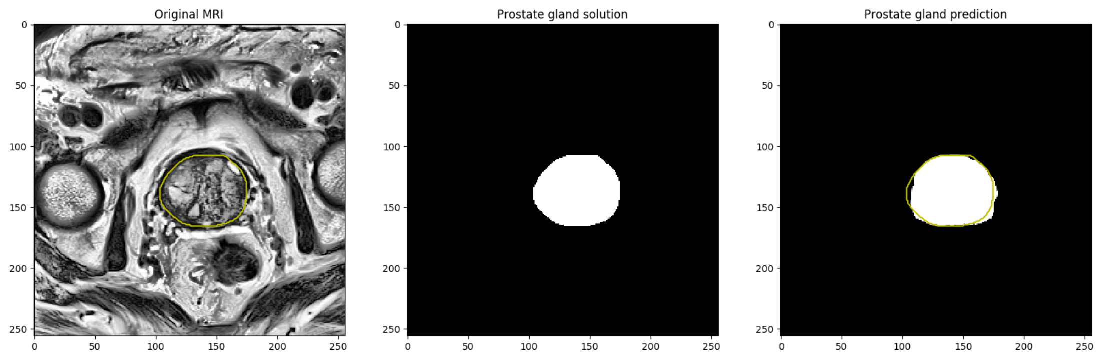


Figure F.1: The U-Net segmentation result on a random sample from the validation set after training on real and generated data combined, where fake segmentation masks are generated after 1500 epochs, and fake MRIs are generated after 162 epochs.

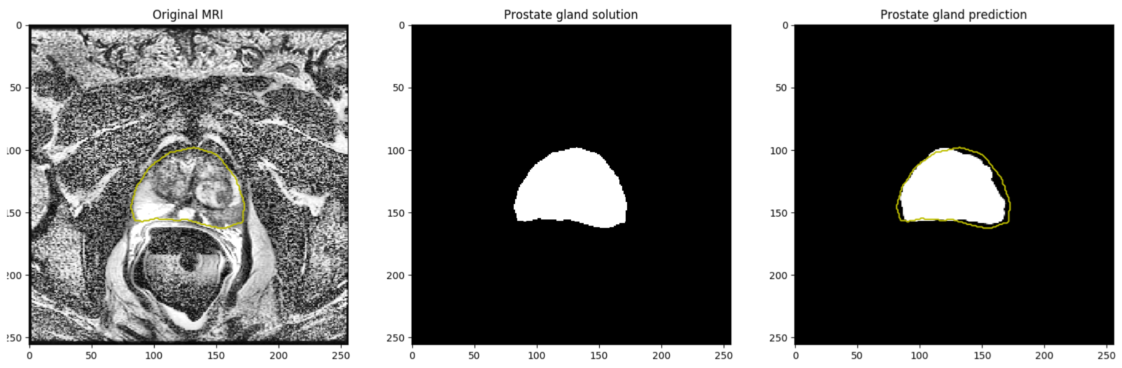


Figure F.2: The U-Net segmentation result on a random sample from the validation set after training on real and generated data combined, where fake segmentation masks are generated after 1500 epochs, and fake MRIs are generated after 162 epochs.

