University of Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme/specialisation: Master of Science in Computational Engineering	Spring semester, 2021 Open/Confidential: Open
Author: Omer Parvez	
Programme coordinator: Aksel Hiorth Supervisor(s): Ketil Oppedal and Álvaro Fernández Quílez	
Title of master's thesis: Conditional data generation for an improved diagnosis in prostate cancer	
ECTs: 30	
Keywords: Generative Adversarial Networks, DCGAN, cGAN, Binary classification, VGG16, Prostate,	Number of pages: 51 + supplemental material/other: 29 Stavanger, July 15, 2021

Abstract

Prostate Cancer is the second most common cancer in men worldwide, the fourth most commonly occurring cancer overall and the sixth leading cause of cancer death among men worldwide. Early detection of prostate cancer is crucial for survival. MRI examination is an essential and a comfortable tool towards a precise diagnosis at an early stage. But this diagnosis is dependent on the experience and expertise of the reader. Deep learning methods can be used for the classification of tumors, as various DL methods have proven to be helpful for the classification and detection tasks.

To work towards this goal, this thesis will explore how generative adversarial networks can be used to improve prostate MRI classification. Different deep learning architecture have been proven accurate in the classification of the biomedical images. However, large volume of training data is required, which is difficult to obtain due to the patient privacy policy. To this end, this thesis proposes a two step approach to improve the classification, the first step is to generate anonymized training data using using two GAN architectures and the second step is to use the anonymized training data to train classification network. This thesis proposes two GAN architectures, DCGAN and cGAN to generate training data to improve the image classification. The synthetic data is used together with original data for the training of VGG16 classification network and compare the performance of the classification network with generated data and without it.

The final results indicate significantly better image classification using cGAN when compared with the original data and classic augmentation methods. The network performs better for the generated data when compared with the original and augmented data.

Acknowledgments

This thesis marks the end of my Master of Science degree in Computational Engineering at University of Stavanger, Department of Energy Resources (IER). The thesis was conducted during the spring semester of 2021, and the journey was educational and exciting due to support of my supervisors.

First of all, I am grateful to the University of Stavanger for giving me the opportunity to work with the state-of-the-art hardware. I would like to thank my head supervisor Ketil Oppedal and co-supervisor Álvaro Fernández Quílez for supporting me and guiding me during the thesis work. I would also take this opportunity to thank the University staff, Theodor Ivesdal and Rune Wetteland for providing the support when I most needed it.

Lastly, I would like to thank my wonderful friend Habib-Ullah for always supporting and encouraging me for the past two years without whom this wonderful journey would not have been possible.

Contents

List of Figures

List of Tables

Abbreviations:

2D	Two - Dimensional
3D	Three - **D**imensional
MRI	Magnetic Resonance Imaging
T2W	T2 - Weighted **I**mage
ADC	Apparent Diffusion Coefficient
DWI	Diffusion Weighted **I**maging
DL	Deep Learning
NN	Neural Networks
CNN	Convolutional Neural Networks
GAN	Generative Adversarial Network
DCGAN	Deep convolutional Generative adversarial Network
cGAN	Conditional Generative adversarial Network
DRE	Digital Rectum Examination
TRUS	Transrectal Ultrasound Scan
NumPy	Numerical Python Python
LR	Learning Rate
BS	Batch Size
LD	Latent Dimension
MSE	Mean Squared Error
SSIM	Structural Similarity Index Measurement
KL	Kullback – Leibler
ANN	Artificial Neural Networks
API	Application Programming Interface
DICOM	**D**igital **I**maging and **C**ommunication in **M**edicine
ROC	**R**eceiver **O**perating **C**haracteristics **Curve**
AUC	**A**rea under the **C**urve
UiS	**U**niversity of **S**tavanger
ML	**M**achine **L**earning
ReLU	**R**ectified **L**inear **A**ctivation **F**unction
GPU	**G**raphics **P**rocessing **U**nit
IDE	**I**ntegrated **D**evelopment **E**nviorment

Chapter 1

Introduction

1.1 Motivation

Prostate Cancer is the second most common cancer in men worldwide, the fourth most commonly occurring cancer overall and the sixth leading cause of cancer death among men. There were 1,414,259 new estimated cases and 375,304 estimated deaths in 2020[10]. This number is estimated to increase to approximately 2.3 million new cases by 2040 due to population growth and increased expected life.[14]. Most medical organizations encourage men in their 50s to discuss the pros and cons of prostate cancer screening with their doctors. There are various screening tests for the prostate performed by the General Physician in a clinic, like Digital rectal exam (DRE) and Prostate-specific antigen (PSA) test [41]. If prostate cancer screening detects an abnormality, the doctor may recommend further tests to determine whether the cancer is present such as ultrasound, magnetic resonance imaging (MRI) and prostate biopsy. However, several patients complain of getting an infection after a guided biopsy at the hospital.MRI has proven to be successful for the detection and diagnosis of prostate cancer[41]. Increased MRI has made the examination more comfortable and more efficient, resulting in an improved prostate gland examination and detection of malignant tumours in the gland. However, MRI results are reader-dependent, leading to variability in the outcome depending on the person's expertise in the examination. The field of medicine has relied on experts who gain knowledge through experience and self-learning, which is necessary for a dynamic healthcare environment. The increase in knowledge and understanding of diseases is closely linked with an increase in1 data and information due to advanced tools that generate quantitative and qualitative measurements of different parameters. Such a big data field is ready for the application of machine learning tools. There is a growing development in the application of ML in the medical industry, which can collect information from numerous sources and aid the decision-making process of highly skilled workers. Machine learning is being used in the areas in the healthcare industry, from diagnosis and prognosis to drug development and epidemiology, with significant potential to transform the medical landscape[4].In recent years, various types of medical image processing and recognition have adapted machine learning methods, including MRI images, ultrasound images, pathological images, and much more. At present, deep learning methods are used in classification and segmentation in medical images.[4] A reliable tool could improve the existing examination time by freeing up the time of the medical experts while maintaining the quality of the diagnosis and allow more time for other patients. Training a Neural network or machine learning(ML) algorithm requires extensive data to give precise and accurate results. Gaining access to a large amount of medical image data is often difficult to obtain due to patient privacy. There are existing approaches to extend the data-set called data augmentation, which helps train the ML sometimes, but the correlation between the augmented data and original data is very high. Other methods to generate data like Generative adversarial networks can be used to increase the network efficiency by generating data that is not correlated and labelled, saving expert hours for more productive work.

1.2 Problem Definition

This thesis's primary goal is to improve the convolutional neural network classification (CNN) in binary classification performed on apparent diffusion coefficient (ADC) MRI images. The main challenge of using medical data is the privacy policy, limiting access to a large amount of labelled data, thus training the neural network more challenging. To extend the data-set, GAN will be used to generate MRI images containing prostate cancer tumours which will be used to train the CNN for image classification performed on the MRIs of the prostate.

1.2.1 Objectives

- To generate new anonymized labelled training data using two Generative Adversarial Network (GAN) architectures, named Deep Convolutional Generative Adversarial Networks (DCGAN) and Conditional Generative Adversarial Network (cGAN).
- Use the real and generated MRI images to train a CNN and compare the results obtained from the standard image and the images generated using GAN.

1.2.2 Proposed Method Overview

The data containing ADC MRI images are loaded. Then this data is pre-processed and split into two categories based on the tumour type in patients. Afterwards, this data is used to train DCGAN and cGAN to generate more data in a supervised manner. After training the GAN, data is generated from trained models and is then further used to train the binary classifier to compare training results with generated and original data. Figure 1.1 shows overview for the thesis.

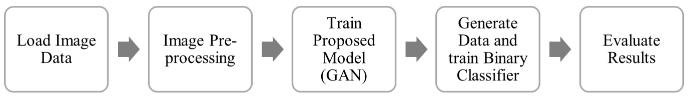

Load Image Data
Image Pre-processing
Train Proposed Model (GAN)
Generate Data and train Binary Classifier
Evaluate Results

Figure 1.1: The figure illustrates overview of this thesis methodology

1.3 Related Work

This thesis uses three different neural networks to generate images and test the results. The neural networks used to generate images are *Deep Convolutional Generative Adversarial Networks* (DCGAN) and *Conditional Generative Adversarial Networks.* DCGAN was first introduced by Alec Radford et al. in the paper *"Unsupervised representation learning with deep convolutional generative adversarial networks"*[33]. The goal of the author was to combine the existing Convolutional Neural Network and "Generative Adversarial Networks" with unsupervised learning with unsupervised learning.

The second architecture utilized to generate images is called Conditional Generative Adversarial Networks (cGAN) was first proposed by Mehdi Mirza and Simon Osindero in their 2014 paper titled *"Conditional Generative Adversarial Nets."*[27]. In this paper the authors tried to direct the image generation process of the generator model by providing the model additional information. This network takes labels as an additional information and generates directed images.

The paper *"Synthetic data augmentation using GAN for improved liver lesion classification"*[[19]] apply GAN to increase the liver lesion data and then use the augmented data to improve the classification results.

The VGG16 network is used as a classifier in this thesis and it was first proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" [37]. The model was proposed to increase the depth of using very small convolution filters to show the improvement on the prior networks.

1.4 Outline

This chapter gives introduction and explains the motivation for the subject. The remaining part of this thesis is structures in to seven different chapters.

- The second chapter is Medical Background and it describes the essential medical theory used in this thesis.
- The third chapter is Technical Background and contains the background theory related to the technology used in this thesis.
- The fourth chapter is Data-set and Image Pre-Processing which describes the data-set used in this thesis and pre-processing steps used to organize and store the data.
- The fifth chapter is Solution approach and describes the proposed method to generate new data using GAN.
- Chapter six is Experimental Evaluation and Results, in which the results are discussed and compares various augmentation methods used.
- Chapter seven presents a decision of the results and limitations of this thesis.
- Chapter eight presents the conclusion for this thesis and future recommendations.

.

Chapter 2

Medical Background

This chapter will provide an overview of the prostate gland, various examination techniques for diagnosis and detection of prostate cancer and treatment.

2.1 Prostate Cancer

Prostate Cancer is the second most occurring cancer in men worldwide, the fourth most commonly occurring cancer overall and the sixth leading cause of cancer death among men. There were 1,414,259 new estimated cases and 375,304 estimated deaths in 2020.[10]. This number is estimated to increase to approximately 2.3 million new cases by 2040 due to population growth and increased expected life.[15]

Cells are the basic units that make up the human body. Cells grow and divide to make new cells as the body needs them. Cells die when they get old or damaged. Then, new cells take their place. Cancer starts when cells in the body begin to grow out of control. Cells in nearly any part of the body can become cancer cells and then spread to other areas of the body. This rapid activity of the cells may cause a mass of cells called a tumour. These tumours can be benign and malignant. A cancerous tumour is malignant, which means it can grow and spread to other parts of the body. A benign tumour means that it can grow but will not spread[40].

The prostate is a gland found only in males. It makes some of the fluid that is part of semen. The prostate is below the bladder and in front of the rectum. Prostate cancer begins when cells in the prostate gland start growing uncontrollably. Behind the prostate are glands called seminal vesicles that make most of the fluid for semen. The urethra, a tube that carries urine and semen out of the body through the penis, goes through the centre of the prostate[22].

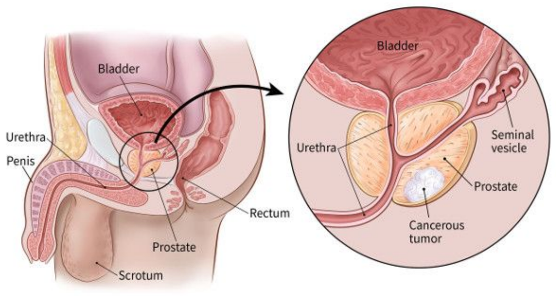


Figure 2.1: Prostate Gland with cancer cells[22]

The size of the prostate changes with a man's age, it is the size of a walnut at a young age, but it can grow bigger in older men.

2.2 Prostate Cancer Examination Methods

There is no definitive test for prostate cancer. All the tests used to help diagnose the condition have some pros and cons; hence they are used in combination. The following screening tests are usually involved in the complete diagnosis process if the patient experiences one or more symptoms.

2.2.1 Digital Rectum Examination

Digital Rectum Exam (DRE) involves inserting a gloved, lubricated finger into the rectum which allows a GP to physically feel the prostate's back. The physical examination help the GP to determine the size and shape of the prostate. DRE is usually carried out by a GP before referring the patient to a specialist. Figure 2.2 shows the demonstration of DRE.

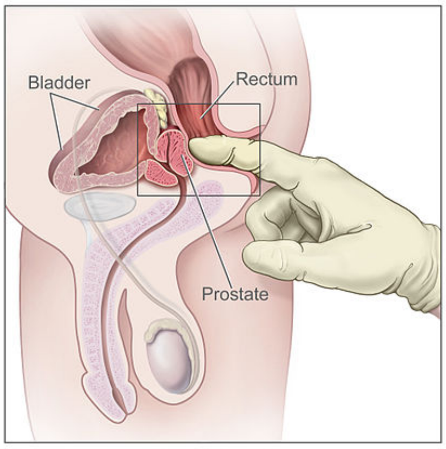


Figure 2.2: Digital Rectum Examination[34].

2.2.2 Prostate-Antigen Specific Test

Prostate-Antigen Specific (PSA) test involves taking a blood sample to determine the amount of Prostate-Specific Antigen in the patient's blood. When a patient has prostate cancer, the quantity of PSA in his blood rises, however it can also rise when the patient has a benign prostate enlargement or an infection in the prostate. This test is most beneficial when a GP can compare the results of test, before and after the patient got cancer or infection[32].

2.2.3 Biopsy

Biopsies will be performed if the doctor suspects cancer. A transrectal ultrasound scan (TRUS) is the most frequent method of prostate biopsy. Even with preventative measures, however, the risk of infection is estimated to be between 5% and 7%. This test is done in a hospital by professionals, and it involves inserting a needle into the prostate eight to ten times to collect tissue from various areas.[31] The TRUS procedure is depicted in the figure 2.3. The Gleason score grading method is used to determine the patient's prognosis. The Gleason score is used by pathologists to determine the stage of prostate cancer. A high Gleason score suggests a malignancy that is aggressive and has a poor prognosis.

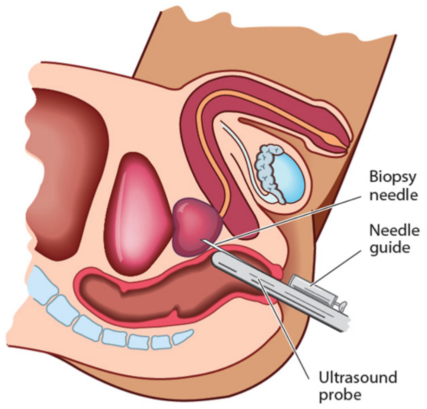


Figure 2.3: Trans-rectal ultrasound scan examination[9]

2.2.4 Magnetic Resonance Imaging

MRI scans are utilized to determine the exact position of a lesion and are generally done prior to a biopsy. According to some studies, MRI can also assist determine the type of the lesion, such as whether it is benign or cancerous. A greater understanding between medical professionals and radiologists can help with the examination and diagnosis of prostate cancer.

A radiologist uses the Prostate Imaging Reporting and Data System (PI-RADS) grading system to analyze MRIs. PI-RADS was created to improve worldwide standardization of multi-parametric magnetic resonance imaging (mpMRI) examinations of the prostate. Standardization seeks to reduce needless biopsies by improving the identification of clinically significant malignancy and locating benign illnesses[31].

Chapter 3

Technical Background

This chapter will present the technical background for the terminologies and concepts used in this thesis.

3.1 Magnetic Resonance Imaging (MRI)

This section will explain some terminologies in the topic of MRI technology and the process involved in it.

3.1.1 Basic Terminology

MRI is a non-invasive imaging technology that produces detailed three-dimensional images. It is used for disease detection, diagnosis, and treatment monitoring. It is based on technology that excites and detects the change in the direction of the rotational axis of protons found in the water that makes up living tissues[25].