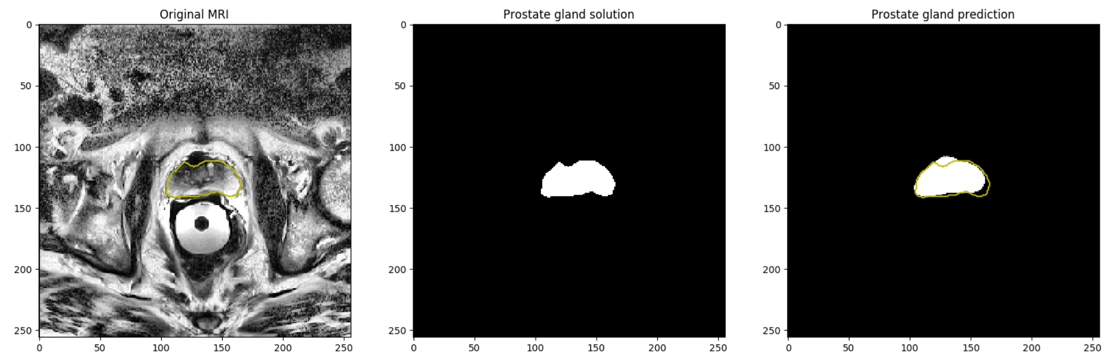


Figure F.3: The U-Net segmentation result on a random sample from the validation set after training on real and generated data combined, where fake segmentation masks are generated after 1500 epochs, and fake MRIs are generated after 162 epochs.

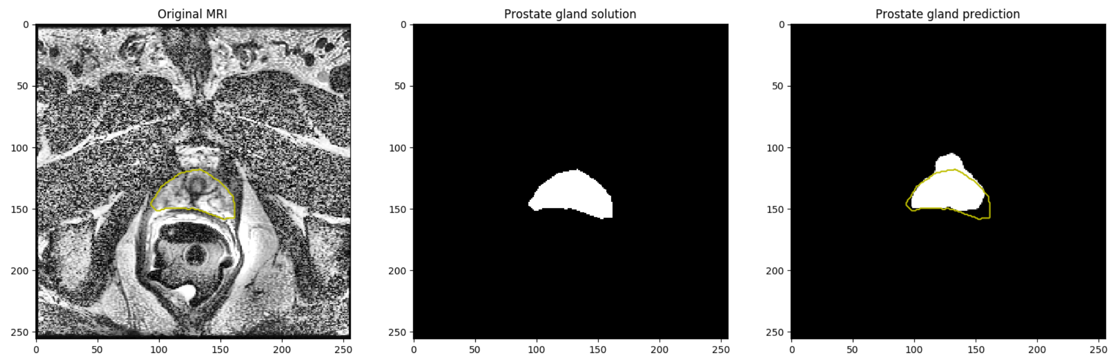


Figure F.4: The U-Net segmentation result on a random sample from the validation set after training on real and generated data combined, where fake segmentation masks are generated after 1500 epochs, and fake MRIs are generated after 162 epochs.

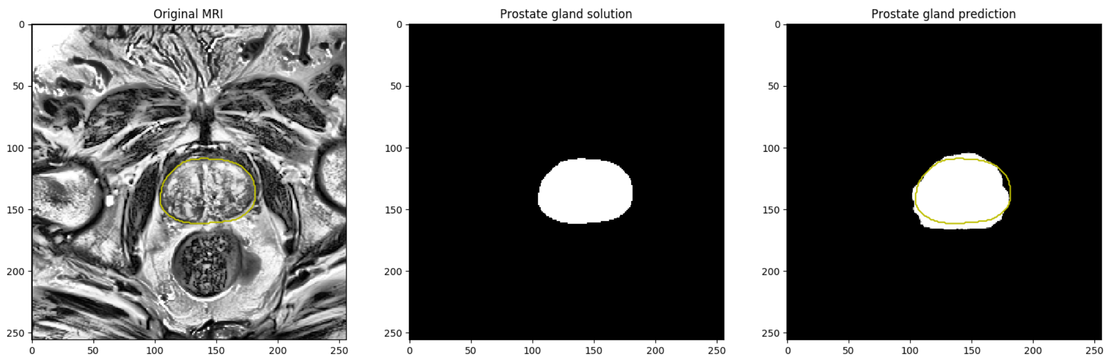


Figure F.5: The U-Net segmentation result on a random sample from the validation set after training on real and generated data combined, where fake segmentation masks are generated after 1500 epochs, and fake MRIs are generated after 162 epochs.

Appendix G

Python Code

This appendix explains the python scripts used in the implementation of this thesis. Below is a list of the packages required to execute the code.