MRI uses the properties of hydrogen in water or lipids to capture images. Two of the most fundamental parameters are repetition time (TE) and time to echo (TR). TR is the amount of time between successive pulse sequences applied to the same slice. TE is the time between the delivery of the RF pulse and the receipt of the echo signal[29].

Tissues can be characterized by two different relaxation times that are T1 and T2. T1, which represents longitudinal relaxation time, is the time constant that determines the rate at which excited protons return to the equilibrium. It is a measure of the time taken for spinning protons to realign with the external magnetic field. T2, which represents transverse relaxation time, is the time constant that determines at which excited protons reach equilibrium or go out of phase with each other. It is a measure of the time taken for spinning protons to lose phase coherence among the nuclei spinning perpendicular to the main field[29].

The most common MRI sequences are T1-weighted and T2-weighted scans. T1-weighted images are produced by shorter TE and TR times, while longer TE and TR times produce T2-weighted images.

Diffusion-weighted imaging (DWI) is designed to detect the random movements of water protons to construct patterns in MRI images. A combination of images with different diffusion weighting amounts provides an apparent diffusion coefficient (ADC) map or ADC image. In ADC imaging, protons exhibit free mobility in the tumor region than the surrounding, and therefore, the corresponding area of higher diffusivity is represented as a brighter region, indicating a high ADC value in the obtained ADC map[11].

3.1.2 Analog To Digital Converter

The analog MRI signal is transformed into a digital matrix and visualized as an image. Every pixel in the image corresponds to a value in the matrix. The original signal from MRI is continuous, where each value in the time corresponds to a value. This continuous signal is transformed into discrete signal using the analog-to-digital converter.

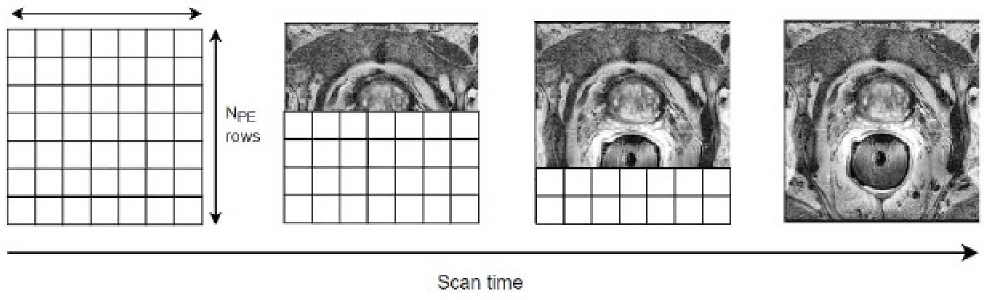


Figure 3.1: Illustration of the process behind capturing a MRI[23]

The pixels in the digital image are sorted into rows and columns in an image array. MRIs also has a third dimension that is called 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 scanner files on row per sequence until the entire array corresponds to the MRI, as illustrated in the **??**[23].

3.2 Neural Networks

Artificial neural networks, simply called neural networks, are computing systems vaguely inspired by the biological neural networks that constitute animal brains.

A neural network is based on a collection of nodes called neurons, which loosely model the neurons in a biological brain. Each neuron calculates and distributes a value via connections to the next layer of neurons. The signal at a connection is a real number, and the output of each neuron is calculated by some non-linear function of the sum of its inputs. The connections are called edges. The edges and neurons have weights that get adjusted as the learning process proceeds. The weight controls the strength of the signal output[3].

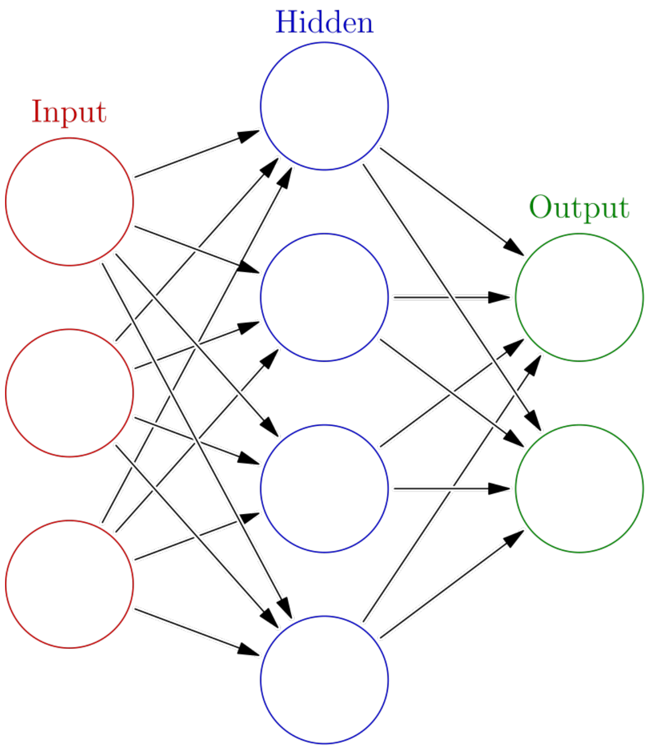


Figure 3.2: A simple neural network[3]

Figure 3.2 illustrates a simple feed-forward network with three input values ($x = [x_1, x_2]$), one hidden layer with four neurons, and two output values ($\hat{y}$).

A neural network contains following components:

3.2.1 Connections and Weights

The network consists of connections, each connection providing the output of one neuron as an input to another neuron. Each connection is assigned a weight that represents its relative importance. A given neuron can have multiple input and output connections.

3.2.2 Backpropagation

Backpropagation, short for "backward propagation of errors," is an algorithm for supervised learning of artificial neural networks using gradient descent. The method calculates the gradient of the error function with respect to the neural network weights.

The "backward" part of the name stems from calculating the gradient proceeds backward through the network, with the gradient of the final layer of weights being calculated first and the gradient of the first layer of weights being calculated last. Partial computations of the gradient from one layer are reused to compute the gradient for the previous layer. This backward flow of the error information allows for efficient computation of the gradient at each layer versus the naive approach of calculating the gradient of each layer separately[5].

3.2.3 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 and learns patterns and regularities automatically 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 labeled and unlabeled data[23]

3.3 Convolutional Neural Networks

A deep convolutional neural network is a deep learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and differentiate one from the other. The technology has existed for a long time, but small data sets 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." [cite: page 326,Deep Learning,2016,[20]]

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

The first layer of a CNN is always a convolution layer. Convolutional layers apply a convolution operation to the input, passing the result to the next layer. A convolution converts all the pixels in its receptive field into a single value. When a convolution is applied to an image, it will decrease the image size and bring all the information in the field together into a single pixel. The final output of the convolution layer is a vector[20].

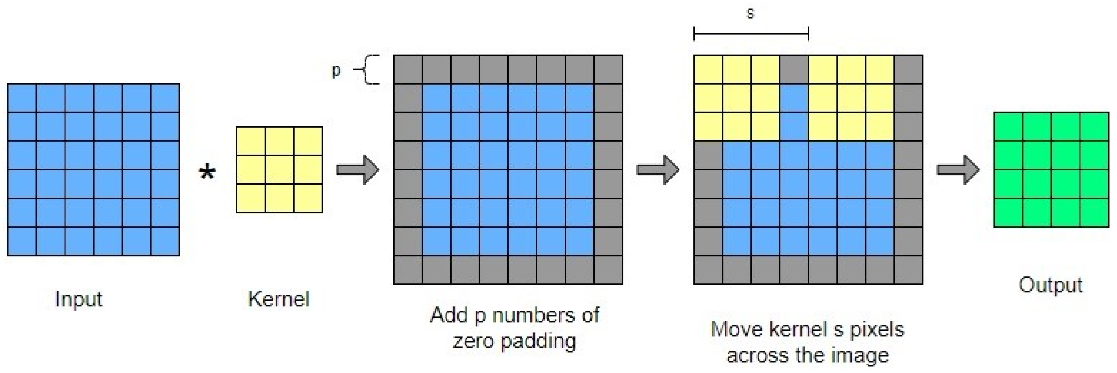

Input
*
Kernel
p
Add p numbers of zero padding
s
Move kernel s pixels across the image
Output

Figure 3.3: Illustration of the process behind Convolutional layer[23].

3.3.2 Transposed Convolutional Layer

The transposed convolutional layer is used during the upsampling of an image. Figure 3.4 demonstrates the process of transposed convolution layer. The output of the transposed convolutional layer. The output of the transposed convolutional layer is also controlled by strides and padding. The output from transposed convolutional is not opposite in terms of values, but it only reverses the spatial dimensions of standard convolution[2].

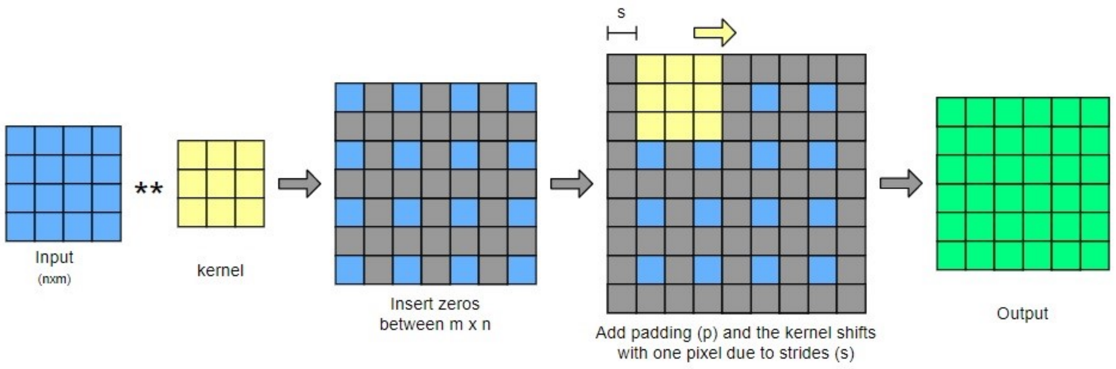

Input
(nxm)
**
kernel
Insert zeros
between m x n
s
Add padding (p) and the kernel shifts
with one pixel due to strides (s)
Output

Figure 3.4: Illustration of the process behind a transposed convolution layer[23]

3.3.3 Dense Layer

A dense layer is a neural network layer connected deeply, which means each neuron in the dense layer receives input from all neurons of its previous layer. The dense layer is the most commonly used in the models.

The dense layer performs a matrix-vector multiplication. The values used in the matrix are parameters that can be trained and updated with the help of backpropagation.

The output generated by the dense layer is an 'm' dimensional vector. Thus, a dense layer is basically used for changing the dimensions of the vector. Dense layers also operations like rotation, scaling, translation of the vector.

This layer is quite beneficial for the fact that it can draw one decision boundary. Also, the number of a neuron depends upon the number of classes in the output[35].

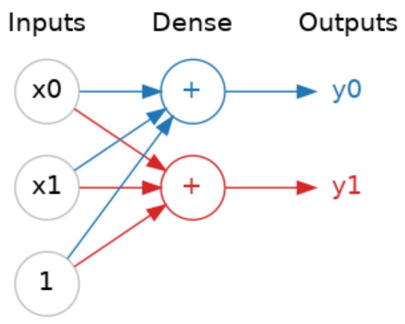


Figure 3.5: Illustration of Dense layer

3.3.4 Augmentation

A significant problem in medical data processing is the lack of data. In order to solve this problem, a technique called augmentation is mainly used to extend the data-set artificially by performing shifting, rotating, horizontal or vertical flip, and blur. The augmentation does not make new images but provides a new version of the same data set. This thesis makes use of OpenCV and Sklearn built-in function for image augmentation.

3.4 Generative Adversarial Network (GAN)

The GAN architecture was first introduced in 2014 by Ian Goodfelloe et al. in the paper titled " *Generative Adversarial Networks*"[[21]]. 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 data-set as true and the generator is 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, where the discriminator acts as a label. The two following subsections explain two GAN architectures that generate images in this thesis[6] Generative Models for Image Synthesis and Image Translation. Machine Learning Mastery, 2019.]

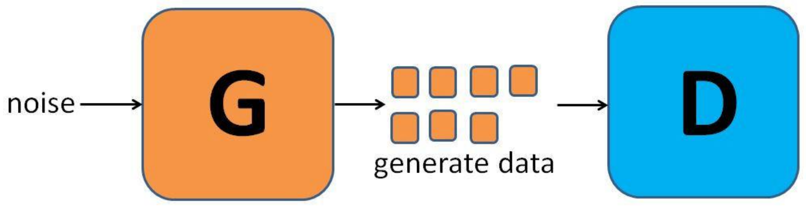


Figure 3.6: A simple GAN Model

3.4.1 Deep Convolutional Generative Adversarial Networks (DCGAN)

The deep convolutional generative adversarial networks, or DCGAN for short, is an extension of the GAN architecture for using deep convolutional neural networks for both the generator and discriminator models and configurations for the models and training the result in the stable training of the generator model.

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 3.7 illustrates the structure of the generator and discriminator in a DCGAN model.

Batch normalization is used while building both the discriminator and the generator. This mainly tackles two problems in DCGAN and deep neural networks in general.

1. It normalizes the input to each unit of a layer.
2. It also helps to deal with poor initialization that may cause problems in gradient flow.

[cite:researchgate publication]

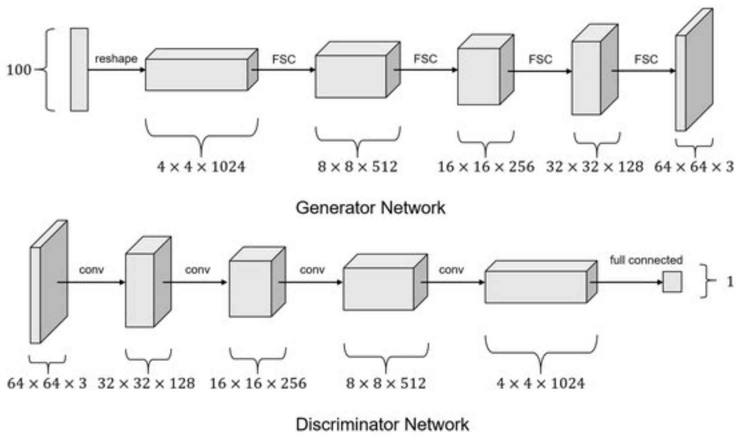


Figure 3.7: Illustration of basic principle of DCGAN Model [42]

3.4.2 Conditional Generative Adversarial Network (cGAN)

The conditional generative adversarial network, or cGAN for short, is an extension to the GAN architecture used as a machine learning framework for training generative models. The idea was first published in a 2014 paper titled *Conditional Generative Adversarial Nets* by Mehdi Mirza and Simon Osindero[27].

cGAN is a deep learning method where a conditional setting is applied, meaning that both the generator and discriminator are conditioned on some auxiliary information such as class labels or data from other modalities. As a result, the ideal model can learn the multi-modal mapping from inputs to outputs by feeding it with different contextual information.

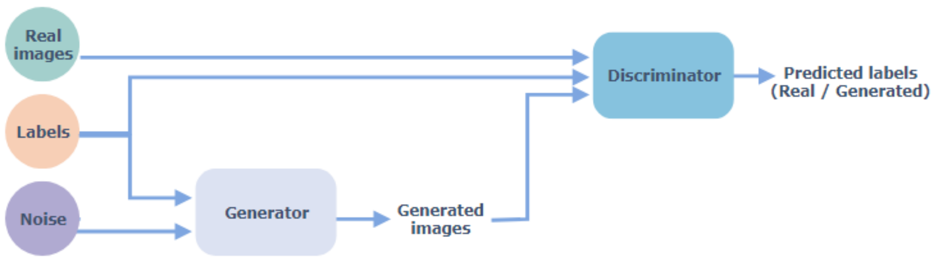


Figure 3.8: Illustration of basic principle of cGAN model[39]

The figure 3.8 illustrates cGAN architecture where class labels are provided to generator and discriminator. Usually, in GAN, we cannot control what specific images generator will produce. In other words, there is no way a requested particular image can be produced.

This is where the cGANs come in, as we can add an extra layer of one-hot-encoded image labels. This additional layer guides the generator in terms of which image to produce.

The input to the additional layer can be a feature vector derived from either an image that encodes the class or a set of specific characteristics we expect from the image. [39]

3.5 VGG16

VGG16 is a convolution neural network for classification and detection proposed by K. Simoyan and A. Zisserman from the University of Oxford in the paper *Very Deep Convolutional Networks for Large-Scale Image Recognition*[37]. The model achieves 92.7 % top-5 test accuracy in **ImageNet**, which is a data-set of over 14 million images belonging to 1000 classes. It improves **AlexNet** by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. The most unique thing about VGG16 is that instead of having a large number of hyper-parameter, they focused on having convolution layers of 3×3 with a stride 1 and always used the same padding and max pool layer of 2×2 of stride 2. It follows this arrangement of convolution and max pool layers consistently throughout the whole architecture. In the end, it has 2 FC (fully connected) layers followed by a softmax for output. The 16 in VGG16 refers to it has 16 layers that have weights as shown in figure 3.9. It is a pretty extensive network, and it has about 138 million (approx) parameters. VGG16 was trained for weeks and was using Nvidia titan black GPUs[38].

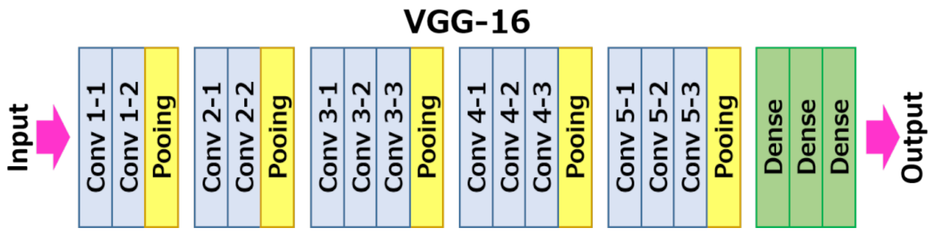


Figure 3.9: Layers of VGG16 network[38].

3.5.1 The Architecture

The architecture depicted below is VGG16:

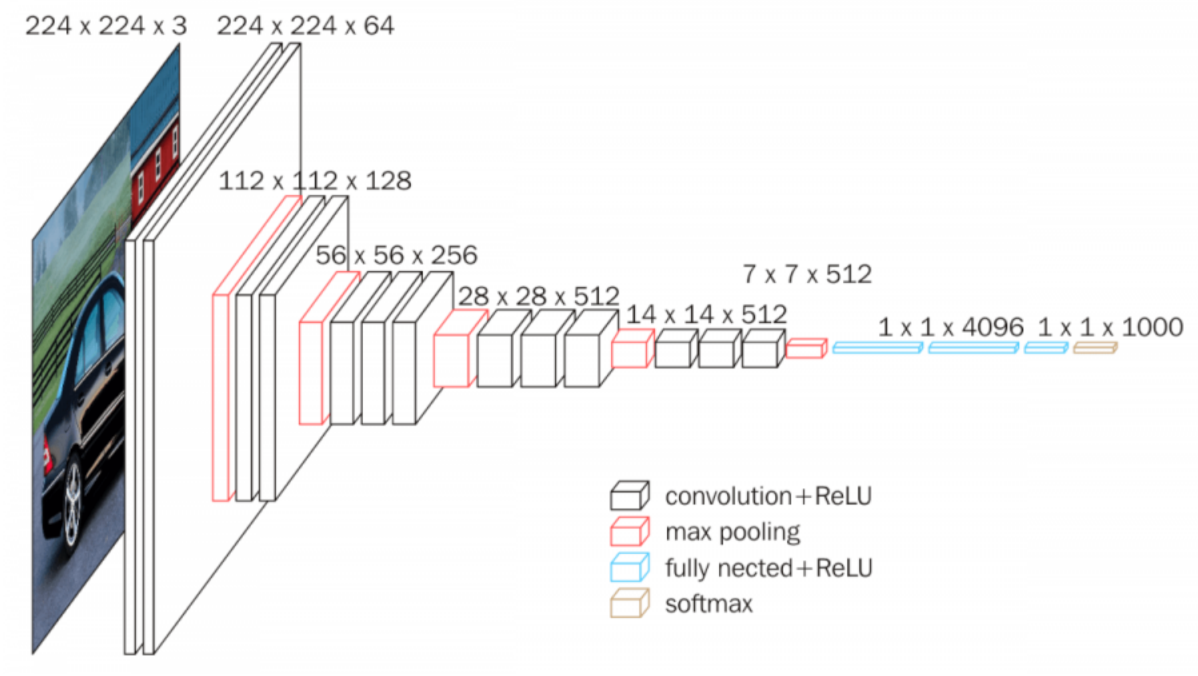


Figure 3.10: Architecture of VGG16 [38].

The input to *conv*1 layer is of fixed size 224×224 RGB image. The image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very small receptive field: 3×3. In one of the configurations, it also utilizes 1×1 convolution filters, which can be seen as a linear transformation of the input channels followed by non-linearity. The convolution stride is fixed to 1 pixel; the spatial padding of conv. Layer input is such that the spatial resolution is preserved after convolution, i.e., the padding is 1-pixel for 3×3 conv. layers. Spatial pooling is carried out by five max-pooling, which some of the Conv. layers (not all the conv. layers are followed by max-pooling). Max pooling is performed over 2×2 pixel window, with stride 2.

Three fully connected (FC) layers follow a stack of convolutional layers. The first two have 4096 channels each. The third contains 1000 channels (one for each class). The final layer is the softmax layer. The configuration of the FC layers is the same in all networks. All the hidden layers are equipped with rectification (ReLU) non-linearity.

3.5.2 Drawbacks

There are two major drawbacks[38] with VGGNet:

1. Its is extremely slow to train
2. The network architecture weights themselves are quite large thus making its deploying a tiresome task.

3.6 Softwares

The technical part of this thesis is implemented with the programming language named Python. Python is a high-level, multi-purpose programming language. A high-level programming language makes the development process simpler by the use of natural language. A general-purpose programming language is used to develop a wide range of software applications.

Python has a wide range of external libraries with additional functions. This section describes some of the main libraries used for implementing models in this thesis.

3.6.1 Tensorflow

Tensorflow is an interface and implementation to express and execute machine learning algorithms. The library can implement a wide variety of algorithms for deep neural networks, like training and inference algorithms[1].

3.6.2 Keras

The DL application programming interface (API) used in this thesis is Keras. Keras library uses Tensorflow to enable rapid experimentation and implementation of DL ideas[13].

3.6.3 Numerical Python

The Numerical Python (NumPy) library is introduced into the Python programming language to analyze and implement high-level scientific computing and data analysis of numeric data and multi-dimensional arrays. This library is used for many tasks ranging from generating random integers or arrays to advanced mathematical functions. NumPy is also employed to use by other libraries like Tensorflow to generate Tensor objects and more[**harris2020array**].

3.6.4 OpenCV

OpenCV is an open-source library primarily used for computer vision, image processing, and machine learning. The usage of this library in this thesis allows performing several actions on the medical images, ranging from image pre-processing to image augmentation. When this library is coupled with other various libraries, such as NumPy, Python can process OpenCV array Structures[**opencv˙library**].

3.6.5 Pydicom

DICOM (Digital Imaging In Medicine) is the bread and butter of medical image datasets, storage, and transfer. Pydicom is a pure python package working with DICOM files such as medical images, reports, and radiotherapy objects. pydicom makes it easy to read these complex files into natural pythonic strictures for easy manipulation. The most common use of pydicom is to read an existing DICOM file, alter some items, and write it back out again[26].

pydicom is not a DICOM server (see pynetdicom instead) and is not primarily about viewing images. It is designed to let you manipulate data elements in DICOM files with Python code.

3.6.6 Scikit-learn

Scikit-learn is an open-source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection and evaluation, and many other utilities. It provides dozens of built-in machine learning algorithms and models, called estimators. Each estimator can be fitted to some data using its fit method[30].

Chapter 4

Data-set and Image Pre-processing

The data-set of prostate MRI used in this thesis is a part of PROSTATEx Challenge data and was collected by performing a clinical examination. MRI scans at the Radboud University Medical Centre (Radboudumc), Netherlands, in the Prostate MR Reference center under the supervision of prof. Dr. Barentsz. The data-set was collected and curated for research in computer-aided diagnosis of prostate MR under the supervision of Dr. Huisman, Radboudumc. The two different Siemens 3T MR scanners, the MAGNETOM Trio and Skyra, were used to collect the images. No endorectal coil was used in the acquiring of the images. Figure ?? shows a randomly taken ADC MRI slice with a corresponding mask from the PROSTATEx data-set.

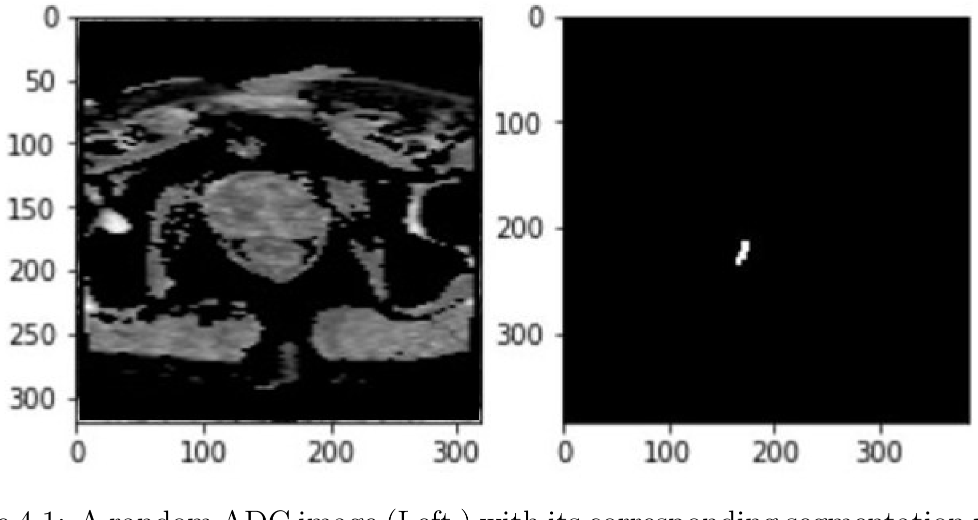


Figure 4.1: A random ADC image (Left) with its corresponding segmentation masks from PROSTATEx data-set

The PROSTATEx challenge aimed to focus on the quantitative methods for the medical images analysis to classify the clinically significant prostate cancer, and it was held in conjunction with the 2017 SPIE Medical Imaging Symposium[24]. The relevant data-set used in this thesis contains 201 subjects split into training, testing, and validation data. The images have all kinds of stored information in the metadata, like name, age, slice thickness, etc. The mask of each MRI case provides information on the location, size, and shape of the prostate lesion present in that case.

MRIs relate to a bundle of 2D images that adds up to show three-dimensional (3D) images. Due to varying data protocols, changing parameters in data makes it acceptable in medical clinics worldwide. Different data sets are available in the cancer imaging archive, containing a variety of medical prostate MRIs. However, the data in this thesis is using the DICOM format of medical images[18].

4.1 Image Pre-Processing

This section provides a discussion about the pre-processing techniques used in this thesis. The pre-processing of the data-set is inspired by the work presented on image data in Data Science Bowl held in 2017 by Booz Allen Hamilton and Kaggle[16]. The operations performed for pre-processing are explained as follows.

4.1.1 Loading of Data

The images are downloaded from The Cancer Imaging Archive (TCIA) using an NBIA data retriever to download the required DICOM images. Afterward, these images are then loaded and analyzed for further processing in Jupyter Notebook using the pydicom library to work with Dicom files in Python. The figure 4.2 shows the random slices from ADC modality.

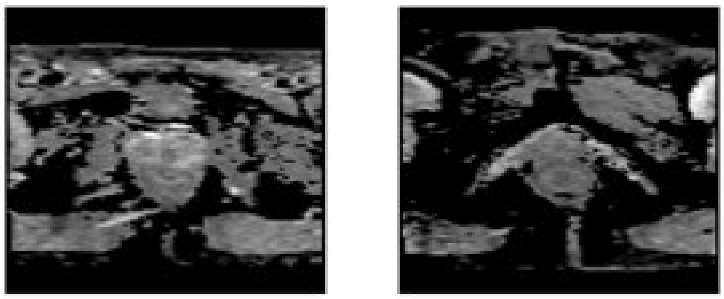


Figure 4.2: MRI slices of Prostate for ADC modality [cite:https://neurohive.io/en/popular-networks/vgg16/].

The table 4.1 shows the details of the loaded images with number of slices with respect to the dimensions of slices for ADC modality.

Data Specifications	**ADC**	**ADC**	**ADC**
Width (pixels)	75	84	128
Height (pixels)	128	128	128
Number of slices	162	5640	100

Table 4.1: Table showing the number of MRI slices and the correlated image sizes for ADC modality.

The masks for each case are present in nii or NIfTI format, primarily used for imaging informatics for neuroimaging. These masks are extracted and loading using the nibabel library in Python. The information of the masks for each case is present in `PROSTATEx_Classes.csv` file, based on whether the mask is clinically significant or not.

4.1.2 Data Filtering

The `image_list.csv` file contains the information about the clinically significant images or slices out of all the images or slices for every case. The relevant significant information for each case is present in this CSV file for ADC images. The regex library filters the data-set to extract the relevant images based on the information from the said CSV file. The filtered data is then stored and copies separately from the original data with the original names along with the patient data for the copied slices that is every case is named as ProstateX-[patient num] where patient number ranges from 0 to 201. The information for cases number 52, 82, and 138 is missing in the data set.

4.1.3 Data Reshaping

CNN trains on images that have similar dimensions. A group of images with different dimensions can be used to train the same neural network but separately. The CNN is designed to fit the dimensions of images in the data set to get better results. The data-set comprising ADC images has height, and weight ranging 96×96 to 208×208 were reshaped to the resolution of 128×128 using OpenCV built-in resize function. The channel represents the depth of the image, RGB images have three channels, and grayscale images have one channel. MRI images are grayscale and have one channel. For VGG16, the pictures are made three-channel using the merge command and copy the same picture three times. Table 4.3 gives the details of total slices for each modality and their respective dimensions after reshaping.

Reshaped Data Specifications	**ADC**
Total Cases	201
Total Number of slices	1285
Final Reshaped width (Pixels)	128
Final Reshaped height (Pixels)	128

Table 4.2: Table showing the number of MRI slices and the correlated image sizes for ADC modality.

4.1.4 Slice Operation and Data Organization

In this thesis, the total subjects used are 201, and 1285 slices are present in the ADC image data-set for all the subjects. This data set contains both the significant and non-significant tumor slices of the subjects. Non-significant slices are those in which tumors are benign and not harmful to the body. Significant slices are those in which the tumor is malignant and is harmful to the body. The slices are sorted and stored in two different arrays based on the presence of lesions and labels. Then, the sorted and stored data is divided into three different arrays for the formation of training, testing, and validation data. Around 80% of both significant and non-significant slices are stored in training data-set, 10% is for testing and validation each. During the stratification, it is strictly considered to put all the slices of one patient in one data set to avoid leaking lesions between different data sets. DCGAN is trained on significant and non-significant separately; however, for training CGAN, both data-sets are used simultaneously using labels as a second input for it.