G.1 Main Additional Packages

Package	**Version**
Keras-Applications	1.0.8
Keras-Preprocessing	1.1.0
matplotlib	3.0.3
numpy	1.16.4
opencv-python	4.2.0.32
pip	20.0.2
scikit-image	0.15.0
scipy	1.4.1
SimpleITK	1.2.4
tensorflow	1.14.0
tensorflow-gpu	1.14.0

G.2 data_pros.py

The file `data_pros.py` loads the PROMISE12 data and sperate the data used for training, validation, and test. All data is pre-processed and saved in separate NumPy arrays. This file is only executed when the PROMISE12 data is loaded for the first time and need to be pre-processed.

G.3 metrics_and_loss.py

This file includes all custom matrics and loss functions used in this thesis. This file is not executed individually but used by this thesis' python scripts.

G.4 dcgan.py

The file `dcgan.py` includes the DCGAN generator model, discriminator model, and the code to train these models. This script is executed to train the DCGAN models and save model weights that can be used to generate synthetic segmentation masks.

G.5 test_dcgan.py

This file loads the saved DCGAN model weights and uses the generator model in `dcgan.py` to generate new segmentation masks and save them in a separate NumPy array.

G.6 pix2pix.py

The `pix2pix.py` file contains the generator model, discriminator model, the code to train these models, and the code to generate synthetic MRIs based on the DCGAN generated segmentation masks. These MRIs are generated during training and saved in a separate NumPy array. This file does also saves model weights that can be used to generate new MRIs separately. The `pix2pix.py` file is executed to train the pix2pix models and to generate synthetic MRIs.

G.7 unet.py

The file `unet.py` contains the U-Net model and the code to train this model using the original data, the augmented data, and the generated data. This file is executed to train the segmentation model and save model weights that can be used to test the performance on a separate dataset.

G.8 test_unet.py

The `unet_test.py` file includes the code to test the segmentation performance using the saved model weights and the model in `unet.py`. This code is executed to test the segmentation performance.

Bibliography

[1] F. Lam et al. J. Ferlay, M. Ervik. Global cancer observatory: Cancer today. international agency for research on cancer, lyon 2018. `https://gco.iarc.fr/today`, 2019. [Online; accessed 15-May-2020].

[2] MaryBeth B Culp, Isabelle Soerjomataram, Jason A Efstathiou, Freddie Bray, and Ahmedin Jemal. Recent global patterns in prostate cancer incidence and mortality rates. *European urology*, 77(1):38–52, 2020.

[3] Bristol-Myers Squibb Oslo Economics. kostnader for pasientene, helsetjenesten og samfunnet. In *Kreft i Norge*, pages 125–134, 2016. URL `https://occincubator.com/ny-rapport-kreftkostnader-i-norge/`.

[4] Veeru Kasivisvanathan, Antti S Rannikko, Marcelo Borghi, Valeria Panebianco, Lance A Mynderse, Markku H Vaarala, Alberto Briganti, Lars Budäus, Giles Hellawell, Richard G Hindley, et al. Mri-targeted or standard biopsy for prostate-cancer diagnosis. *New England Journal of Medicine*, 378(19):1767–1777, 2018.

[5] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938, 2018.

[6] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[7] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[8] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[10] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[11] Hoo-Chang Shin, Neil A Tenenholtz, Jameson K Rogers, Christopher G Schwarz, Matthew L Senjem, Jeffrey L Gunter, Katherine P Andriole, and Mark Michalski. Medical image synthesis for data augmentation and anonymization using generative adversarial networks. In *International workshop on simulation and synthesis in medical imaging*, pages 1–11. Springer, 2018.

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[13] Geert Litjens, Robert Toth, Wendy van de Ven, Caroline Hoeks, Sjoerd Kerkstra, Bram van Ginneken, Graham Vincent, Gwenael Guillard, Neil Birbeck, Jindang Zhang, et al. PROMISE12 Details. `https://promise12.grand-challenge.org/Details/`, . [Online; accessed 1-February-2020].