Details of ADC Data-set	**ADC**
Total data-set	1285
Training data-set	894
Test data-set	203
Validation data-set	188

Table 4.3: Details of Image stratification

4.1.5 Data Normalization

Neural Networks (NN) usually calculate small weights to proceed with input images. The pixel values in most of the images are integers, ranging from 0 to 255. The larger pixel values can make the learning process slow and cost computing efficiency; also, weight decay and Bayesian estimation can be done more conveniently with standardized inputs. Therefore, normalizing the pixel values to Normal or Gaussian distribution is often considered if normalization is done by the standard deviation[12]. In this thesis, normalization is achieved using the equation **??**.

$$Normalization = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{4.1}$$

In equation 4.1, X is the original image pixels, X_{min} is the minimum pixel value and X_{max} is the maximum pixel value of the image. The images are normalized to have a range of $[0, 1]$. The data is also standardized to have unit variance, and zero mean.

4.2 Save Organized Data

In this thesis, the Numpy library saves the organized data as a four-dimensional NumPy array in a file with an extension .npy. The first index of the array represents slices for all the subjects; the next two indexes represent the height and width of the images. The last index represents the depth of the image, that is, several channels. In this thesis, the number of channels is equal to one when training the GAN architecture, and several channels are three when running the VGG16. All the images used in this thesis are gray-scale. The input shape of the stored NumPy array in DL architecture is (number of slices, height, width, channels).

Chapter 5

Solution Approach

5.1 Existing Approaches

The paper *Medical Image Synthesis for Data Augmentation and Anonymization using Generative Adversarial Networks* written by Hoo-Chang Shin et al.[[36]], uses Pix2Pix to generate MRIs of the brain. This paper uses real segmentation masks and is not generating new ones. Masks used in this paper have multiple classes and black backgrounds surrounding the brain.

A master thesis with the title *Data augmentation in deep learning using generative adversarial networks*[28] written by Thomas Neff for the Graz University of Technology uses Wasserstein GAN (WGAN) to generate both X-radiation (x-rays) images of the lung and their corresponding masks.

Another paper with the title *Improving Prostate Whole Gland Segmentation In T2-Weighted MRI With Synthetically Generated Data* written by Alvaro Fernandez-Quilez et al.[17], uses DCGAN and Pix2Pix GAN to improve the quality of segmentation of whole gland (WG) in T2W images. This paper generates new images and their corresponding masks to improve the segmentation of the whole gland compared to standard augmentation methods.

5.2 Proposed Method using GAN to Expand the Data-set

The testing network named VGG16 requires images and labels as input to learn the features and classify the tumors into significant and non-significant. In this thesis, two GAN architectures are being used to extend the data set. However, both of these architectures are independent of each other. The first architecture is DCGAN that generates images without any labels, and the second architecture is conditional GAN (cGAN) that generates images along with labels and can generate images of specific classes based on these labels. Both GAN networks contain a generator and a discriminator, designed as two models that update weights separately. DCGAN and cGAN only train on data that contain tumors, both malign and benign, thus narrowing the data-set to 1285 samples.

This thesis has a two-step approach; in the first step, data is generated using GAN networks, and in the second step, this generated data is used to augment the original data and used as an input to the classifier, which in our case is VGG16 in order to compare the results of the augmentation using conventional methods and augmentation using GAN.

5.2.1 DCGAN Methodology

DCGAN is used to generate images, as the training process has proven to be more stable than the original GAN.

DCGAN trains for 5,000 epochs, where each epoch takes around 5 seconds. The training process is performed on the University of Stavanger gorina6 server, which uses Nvidia Tesla V1000 GPU with 32 GB memory. The GAN discriminator is used to measure the performance. The GAN is appropriately trained when the discriminator can no longer distinguish between real and generated images. Other than this, there are no evaluation metrics to know when the generated images are realistic, and the training process should stop. The network train for 5000 epochs and share visual of generated images after every 100 epochs. The batch size contains 128 samples recommended by the original paper[21], and each epoch is completed in seven steps. The data-set used in the original paper[21] contained 3 million samples, while our data-set has only 367 samples with significant tumors and 918 non-significant tumors.

DCGAN is trained separately for both significant and non-significant tumors because it cannot generate specific images when provided a label, so in order to avoid mismatch of the generated samples, this method is being adopted. All models train using adam optimizer with a learning rate of 0.0002 and momentum (β_1) equal to 0.5, as recommended by the original paper[21].

Model Design

The DCGAN implementation is inspired by Jason Brownlee's work on DCGAN for gray-scale mnist handwritten digit data-set[6] and the original DCGAN architecture.

Jason Brownlee's work on DCGAN generates images of size 28×28 pixels, and original paper[21] generates images of 64×64. In this thesis, the model used in the original paper[21] and Jason has been modified to generate images with 128×128.

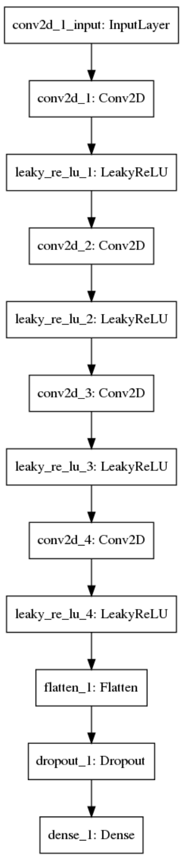
Developing a DCGAN requires both a discriminator neural network for classifying whether the image is real or generated and a neural generator network that transforms the input into a two-dimensional image. The discriminator learns the difference between real and generated images. A visual illustration of the model is shown in figure 5.1. The implementation of the discriminator model is inspired by the original paper[21] and has been extended by adding two convolutional layers to change the input to 128×128. The output is a single neuron with a sigmoid activation function, and the loss function is a binary cross-entropy, as recommended.

- **Discriminator input**: A half batch of real images and a half batch of generated images with 128×128 pixel size and one channel.
- **Discriminator output**: A single output, which determines whether the input is real or generated.

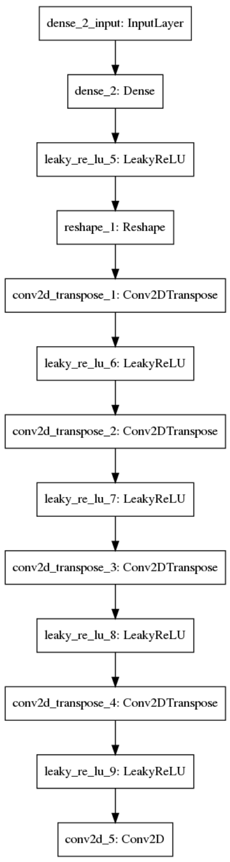
The DCGAN generator is responsible for creating new, fake, but plausible images. The model input corresponds to a 100 elements vector of Gaussian distributed random numbers. A visual illustration of the model is shown in figure 5.1.

- **Generator Input**: A vector containing 100 elements of Gaussian distributed random numbers.
- **Generator Output**: Two-dimensional square gray-scale image of 128×128 with pixel values in [0,1].

The first two layers of the generator are dense and reshape, which convert the input array into a 2D array with a shape of $8 \times 8 \times 128$. The dense layer has an activation size of 8192. The generator uses the transpose convolution layer to learn weights while up-sampling the input. The transpose convolution layer uses the kernel with a width and height of 5 and 2 number of strides. All the transpose convolutional layers use the same kernel and strides while the number of filters decreases by half with each successive layer. The first layer has a 1024 number of the filter size, and the last transpose convolutional layer has 128 filters, thus making 4 transpose convolutional layers.



Discriminator model



Generator model

Figure 5.1: Proposed Discriminator and Generator model for DCGAN

All the transpose convolutional layers use LeakyReLU with a α value of 0.2. The last layer in the generator is a convolutional layer with 1 filter and kernel size of (7×7) to get the output for the required size, which in our case is 128×128 with a channel of 1 and uses a TanH activation function to ensure output values in the range of $[-1, 1]$.

The generator is updated based on the feedback from the discriminator. This model uses the latent size of 100 for the generator's input and feeds the generated images to the discriminator. The discriminator classifies each input and returns a value to update the generator weights. The generator is not compiled individually, while the discriminator and the combined model are compiled. After the training, the generator can be used to produce output. More detailed model designs are in Appendix B.

5.2.2 cGAN Methodology

GAN models can generate new random images for a given data set, but there is no way to control the types of images generated other than trying to figure out the complex relationship between the latent space and the generated images[8].

The conditional generative adversarial network, or cGAN for short, is a type of GAN that involves the conditional generation of images by a generator model. Image generation can be made conditional on a class label, allowing the targeted generalization of images of a given type. The cGAN was first described by Mehdi Mirza and Simon Osindero in their 2014 paper titled "Conditional Generative Adversarial Nets."[27] In the paper, the authors motivate the approach based on the desire to direct the image generation process of the generator model.

The architecture consists of a generator and a discriminator model. The generator model generates new images that are indistinguishable from actual images in the data set. The discriminator model is responsible for classifying a given image as either real or generated.

DCGAN is effective at image synthesis and can generate new examples of images for a target data-set. Some data sets have additional information in the form of class labels, and it is desirable to use this information. There are two reasons for making use of the class label information in a GAN model.

- Improve the GAN
- Targeted image generation

Additional information related to the input data-set, such as class labels, can improve the GAN. The improvement may be in the form of more stable training, faster converging , and generated images that have better quality. Class labels can also be used for the targeted generation of images of a given type. This type of model is called Conditional Generative Adversarial Networks, or cGAN for short.

The network trains on both labels and MRIs obtained from the PROSTATEx data-set, but only those with significant and non-significant slices. The network is trained for 6000 epochs where each epoch takes around 12 seconds on the University of Stavanger gorina6 server, which uses Nvidia Tesla V1000 GPU with 32 GB memory. The batch size is equal to 128, and visual examples are printed after every 100 epochs. All models use Adam optimizer with a learning rate equal to 0.0002 and momentum (β_1) equal to 0.5, as recommended in the original paper[27]. The cGAN can be trained on both significant and non-significant tumor data-set provided when labels are available for both of them.

Model Design

Starting with the discriminator model, a second input is defined that takes a value for the class label of the image. This makes the input image depended on the class label that was provided to the model. The class label is then passed through an embedding layer with a size of 50, which means that each of the two classes in the PROSTATEx cancer data will map to a different 50-element vector representation that the discriminator model will learn.

The output of the embedded layer is then passed to a fully connected layer with a linear activation. The fully connected layer has enough activations to be reshaped into one channel of a 128×128 image. The activations are reshaped into a single 128×128 activation map and concatenated with the input image. This has the effect of looking like a two-channel input image to the next convolutional layer. The parameterized shape of the input image is also used after the embedding layer to define the number of activations for the fully connected layer to reshape its output. The number of classes in the problem is also parameterized in the function and set.

In order to make architecture clear, figure 5.2 is a plot of the discriminator model. The plot shows the two inputs: first, the class label that passes through the embedding layer(left) and the image (right), and their concatenation into a two-channel 128×128. The rest of the model is identical to the discriminator explained in section 5.2.1.

- **Discriminator Input**:Half batch of real images and half batch of generated images with 128×128 pixel size and labels for the images.
- **Discriminator Output**: A single output, which determines whether the input is real or generated.

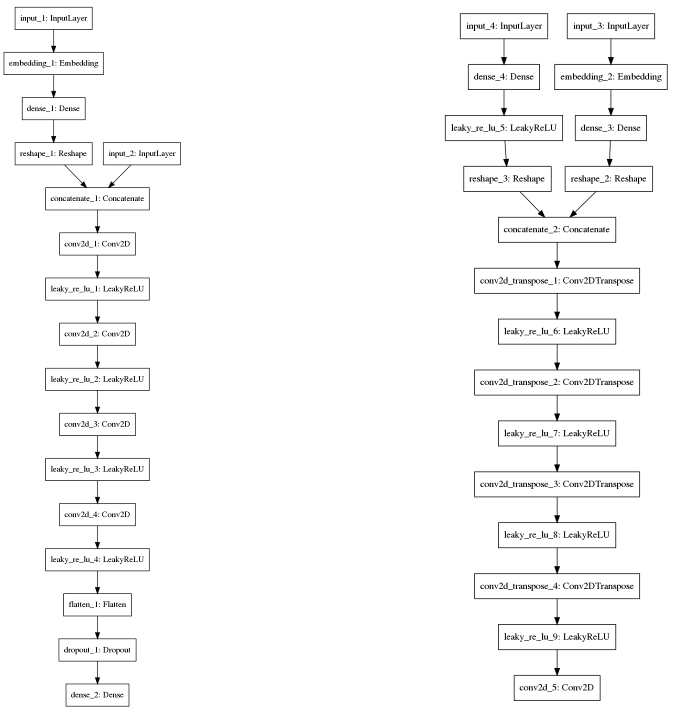
Next, the generator model must be updated to take the class label. This has the effect of making the point in the latent space conditional on the provided class label.

Similar to discriminator, the class label is passed through an embedding layer to map it to a unique 50-element vector and is then passed through a fully connected layer with a linear activation before being resized[8]. In this case, the activations of the fully connected layer are resized into a single 8×8 feature map. This is to match the 8×8 feature map activations of the unconditional generator model. The new 8×8 feature map is added as one more channel to the existing 128, resulting in 129 feature maps that are then up-sampled as in the prior model[8].

In order to make architecture clear, figure 5.2provides a plot of the conditional generator model.

- **Generator Input**: A vector containing 100 elements of Gaussian distributed random numbers along with a label.

- **Generator Output**: Two-dimensional square gray-scale image of 128×128 with pixel values in $[0, 1]$.



Discriminator model Generator model

Figure 5.2: Proposed Discriminator and Generator model for cGAN

In this case, the $100-element$ point in latent space as input and subsequent resizing (left), then the concatenation of the two sets of feature maps (center). The remainder of the model is the same as the generator model explained in section 5.2.1.

Finally, the new GAN model will take a point in latent space as input and a class label to generate a prediction of whether the input was real or fake. It is crucial to connect the image-generated output from the generator and the class label input, both as input to the discriminator model. This allows the same class label input to flow down into the generator and down into the discriminator[8].

The figure 5.3 summarizes the composite GAN model. It shows the generator model in full with the point in latent space and class label as input, and the connection of the output of the generator and the same class label as input to the discriminator model and the output of a single class label classification of real or fake.

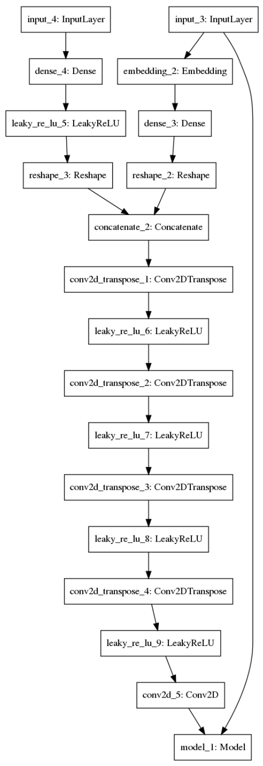


Figure 5.3: cGAN Proposed Model

More detailed model designs are in Appendix B.

5.2.3 VGG16

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a data-set of over 14 million images belonging to 1000 classes. [38]

Model Design

In this thesis, pre-trained weights of the VGG16 network are used along with an output layer whose output will be equal to the required number of classes. In order to add pre-trained weights, the weights will be downloaded from the keras api. The weights of the VGG16 are big in size and covers 512 MB. The data-set contains 3 directories: Training, Validation and Testing. Each directory contains sub-directories with images of classes which in this thesis is significant tumors and non-significant tumors. The weights of the VGG16 are loaded without the top layer to change the input shape to 128×128 from 224×224 RGB image the image has 3 channels. In order to convert, 1 channel into 3 channel image, single channel image is merged together 3 times using NumPy python library. The layers of the VGG are excluded from training phase because they are already trained because of the *imagenet* weights. The activation function used in the output layer in this case is *softmax* because VGG16 is a multi-label classification problem. Loss function used is *categorical crossentropy* with adam optimizer, The metrics used are accuracy and AUC to monitor the network with an EarlyStopping as a checkpoint. More detailed model designs are in Appendix B.The batch size used is 32 and number of epochs used is 300. The input and output of the model is listed below:

- **VGG16 Input:** Image with a 128×128 pixel size with 3 channels.
- **VGG16 Output:** Output array with a size of 2 because input has two classes.

The figure 5.4 shows proposed model of VGG16

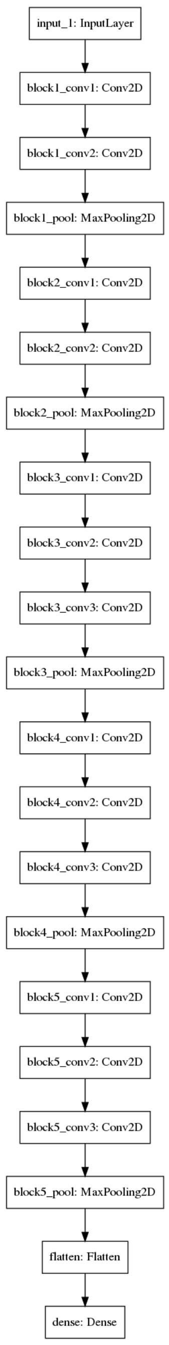


Figure 5.4: Proposed VGG16 Model

Chapter 6

Experimental Evaluation and Results

[l] In this chapter, the experiments, classification evaluations, and results are proposed. Various configurations for the models are also presented, and evaluation of the results produced by VGG16 using DCGAN and cGAN.

6.1 Experimental Setup

The experiment is performed by augmentation of the organized data-set using various methods and then evaluating the augmented data-set using the VGG16 classification network. The loss function under observation is the area under the curve (AUC), and ROC curves evaluate the model performance during training. This thesis compares the conventional augmentation techniques like translation, rotation, and horizontal flipping of the data with the data generated by the GAN architectures. VGG16 network is trained for 1000 epochs with early stopping callback Keras function that monitors the validation AUC with the patience of 20 epochs, meaning that it will stop the training process if the validation AUC does not change for 20 consecutive epochs.

In this thesis, VGG16 is used with pre-trained weights and then adds a new output layer with two classes. To use the pre-trained weights, a new data set is added that contains the weights. The data set is divided into three groups for training, validation, and testing purposes. The weights of the VGG are loaded without the top layer. These weights are trained on *Imagenet* data-set. The network uses the Adam optimizer with a learning rate of 0.001. The network monitors the validation AUC and accuracy, only saving the best model using the checkpoint.

6.2 Selected configuration for DCGAN

Generative Adversarial networks are not easy to train and require precise configurations because if the discriminator is too good at detecting real and generated images, it will stall the generator's performance, resulting in a failure mode where the generator cannot generate any good images. If the discriminator is too weak then generator, it will take any input and classify it as real thus will not improve the generator which results in another failure mode.

For a reason mentioned above, DCGAN has to be configured differently for different image sizes. The model used in this thesis is inspired by Jason Brownee[7] model. The model has been modified to fit for 128×128 input image size by adding extra layers and increasing the filter size to accommodate the large pixel size along with other changes. Hype parameters have been modified to fit the desired results. High learning rates (LR) like 0.01 and 0.1 were used, which lead to quick failure mode where the discriminator loss is zero while generator loss is high. Alongside this, small learning rates 0.00001 and below were also tested which lead to very slow development of images at the start and then generator blows out quickly which results in GAN training failure.So, after testing various settings, the learning rate of 0.0002 was used with the α_1 value of 0.5 DCGAN is very sensitive to the hyper-parameters, for example, number of filters, number of epochs, kernel size, number of layers. The convolutional and transpose convolutional layers in discriminators and generators are four, with an equal number of filters ranging from $[512, 64]$ and strides size of 2 for both models. Selected kernel size for discriminator is $(3, 3)$ while for the generators is $(4, 4)$. The number of epochs used for the training is 10,000 with a batch size of 128, taking one week for training. DCGAN generator output was stored and visualized after every 1000 epochs which will be discussed in the next chapter. After testing various settings, the configurations that are used for DCGAN are as follows:

Parameters for DCGAN	**Value**
Batch Size	128
Optimizer	Adam (Lr = 0.0002 $\beta_1 = 0.5$)
Latent Dimension	100
Kernel size (Discriminator)	(3,3)
Kernel size (Generator)	(4,4)
Number of epochs	10,000

Table 6.1: Selected Hyperparameters for DCGAN

he images for training of the GAN are available in Appendix C.

6.3 Selected configuration for cCGAN

cGAN training parameters differ from the DCGAN because it takes more inputs in the form of labels to generate targeted images. The model in this thesis is inspired by Jason brownee [8]with modifications to use the input images with a pixel size of 128 $\times$ 128. The input labels are filtered in corresponding with their images. The images and labels are concatenated in both generators and discriminators to overcome the limitation of DCGAN where it cannot generate targeted images.

In cGAN, the number of convolutional and transpose convolutional layers are similar that is four with a filter size ranging from 1024 for the first layer and 128 for the last convolutional and transpose convolutional layer of both generators and discriminators. cGAN becomes very unstable by changing small values. The computational time required to train cGAN is higher than DCGAN; hence the number of epochs to train is limited to 5000, which took around one week to complete with a batch size of 128. The generator's output and visualization were saved after every 1000 epochs. The learning rate used is 0.0002 with α value of 0.5 for Adam optimizer. Kernel size for discriminator is $(4, 4)$ and for generator is $(5, 5)$ with a stride size of 2. The latent size used in cGAN is 100; when latent size increases, it increases the computational time per epoch and requires more layers to fill the latent space with the required parameters. The recommended number of latent dimensions is 100 and is being used for the cGAN architecture. The configurations used for cGAN are as follows:

Parameters for DCGAN	**Value**
Batch Size	128
Optimizer	Adam (Lr = 0.0002 β_1 = 0.5)
Latent Dimension	100
Kernel size (Discriminator)	(4,4)
Kernel size (Generator)	(5,5)
Number of epochs	5,000

Table 6.2: Selected Hyperparameters for cGAN

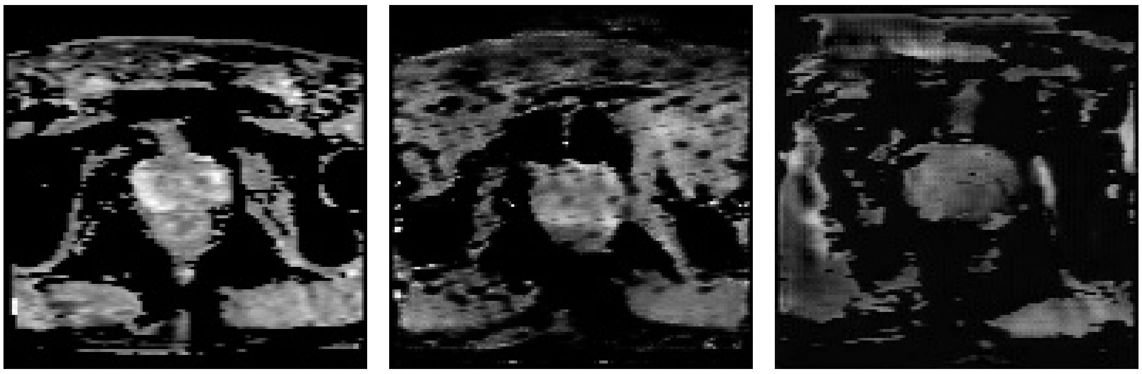
The images for training of the GAN are available in Appendix C.

6.4 Comparison of generated images from DC-GAN versus cGAN

After training both DCGAN and cGAN, images are generated from the saved generator file of each GAN architecture. For further use, the GAN image quality is compared with the original images by visual inspection.

The cGAN generator can generate the images by providing two inputs arguments to it, an input label and latent points with dimension 100. cGAN takes the input from the user and generates specified images with their specified labels. In this thesis, the labels used for non-significant is **0** and for significant images is **1**. Images are also generated from the DCGAN generator by providing one input argument, latent points with dimension 100. DCGAN can generate multiple types of images, but it can not be targeted; hence for multi-class image generation, cGAN is preferred.

The images from both the architectures are presented in the figure **??**, with cGAN image in the center and DCGAN image at the right side of the figure, along with an MRI image from the original data-set at the left side of it. It can be observed from the images that the image generated by the cGAN (center) is well structured, while the image generated by the DCGAN is blurry and has not acquired prostate structure as that of the cGAN. When the images are compared with the real images, one can notice that the cGAN image is close to the real image. It captured the overall structure of the prostate from the data-set and is close to the real image in the sense that it can generate images with the prostate visible. The image generated by the DCGAN has acquired the overall structure of the MRI image, but it does not come close to the cGAN and real image.



Real image | Image generated by cGAN | Image generated by DCGAN

Figure 6.1: Comparison of Images

In this thesis, the images generated from cGAN are used to augmentation of the original data set. The decision is based on comparing results from two different architectures used in this thesis. The images generated by the cGAN have more control over the variety and type of the images (significant or non-significant tumors) compared to the DCGAN, which helps generate more versatile images compared with DCGAN.

6.5 Augmented Images

Various conventional augmentation methods exist to increase the data-set. In this thesis, the augmentation methods used to evaluate the binary classification result are translation, rotation, and flipping of the original data. The data augmentation techniques used will increase the size of the data-set by the same amount as the original data. In our thesis, 918 images containing non-significant tumors and 367 images containing significant tumors will be increased by 100% and will augment the same number of images as the original data set. After augmentation, the data-set size will be equal to 1836 images containing non-significant tumors and 734 images containing significant tumors.

6.5.1 Translated Images

In this section, image translation is discussed. Translation means that the image is shifted by adding or subtracting the X/Y coordinates, and the image is moved within the frame of reference. It is accomplished by adding a transformation matrix that shits the image in the required direction. Figure C.6 shows a translated image with a real image.

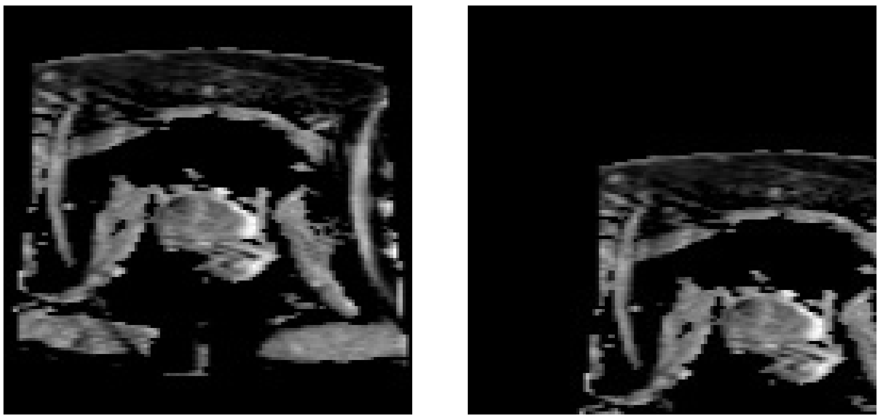
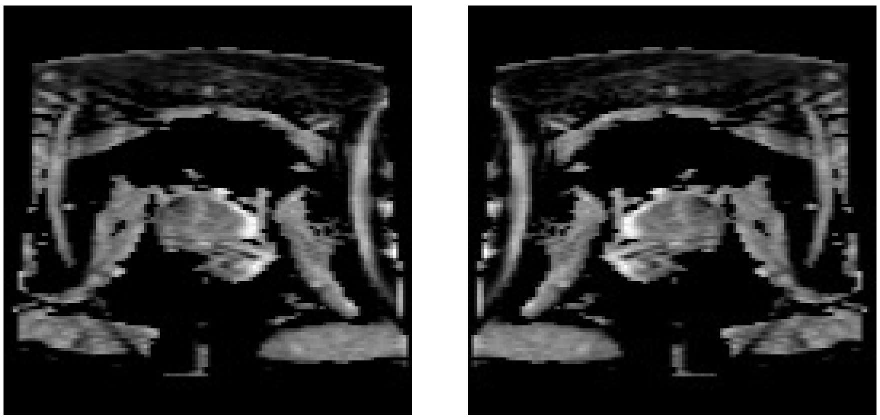


Figure 6.2: Comparison of Real and Translated image

6.5.2 Flipped Images

This section explains the flipping of images. NumPy inbuilt function np.fliplr is used to flip the image. Flipping means that a mirrored version of the picture is produced. This data augmentation type is used widely for increasing the data set. The image on the right side of figure 6.3 shows a horizontally flipped image along with an original image. The reason not to do a vertical flip was to avoid an unrealistic situation in which the network will never come across, or there will be an MRI with an upside prostate in it.



Real Image   Flipped Image

Figure 6.3: Comparison of Real and Flipped image

6.5.3 Rotated Images

Rotation of the image is one of the image transportation operations applied to augment the data set. The image is rotated around the center of the axis by a specified degree, which is 25 degrees. In the figure 6.4 shows a 25°rotated image along with the real image. If the image is rotated more than this, then it becomes distorted and loses its original form.

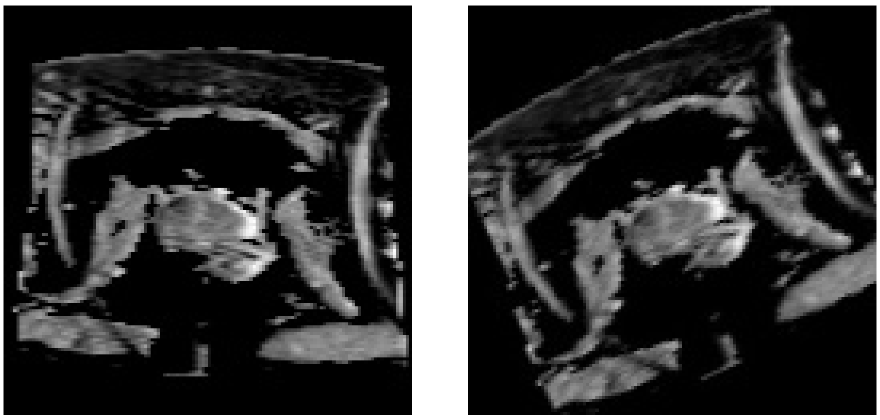


Figure 6.4: Comparison of Real and Rotated image

6.6 Model Performance

In this section, the performance of classifiers is tested against different configurations of the test data-set for both generated and augmented methods. The section will also explain the comparative results for both generated images from cGAN and augmented images using conventional methods. The metric that is used to compare the results for the various configurations is macro AUC-ROC curves. The ROC curve is an evaluation metric for binary classification problems. It is a curve that plots the TPR against FPR at different threshold values and separates the noise from the signal. The Area Under the Curve (AUC) measures how well a classifier distinguishes between classes and is used as a summary of the ROC curve. The higher the AUC, the better the performance for the model at distinguishing between the positive and negative classes.

The data-set contains significant and non-significant images from the original data-set that have not been trained on classifier before, and all the slices of a particular patient are present in a single data-set and are not repeated in another data-set.

6.6.1 Model Performance for Different Percentage of cGAN Generated Images

In this section, model performance is compared between different percentages of generated data added in the original data set. The original data-set is increased by different percentages of 50%\100% \150% along with another case dubbed as balance data-set. In the balance data-set, the class imbalance is removed by increasing the data in both significant and non-significant tumor data sets until the number of images in both classes is equal. Increasing the data-set by 50% means that the original data-set containing 1285 images will be augmented by 643 additional images. The same logic is carried out for 100% and 150% increases in data to study the impact of increasing the data. The data sets are increased by percentage according to the number of images in a specific class. After the data sets are prepared according to the configurations mentioned above, they train the VGG16 network. The network is trained on the data-sets with EarlyStopping with validation AUC under observation with the patience of 20 epochs. If the *val_AUC* does not change for 20 epochs, the network stops training and then calculates the AUC-ROC curves using the best model saved during the training using checkpoint callback function. The same data set is run three times, and then the average is calculated to present results.