[14] Inom Mirzaev. Fully convolutional neural network with residual connections for automatic segmentation of prostate structures from mr images, 2017. URL `https://github.com/mirzaevinom/promise12_segmentation`. [Online; accessed 5-June-2020].

[15] Norsk Helseinformatikk. Prostatakreft. `https://nhi.no/sykdommer/kreft/mannlige-kjonnsorganer-kreft/prostatakreft/`, May 2019. [Online; accessed 1-May-2020].

[16] Cancer.Net Editorial Board. Digital rectal exam (dre), 2019. URL `https://www.cancer.net/navigating-cancer-care/diagnosing-cancer/tests-and-procedures/digital-rectal-exam-dre`. [Online; accessed 25-June-2020].

[17] Michael A Liss, Behfar Ehdaie, Stacy Loeb, Maxwell V Meng, Jay D Raman, Vanessa Spears, and Sean P Stroup. An update of the american urological association white paper on the prevention and treatment of the more common complications related to prostate biopsy. *The Journal of urology*, 198(2):329–334, 2017.

[18] Wikipedia contributors. Gleason grading system, 2020. URL `https://en.wikipedia.org/wiki/Gleason_grading_system`. [Online; accessed 21-June-2020].

[19] Cancer research UK. Transrectal ultrasound scan (trus). `https://www.cancerresearchuk.org/about-cancer/prostate-cancer/getting-diagnosed/`

tests/transrectal-ultrasound-scan-trus, May 2019. [Online; accessed 15-May-2020].

[20] Turkbey B Siddiqui MM, Rais-Bahrami S. et al. comparison of mr/ultrasound fusion–guided biopsy with ultrasound-guided biopsy for the diagnosis of prostate cancer. *JAMA*, page 390–397., 2015. URL https://jamanetwork.com/journals/jama/fullarticle/2091987.

[21] Jeffrey C Weinreb, Jelle O Barentsz, Peter L Choyke, Francois Cornud, Masoom A Haider, Katarzyna J Macura, Daniel Margolis, Mitchell D Schnall, Faina Shtern, Clare M Tempany, et al. Pi-rads prostate imaging–reporting and data system: 2015, version 2. *European urology*, 69(1):16–40, 2016.

[22] Donald W McRobbie, Elizabeth A Moore, Martin J Graves, and Martin R Prince. *MRI from Picture to Proton*. Cambridge university press, 2017.

[23] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[24] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[25] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

[26] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[27] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[28] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[29] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8), 2012.

[30] Wikipedia contributors. Stochastic gradient descent, 2020. URL https://en.wikipedia.org/wiki/Stochastic_gradient_descent#RMSProp. [Online; accessed 27-May-2020].

[31] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[32] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[33] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[34] Isha Salian. SuperVize Me: What's the Difference Between Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning? `https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/`, August 2018. [Online; accessed 1-June-2020].

[35] Jason Brownlee. *Generative Adversarial Networks with Python: Deep Learning Generative Models for Image Synthesis and Image Translation.* Machine Learning Mastery, 2019.

[36] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[37] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[38] Aqeel Anwar. What is transposed convolutional layer? https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11, March 2020. [Online; accessed 30-May-2020].

[39] Jiwon Jeong. The most intuitive and easiest guide for convolutional neural network. https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480, January 2019. [Online; accessed 30-May-2020].

[40] François Chollet et al. Keras. `https://keras.io`, 2015. [Online; accessed 18-May-2020].

[41] François Chollet et al. Dense layer. `https://keras.io/api/layers/core_layers/dense/`, 2020. [Online; accessed 25-June-2020].

[42] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[43] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[44] François Chollet et al. Image data preprocessing. `https://keras.io/api/preprocessing/image/`, 2015. [Online; accessed 11-June-2020].

[45] Jason Brownlee. How to configure image data augmentation in keras. https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/, 2019. [Online; accessed 25-June-2020].