Data-set	**macro ROC-AUC**
Original Data-set	0.55
Balanced Data-set	0.71
150% Data-set	0.58
100% Data-set	0.61
50% Data-set	0.59

Table 6.3: Comparison of ROC-AUC of generated and original data-sets

From the table 6.3, it can be seen that the best AUC-ROC is provided by the balance data-set. It has the highest AUC compared to the original data-set and shows an overall improvement of 18% from the original one. It can also be observed by analyzing the figure 6.6 that the difference between the macro ROC-AUC for both 100% and 50% is close to each other, implying that adding less data can help increase the model performance significantly without increasing the computational time. It can also be observed that AUC improvement for 150% is less than compared to all other augmented data sets, which implies that increasing the data-set beyond a specific value can have negative impacts while also increasing the computation time.

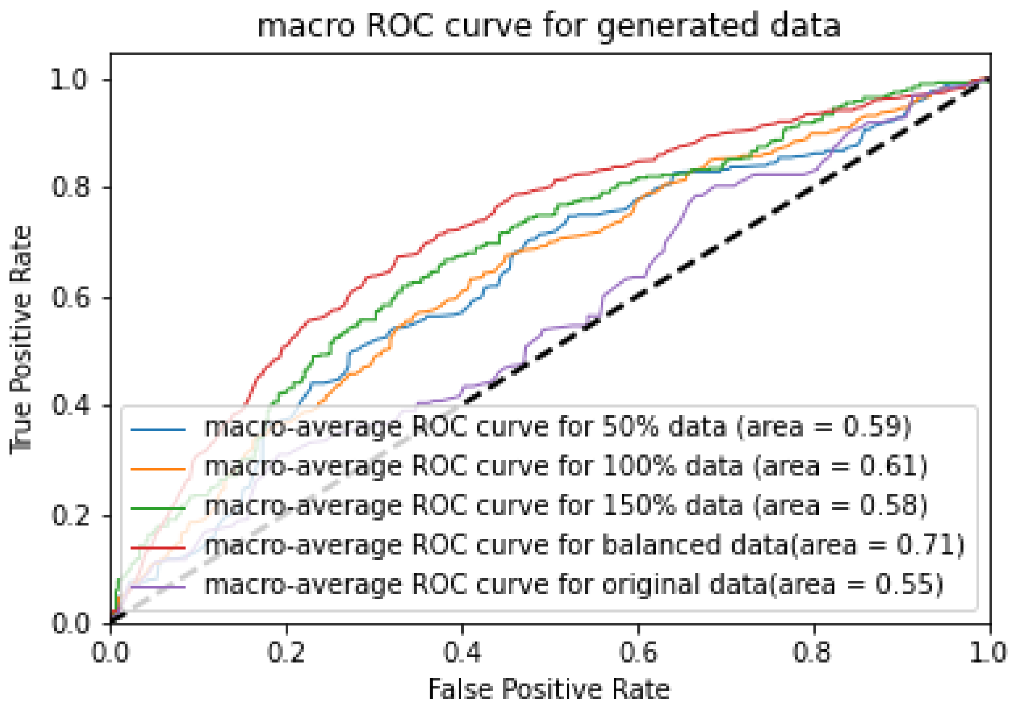


Figure 6.5: Illustrated AUC-ROC curves for all generated data-sets

6.6.2 Model performance for Augmented versus Generated Images

In this section, model performance is compared between generated and conventional augmentation methods. The data sets that will be compared are prepared by using the methods mentioned in the section 6.5. All of the data-set is increased by 100%. Conventional augmentation methods results are mentioned in the table 6.4. It can be observed that all three augmentation methods have less AUC when compared with the generative methods. Image translation has the worse AUC and does not contribute any improvement in the classification. All the generative methods have better AUC than any of the augmentation methods. In augmentation methods, image flipping performs better than translation and rotation. The AUC for image flipping is approximately the same as was obtained using 150% data generation using GAN. Balanced data-set has performed better than any other generative or conventional augmentation method with an AUC of 0.71, which shows an improvement of 18 percent from the original data-set, which has an AUC of 0.55. The AUC mentioned in table 6.4 are presented after running the model 3 times and taking their average.

Data-set	**macro ROC-AUC**
Original Data-set	0.55
Balanced Data-set	0.71
100% Data-set	0.61
Rotated Data-set	0.55
Translated Data-set	0.53
Flipped	0.56

Table 6.4: Comparison of ROC-AUC of generated and augmented data-sets

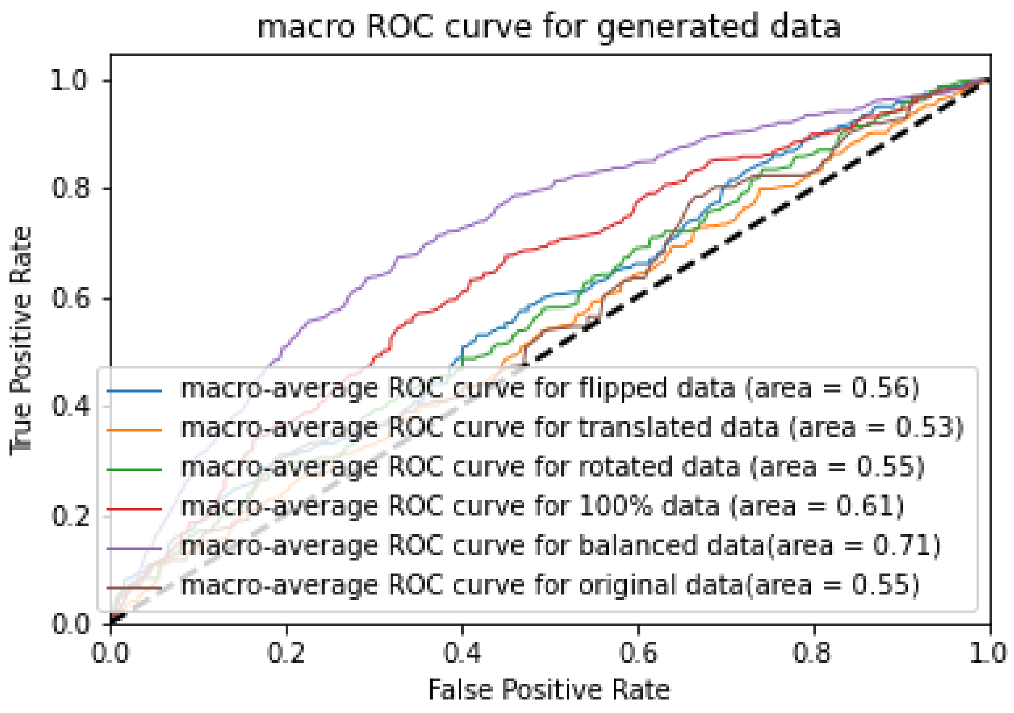


Figure 6.6: Illustrated AUC-ROC curves for all augmented and two generative data-sets

Chapter 7

Discussion

This chapter presents the discussion of the achieved results, different factors that affect the model's performance, evaluation of the proposed methods, and limitations of the presented methods.

7.1 Effectiveness of Proposed Methodology

The analysis of achieved results explained in chapter 6 shows that the proposed method successfully proves that data augmentation using the generative method is more effective than the standard augmentation methods. The average macro ROC-AUC of 0.71 for the proposed model provides evidence that the model can provide better augmentation for the training of classification models for improving the binary or categorical classification. The results produced in this thesis presents comparative studies between generative and augmentation methods.

7.2 Evaluation of Generative Models

In this thesis, two generative models were discussed and tested, cGAN and DCGAN. In order to generate images, cGAN was selected due to multiple reasons. cGAN has particular advantages over DCGAN, given that we can provide the additional information required for the cGAN. The images generated by DCGAN are random, and it is challenging to obtain the required images; it takes more time for the DCGAN to converge than the cGAN to train on different data sets to obtain the required images. These problems are solved using cGAN, where we can provide the labels to the cGAN and get the required images from it. cGAN can be trained on more than two classes and generates images when provided a specific label—this helps to train at the whole data without struggling to get the required images from the model. cGAN saves time for the training and gives more control over the images generated from the data. Since cGAN trains on more data than DCGAN, the image quality is better for it.

The results of training a binary classification model on the combination of original and synthetic data are promising because the performance is higher when the combined data is used than only original data. The generated data is not similar to augmentation data as all the augmentation performance is lower than flip.

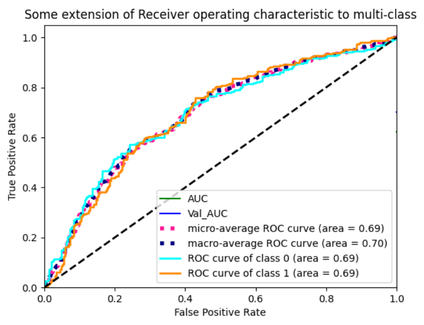
7.3 Evaluation of the Augmented Methods

Data augmentation for deep neural networks is a method to extend the data-set artificially. Data augmentation methods re-arrange the pixels enough for the model to not recognize it as the original and deems it separate from the original MRI. Augmentation is counter-productive when it is too different from the original data, making it necessary to analyze the training data to find the best augmentation techniques to preserve the critical parameters of the MRI images. These three data augmentation types were used, that are rotation, horizontal flip, and translation.

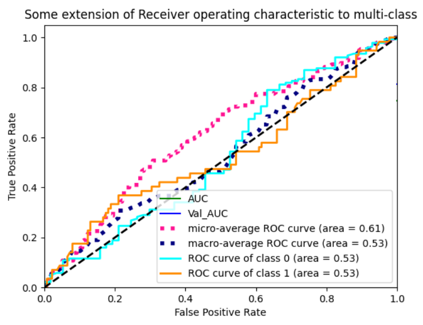
When the augmented data sets are used for the VGG16 network, it produces slightly better results than the original data set, and the reason is that these images do not have the same structure as the original data-set MRI images, hence the network does not recognize the same images.

7.4 Macro ROC-AUC

In this thesis, Macro ROC-AUC is considered and discussed because the data is highly imbalanced. Number of non-significant patients is lot higher than significant patients which creates a class imbalance in the data-set. VGG16 is a categorical classifier that means it can classify more than 2 classes, due to which categorical ROC-AUC is used. The categorical ROC-AUC generates individual class ROC-AUC curves and then has two curves macro and micro. Macro ROC-AUC is the average of class AUC and in our case represents the actual ROC-AUC case. Micro ROC-AUC is weighted average of class ROC-AUC which means that if the classifier identifies one class better than the other one then micro ROC-AUC will show high value since most of the majority class is correctly predicted and will show higher results which do not represent the actual case. The data used in this thesis, is highly imbalanced and micro ROC-AUC is always higher because the majority class (non-significant tumors) which in this case is approx. 80% of total original data-set will be predicted correctly due to this reason micro ROC-AUC does not constitute a good metric. More individual ROC-AUC curves are available in the appendix A to confirm this.



Balanced Data



Real Data

Figure 7.1: ROC-AUC for balanced (left) and Original (right) data-sets

From the figure 7.1 it can be observed that the micro ROC-AUC of the real data is lot higher than the macro ROC-AUC because the insignificant slices far exceed the significant slices in the original data thus causing this imbalance. When the data is augmented using GAN and both the classes are balanced, then both the micro ROC-AUC and macro ROC-AUC becomes significantly close to each other. So, the model was evaluated on macro to avoid the biases on which micro ROC-AUC is calculated.

7.5 Evaluation of Performance

The thesis's primary focus has been to develop a method to extend the data-set with synthetic data using GAN architecture. In this thesis, only cGAN generated images are used to generate the data, which proved to be more beneficial for improving the binary classification. Since various percentages were used to analyze the impact to check what percentage of data can be used to save the computation time for the training of classification network and provide higher results compared to conventional augmentations, the binary classification network VGG16 has the same settings as the original data, augmented data, and synthetic data combined with the original data and should be suitable for comparing the results.

7.6 Limitations

There are various limitations of this project that will be discussed be in this section.

7.6.1 Evaluation of the GAN metrics

There is a problem in GAN metrics that there is no definite way to evaluate the GAN architectures. Various metrics were observed to evaluate the results. The metrics that were considered were Structure similarity index measure (SSIM), Frechet Inception Distance (FID), mean squared error (MSE), and other image quality measures, but the problem with this specific data-set is that each patient MRI images differ from each other. So, when these measures are applied to the images from this data set. Two images from the same data-set differ from each other to such an extent that they look different from each other, even on a visual basis. For example, when MSE is used on two images, it might show that these images are similar to each other, but when seen by an expert, they might be horrible images because MSE does not account if the image is blurry or distorted.

7.6.2 Data

The model was able to produce acceptable results for the data generation part. The data points used in the original paper[21] were around 3 million images, but since we are dealing with medical image data which is hard to obtain due to patient secrecy. However, the data available from PROSTATEx is minimal, and when the data is organized to use for training the model, the data-set becomes even more scarce and has less than 1300 images, including both significant and non-significant tumors. Due to this, the results obtained from cGAN are considered very good that even with limited data, it produced acceptable results.

7.6.3 Computational Limitations

The training of deep learning models like DCGAN and cGAN is a time-consuming and computationally expensive task. The training of the neural networks requires high-powered GPU servers to perform the demanding computations. The University of Stavanger has servers that used to have excellent GPU servers, but it has long queues to access the servers, which can cause some delays in the training process.

It took one week to train cGAN and DCGAN using the UiS servers so, if the server is refreshed or some other user runs, any process on the server will overload, and the training process will stop. Even if the server is powerful, it can still not perform specific actions, which causes the memory to overload, thus limiting the parameters that can be used to enhance the results.

Chapter 8

Conclusion and Recommendations

8.1 Conclusion

The thesis explores the potential of extending the data-set with synthetic MRIs using two different GAN architectures, namely cGAN and DCGAN, to obtain good quality data to train a reliable binary classification network. This method uses cGAN architecture to generate two classes of data, and then the data is used with a combination of original data to train the VGG16 classification network to check the ROC-AUC of the network on various configurations.

This thesis uses a two-step methodology; the first step is pre-processing the data and using that data to train the GAN to generate more synthetic data and data augmentation techniques. The second step involves using the original data and augmented data to train the classification network and evaluate the results of the different data augmentation techniques. The thesis directly compares the ROC-AUC score that is generated from the proposed classification model.

A set of experiments was performed to find optimal hyper-parameters, augmentation techniques, and neural network structure to generate the best possible data and classify the data correctly. The results produced by this thesis suggest that generated data is better than the augmented data to improve the classification of the binary classifiers.

The main challenge in this thesis was the training of the GAN architectures because it is sensitive to hyper-parameters and limited data available to train and adjust five different models.

The conclusion is that the obtained results support the primary aim of this thesis: that the proposed methods can be employed to generate new images and improve the classifier's performance by using Generative Adversarial Networks.

8.2 Future Recommendations

This section is dedicated to personal recommendations from the writer to future members of the project.

- **Evaluation of GAN:** There is an ongoing problem with the GAN architectures that there is no one metric that everyone agrees to check the image quality or determine the metric that can evaluate the GAN. Work can be done to identify which metrics can be used to evaluate the GAN architecture.
- **Different GAN architectures:** In this thesis, two GAN architectures were tested for image generation. More GAN architectures can be used to generate better quality images to augment the data site.
- **Classifiers:** VGG16 was the preferred classifier to be used in this thesis, other classifiers such as resnet, vgg19, or exceptions can be explored to apply the same methodology on them.
- **Data:** Available data was limited in this work, so, to further improve the image quality, more data can be gathered to improve the GAN image quality further.

References

[1] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. 2016. arXiv: 1603.04467 [cs.DC].

[2] Aqeel Anwar. *What is Transposed Convolutional Layer?* en. Apr. 2021. URL: https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11 (visited on 07/15/2021).

[3] *Artificial neural network*. en. Page Version ID: 1032050958. July 2021. URL: https://en.wikipedia.org/w/index.php?title=Artificial_neural_network&oldid=1032050958 (visited on 07/05/2021).

[4] "Ascent of machine learning in medicine". en. In: *Nature Materials* 18.5 (May 2019). Bandiera_abtest: a Cg_type: Nature Research Journals Number: 5 Primary_atype: Editorial Publisher: Nature Publishing Group Subject_term: Biomedical engineering;Computational biology and bioinformatics;Machine learning Subject_term_id: biomedical-engineering;computational-biology-and-bioinformatics;machine-learning, pp. 407–407. ISSN: 1476-4660. DOI: 10.1038/s41563-019-0360-1. URL: https://www.nature.com/articles/s41563-019-0360-1 (visited on 06/26/2021).

[5] *Backpropagation — Brilliant Math & Science Wiki*. en-us. URL: https://brilliant.org/wiki/backpropagation/ (visited on 07/05/2021).

[6] Jason Brownlee. "Generative Adversarial Networks with Python". en. In: (), p. 654.

[7] Jason Brownlee. "Generative Adversarial Networks with Python". en. In: (), p. 654.

[8] Jason Brownlee. *How to Explore the GAN Latent Space When Generating Faces*. en-US. July 2019. URL: https://machinelearningmastery.com/how-to-interpolate-and-perform-vector-arithmetic-with-faces-using-a-generative-adversarial-network/ (visited on 06/08/2021).

[9] *Can a biopsy spread prostate cancer? — PCFA*. URL: https://www.prostate.org.au/news-media/news-archive/news-archive-2018/can-a-biopsy-spread-prostate-cancer/ (visited on 07/14/2021).

[10] *Cancer today*. en. URL: http://gco.iarc.fr/today/home (visited on 06/26/2021).

[11] Elizabeth M Charles-Edwards and Nandita M deSouza. “Diffusion-weighted magnetic resonance imaging and its application to cancer”. In: *Cancer Imaging* 6.1 (Sept. 2006), pp. 135–143. ISSN: 1740-5025. DOI: 10.1102/1470-7330.2006.0021. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1693785/ (visited on 07/15/2021).

[12] Francois Chollet et al. *How to manually scale image pixel data for deep learning,2019.* 2015. URL: https://machinelearningmastery.com/how-to-manually-scaleimage-pixel-data-for-deep-learning/..

[13] Francois Chollet et al. *Keras.* 2015. URL: https://github.com/fchollet/keras.

[14] MaryBeth B. Culp et al. “Recent Global Patterns in Prostate Cancer Incidence and Mortality Rates”. eng. In: *European Urology* 77.1 (Jan. 2020), pp. 38–52. ISSN: 1873-7560. DOI: 10.1016/j.eururo.2019.08.005.

[15] MaryBeth B. Culp et al. “Recent Global Patterns in Prostate Cancer Incidence and Mortality Rates”. eng. In: *European Urology* 77.1 (Jan. 2020), pp. 38–52. ISSN: 1873-7560. DOI: 10.1016/j.eururo.2019.08.005.

[16] *Data Science Bowl 2017.* en. URL: https://kaggle.com/c/data-science-bowl-2017 (visited on 07/15/2021).

[17] Alvaro Fernandez-Quilez et al. “Improving Prostate Whole Gland Segmentation In T2-Weighted MRI With Synthetically Generated Data”. In: *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI).* 2021, pp. 1915–1919. DOI: 10.1109/ISBI48211.2021.9433793.

[18] Smith K Freymann et al. “SThe cancer imaging archive (tcia): Maintaining and operating a public information repository”. In: *Journal of Digital Imaging* 26 (2013).

[19] Maayan Frid-Adar et al. “Synthetic data augmentation using GAN for improved liver lesion classification”. In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018).* ISSN: 1945-8452. Apr. 2018, pp. 289–293. DOI: 10.1109/ISBI.2018.8363576.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* http://www.deeplearningbook.org. MIT Press, 2016.

[21] Ian Goodfellow et al. “Generative adversarial networks”. In: *Communications of the ACM* 63.11 (Oct. 2020), pp. 139–144. ISSN: 0001-0782. DOI: 10.1145/3422622. URL: https://doi.org/10.1145/3422622 (visited on 07/15/2021).

[22] *Key Statistics for Prostate Cancer — Prostate Cancer Facts.* en. URL: https://www.cancer.org/cancer/prostate-cancer/about/key-statistics.html (visited on 06/26/2021).

[23] Steinar Valle Larsen. *Exploring Generative Adversarial Networks to Improve Prostate Segmentation on MRI.* University of Stavanger, 2020.

[24] Geert Litjens et al. *ProstateX Challenge data.* 2017. URL: https://wiki.cancerimagingarchive.net/display/Public/SPIE-AAPM-NCI+PROSTATEx+Challenges.

[25] *Magnetic Resonance Imaging (MRI).* URL: https://www.nibib.nih.gov/science-education/science-topics/magnetic-resonance-imaging-mri (visited on 06/28/2021).

[26] Darcy Mason. "SU-E-T-33: pydicom: an open source DICOM library". In: *Medical Physics* 38.6Part10 (2011), pp. 3493–3493.

[27] Mehdi Mirza and Simon Osindero. "Conditional Generative Adversarial Nets". en. In: *arXiv:1411.1784 [cs, stat]* (Nov. 2014). arXiv: 1411.1784. URL: http://arxiv.org/abs/1411.1784 (visited on 07/03/2021).

[28] Saman Motamed, Patrik Rogalla, and Farzad Khalvati. *Data Augmentation using Generative Adversarial Networks (GANs) for GAN-based Detection of Pneumonia and COVID-19 in Chest X-ray Images.* 2021. arXiv: 2006.03622 [cs.CV].

[29] *MRI Basics.* URL: https://case.edu/med/neurology/NR/MRI%20Basics.htm (visited on 07/15/2021).

[30] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[31] *Prostate biopsy.* en. URL: https://prostatecanceruk.org/prostate-information/prostate-tests/prostate-biopsy/ (visited on 07/15/2021).

[32] *PSA test - Mayo Clinic.* URL: https://www.mayoclinic.org/tests-procedures/psa-test/about/pac-20384731 (visited on 07/15/2021).

[33] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". en. In: *arXiv:1511.06434 [cs]* (Jan. 2016). arXiv: 1511.06434. URL: http://arxiv.org/abs/1511.06434 (visited on 04/26/2021).

[34] *Rectal examination.* en. Page Version ID: 1030217201. June 2021. URL: https://en.wikipedia.org/w/index.php?title=Rectal_examination&oldid=1030217201 (visited on 07/14/2021).

[35] Palash Sharma. *Keras Dense Layer Explained for Beginners — MLK - Machine Learning Knowledge.* en-US. URL: https://machinelearningknowledge.ai/keras-dense-layer-explained-for-beginners/ (visited on 07/15/2021).

[36] Hoo-Chang Shin et al. *Medical Image Synthesis for Data Augmentation and Anonymization using Generative Adversarial Networks.* 2018. arXiv: 1807.10225 [cs.CV].

[37] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv:1409.1556 [cs]* (Apr. 2015). arXiv: 1409.1556. URL: http://arxiv.org/abs/1409.1556 (visited on 07/14/2021).

[38] *VGG16 - Convolutional Network for Classification and Detection.* en-US. Nov. 2018. URL: `https://neurohive.io/en/popular-networks/vgg16/` (visited on 07/03/2021).

[39] *What is a Conditional GAN (cGAN)?* en. URL: `https://www.educative.io/edpresso/what-is-a-conditional-gan-cgan` (visited on 07/15/2021).

[40] *What is Cancer?* en. Aug. 2012. URL: `https://www.cancer.net/navigating-cancer-care/cancer-basics/what-cancer` (visited on 07/15/2021).

[41] David A. Woodrum et al. "Targeted prostate biopsy and MR-guided therapy for prostate cancer". en. In: *Abdominal Radiology* 41.5 (May 2016), pp. 877–888. ISSN: 2366-004X, 2366-0058. DOI: `10.1007/s00261-016-0681-3`. URL: `http://link.springer.com/10.1007/s00261-016-0681-3` (visited on 06/26/2021).

[42] Guangyuan Zhang et al. "A Method for the Estimation of Finely-Grained Temporal Spatial Human Population Density Distributions Based on Cell Phone Call Detail Records". In: *Remote Sensing* 12 (Aug. 2020), p. 2572. DOI: `10.3390/rs12162572`.

Appendix A

ROC-AUC curves

Following ROC-AUC curves are calculated for after the model trained and tested 3 times.

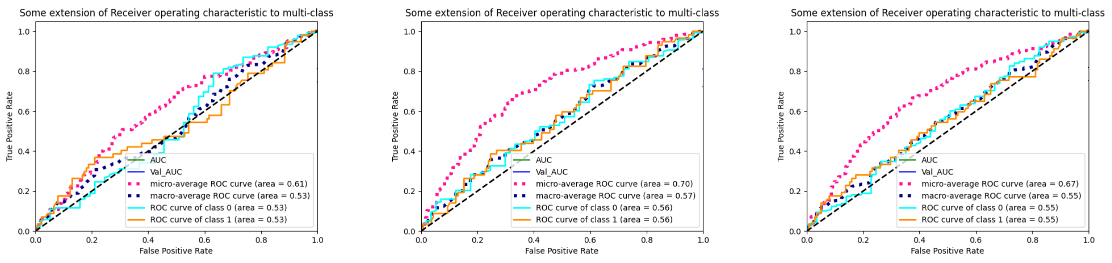


Figure A.1: Real Data

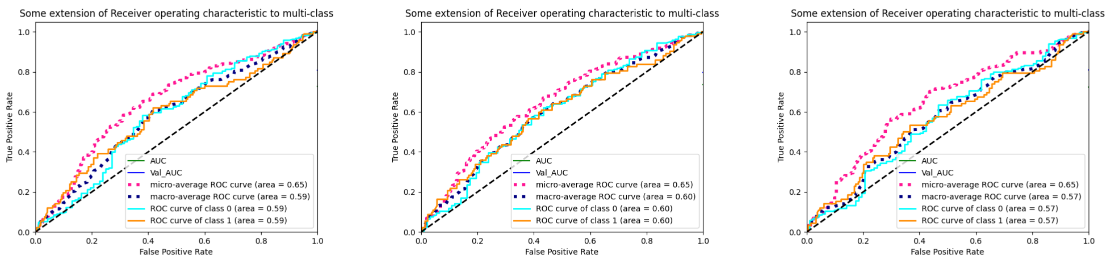


Figure A.2: 50% augmented data in the original data

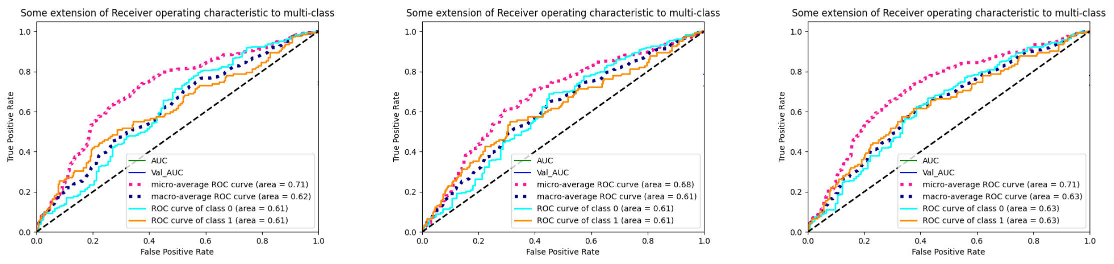


Figure A.3: 100% augmented data in the original data

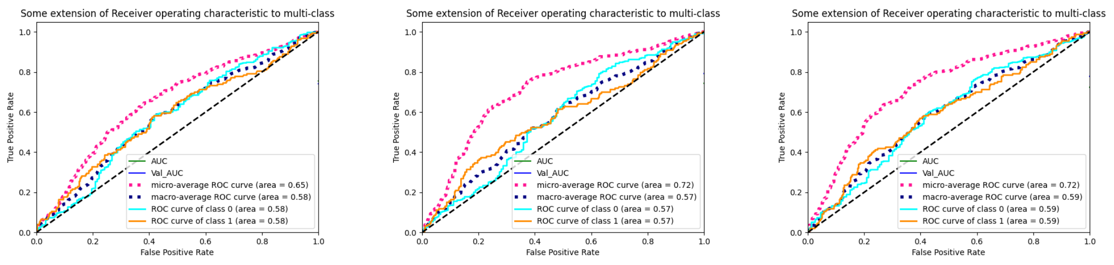


Figure A.4: 150% augmented data in the original data

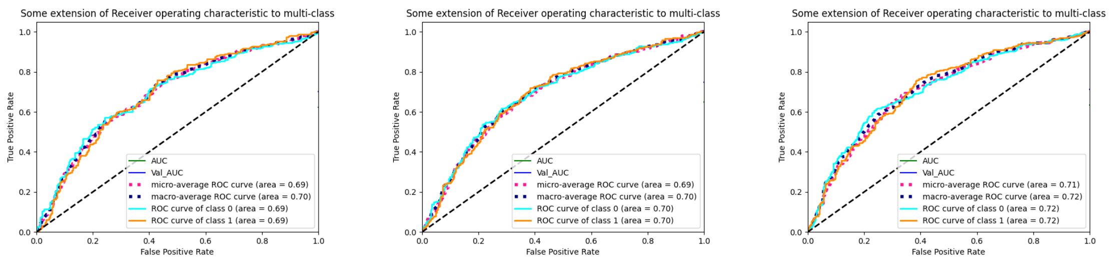


Figure A.5: Balanced data in the original data

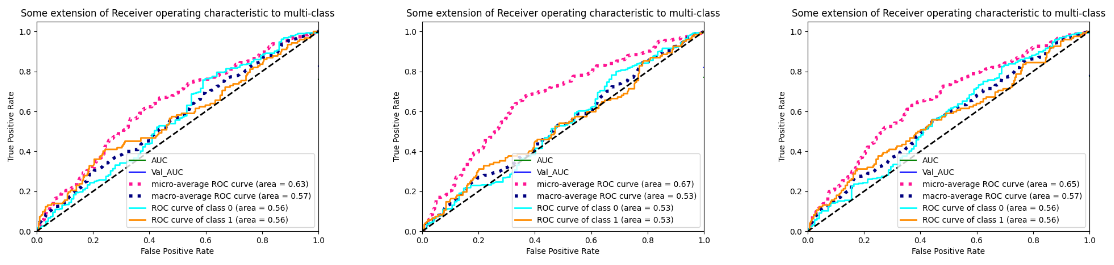


Figure A.6: Rotated augmented data in the original data

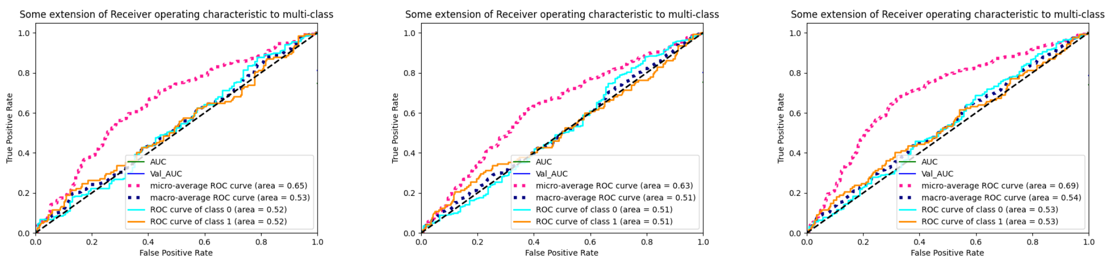


Figure A.7: Translated augmented data in the original data

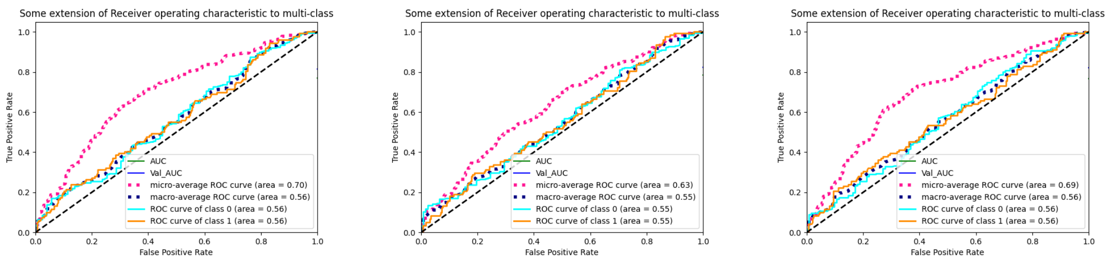


Figure A.8: Horizontal flipped augmented data in the original data

Appendix B

Architectures of the model used

In this appendix, model architecture along with their shapes are shared.