[46] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[47] Lars Nieradzik. Losses for image segmentation. `https://lars76.github.io/neural-networks/object-detection/losses-for-segmentation/`, 2018. [Online; accessed 11-June-2020].

[48] Rodney LaLonde. Capsules for object segmentation (segcaps), file: metrics.py. `https://github.com/lalonderodney/SegCaps/blob/master/metrics.py`, 2020. [Online; accessed 11-June-2020].

[49] mlnotebook. Surface Distance Function. `https://mlnotebook.github.io/post/surface-distance-function/`. [Online; accessed 13-June-2020].

[50] Wikipedia contributors. Hausdorff distance, 2020. URL `https://en.wikipedia.org/wiki/Hausdorff_distance`. [Online; accessed 13-June-2020].

[51] Guido Van Rossum et al. Python programming language. In *USENIX annual technical conference*, volume 41, page 36, 2007.

[52] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org. [Online; accessed 20-May-2020].

[53] Terry S Yoo, Michael J Ackerman, William E Lorensen, Will Schroeder, Vikram Chalana, Stephen Aylward, Dimitris Metaxas, and Ross Whitaker. Engineering and algorithm design for an image processing api: a technical report on itk-the insight toolkit. *Studies in health technology and informatics*, pages 586–592, 2002.

[54] Matthew Michael McCormick, Xiaoxiao Liu, Luis Ibanez, Julien Jomier, and Charles Marion. Itk: enabling reproducible research and open science. *Frontiers in neuroinformatics*, 8:13, 2014.

[55] Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[56] Geert Litjens, Robert Toth, Wendy van de Ven, Caroline Hoeks, Sjoerd Kerkstra, Bram van Ginneken, Graham Vincent, Gwenael Guillard, Neil Birbeck, Jindang Zhang, et al. Evaluation of prostate segmentation algorithms for mri: the promise12 challenge. *Medical image analysis*, 18(2):359–373, 2014.

[57] Geert Litjens, Robert Toth, Wendy van de Ven, Caroline Hoeks, Sjoerd Kerkstra, Bram van Ginneken, Graham Vincent, Gwenael Guillard, Neil Birbeck, Jindang Zhang, et al. PROMISE12 dataset. `https://promise12.grand-challenge.org/Download/`, . [Online; accessed 1-February-2020].

[58] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

[59] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. Module: equalize_adapthist, 2011. URL `https://scikit-image.org/docs/dev/api/skimage.exposure.html#skimage.exposure.equalize_adapthist`. [Online; accessed 5-June-2020].

[60] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[61] Wikipedia contributors. X-ray, 2020. URL `https://en.wikipedia.org/wiki/X-ray`. [Online; accessed 12-June-2020].

[62] Thomas Neff. Data augmentation in deep learning using generative adversarial networks. pages 1–113, 2018.

[63] Soumith Chintala and David Lopez-Paz. Nips 2016 workshop on adversarial training - soumith chintala - how to train a gan. URL `https://www.youtube.com/watch?v=X1mUN6dD8uE&t=1397s`. [Online; accessed 8-June-2020].

[64] Geert Litjens, Robert Toth, Wendy van de Ven, Caroline Hoeks, Sjoerd Kerkstra, Bram van Ginneken, Graham Vincent, Gwenael Guillard, Neil Birbeck, Jindang

Zhang, et al. PROMISE12 Results. `https://promise12.grand-challenge.org/evaluation/results/`, . [Online; accessed 13-June-2020].

[65] Haozhe Jia, Yang Song, Heng Huang, Weidong Cai, and Yong Xia. Hd-net: Hybrid discriminative network for prostate segmentation in mr images. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 110–118. Springer, 2019.

[66] Fabian Isensee, Paul F Jäger, Simon AA Kohl, Jens Petersen, and Klaus H Maier-Hein. Automated design of deep learning methods for biomedical image segmentation. *arXiv preprint arXiv:1904.08128*, 2019.