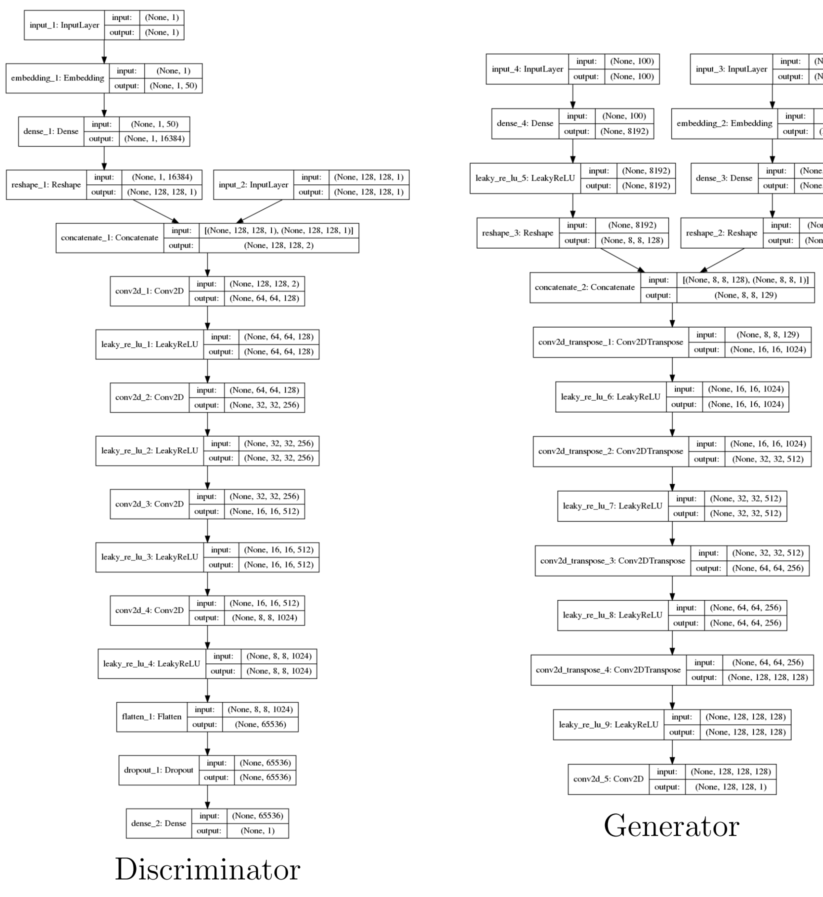

input_1: InputLayer | input: (None, 1) | output: (None, 1)
embedding_1: Embedding | input: (None, 1) | output: (None, 1, 50)
dense_1: Dense | input: (None, 1, 50) | output: (None, 1, 16384)
reshape_1: Reshape | input: (None, 1, 16384) | output: (None, 128, 128, 1)
input_2: InputLayer | input: (None, 128, 128, 1) | output: (None, 128, 128, 1)
concatenate_1: Concatenate | input: [(None, 128, 128, 1), (None, 128, 128, 1)] | output: (None, 128, 128, 2)
conv2d_1: Conv2D | input: (None, 128, 128, 2) | output: (None, 64, 64, 128)
leaky_re_lu_1: LeakyReLU | input: (None, 64, 64, 128) | output: (None, 64, 64, 128)
conv2d_2: Conv2D | input: (None, 64, 64, 128) | output: (None, 32, 32, 256)
leaky_re_lu_2: LeakyReLU | input: (None, 32, 32, 256) | output: (None, 32, 32, 256)
conv2d_3: Conv2D | input: (None, 32, 32, 256) | output: (None, 16, 16, 512)
leaky_re_lu_3: LeakyReLU | input: (None, 16, 16, 512) | output: (None, 16, 16, 512)
conv2d_4: Conv2D | input: (None, 16, 16, 512) | output: (None, 8, 8, 1024)
leaky_re_lu_4: LeakyReLU | input: (None, 8, 8, 1024) | output: (None, 8, 8, 1024)
flatten_1: Flatten | input: (None, 8, 8, 1024) | output: (None, 65536)
dropout_1: Dropout | input: (None, 65536) | output: (None, 65536)
dense_2: Dense | input: (None, 65536) | output: (None, 1)
Discriminator
input_4: InputLayer | input: (None, 100) | output: (None, 100)
input_3: InputLayer | input: (None, 1) | output: (None, 1)
dense_4: Dense | input: (None, 100) | output: (None, 8192)
embedding_2: Embedding | input: (None, 1) | output: (None, 1, 50)
leaky_re_lu_5: LeakyReLU | input: (None, 8192) | output: (None, 8192)
dense_3: Dense | input: (None, 1, 50) | output: (None, 1, 64)
reshape_3: Reshape | input: (None, 8192) | output: (None, 8, 8, 128)
reshape_2: Reshape | input: (None, 1, 64) | output: (None, 8, 8, 1)
concatenate_2: Concatenate | input: [(None, 8, 8, 128), (None, 8, 8, 1)] | output: (None, 8, 8, 129)
conv2d_transpose_1: Conv2DTranspose | input: (None, 8, 8, 129) | output: (None, 16, 16, 1024)
leaky_re_lu_6: LeakyReLU | input: (None, 16, 16, 1024) | output: (None, 16, 16, 1024)
conv2d_transpose_2: Conv2DTranspose | input: (None, 16, 16, 1024) | output: (None, 32, 32, 512)
leaky_re_lu_7: LeakyReLU | input: (None, 32, 32, 512) | output: (None, 32, 32, 512)
conv2d_transpose_3: Conv2DTranspose | input: (None, 32, 32, 512) | output: (None, 64, 64, 256)
leaky_re_lu_8: LeakyReLU | input: (None, 64, 64, 256) | output: (None, 64, 64, 256)
conv2d_transpose_4: Conv2DTranspose | input: (None, 64, 64, 256) | output: (None, 128, 128, 128)
leaky_re_lu_9: LeakyReLU | input: (None, 128, 128, 128) | output: (None, 128, 128, 128)
conv2d_5: Conv2D | input: (None, 128, 128, 128) | output: (None, 128, 128, 1)
Generator

Figure B.1: Cgan Models

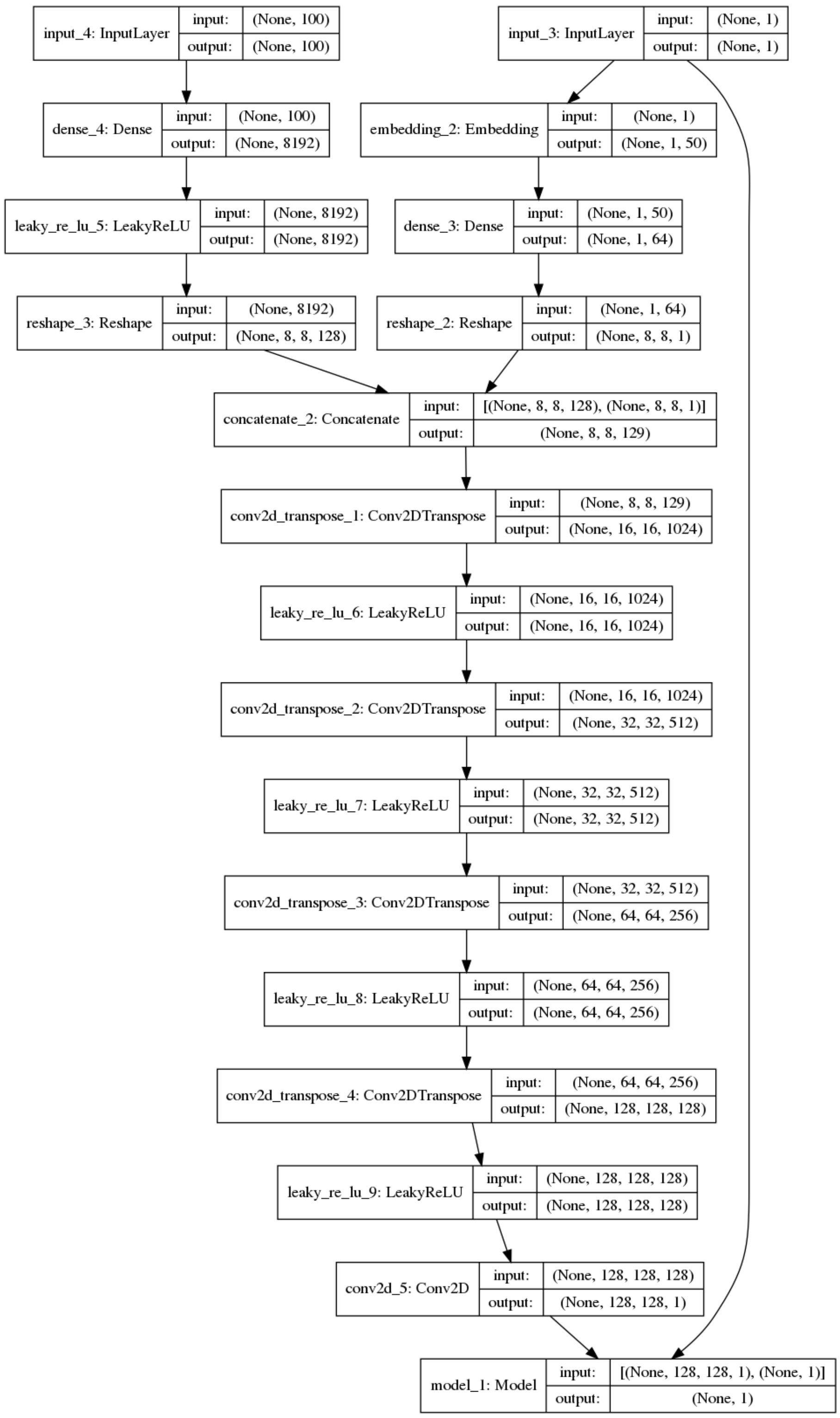
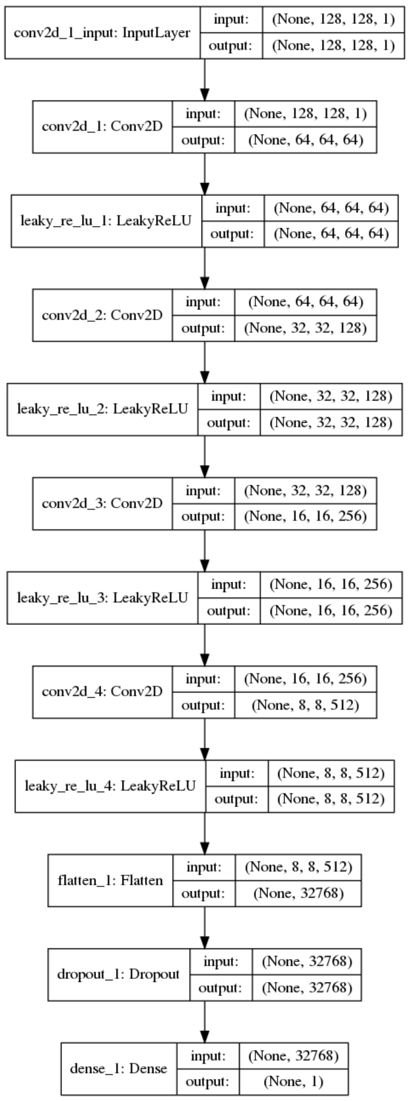
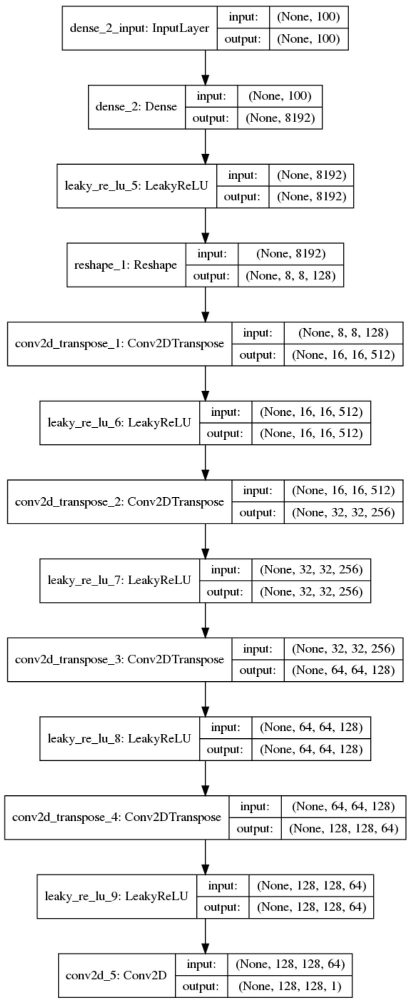

input_4: InputLayer | input: (None, 100) | output: (None, 100)
input_3: InputLayer | input: (None, 1) | output: (None, 1)
dense_4: Dense | input: (None, 100) | output: (None, 8192)
embedding_2: Embedding | input: (None, 1) | output: (None, 1, 50)
leaky_re_lu_5: LeakyReLU | input: (None, 8192) | output: (None, 8192)
dense_3: Dense | input: (None, 1, 50) | output: (None, 1, 64)
reshape_3: Reshape | input: (None, 8192) | output: (None, 8, 8, 128)
reshape_2: Reshape | input: (None, 1, 64) | output: (None, 8, 8, 1)
concatenate_2: Concatenate | input: [(None, 8, 8, 128), (None, 8, 8, 1)] | output: (None, 8, 8, 129)
conv2d_transpose_1: Conv2DTranspose | input: (None, 8, 8, 129) | output: (None, 16, 16, 1024)
leaky_re_lu_6: LeakyReLU | input: (None, 16, 16, 1024) | output: (None, 16, 16, 1024)
conv2d_transpose_2: Conv2DTranspose | input: (None, 16, 16, 1024) | output: (None, 32, 32, 512)
leaky_re_lu_7: LeakyReLU | input: (None, 32, 32, 512) | output: (None, 32, 32, 512)
conv2d_transpose_3: Conv2DTranspose | input: (None, 32, 32, 512) | output: (None, 64, 64, 256)
leaky_re_lu_8: LeakyReLU | input: (None, 64, 64, 256) | output: (None, 64, 64, 256)
conv2d_transpose_4: Conv2DTranspose | input: (None, 64, 64, 256) | output: (None, 128, 128, 128)
leaky_re_lu_9: LeakyReLU | input: (None, 128, 128, 128) | output: (None, 128, 128, 128)
conv2d_5: Conv2D | input: (None, 128, 128, 128) | output: (None, 128, 128, 1)
model_1: Model | input: [(None, 128, 128, 1), (None, 1)] | output: (None, 1)

Figure B.2: cGAN with shapes


conv2d_1_input: InputLayer | input: (None, 128, 128, 1) | output: (None, 128, 128, 1)
conv2d_1: Conv2D | input: (None, 128, 128, 1) | output: (None, 64, 64, 64)
leaky_re_lu_1: LeakyReLU | input: (None, 64, 64, 64) | output: (None, 64, 64, 64)
conv2d_2: Conv2D | input: (None, 64, 64, 64) | output: (None, 32, 32, 128)
leaky_re_lu_2: LeakyReLU | input: (None, 32, 32, 128) | output: (None, 32, 32, 128)
conv2d_3: Conv2D | input: (None, 32, 32, 128) | output: (None, 16, 16, 256)
leaky_re_lu_3: LeakyReLU | input: (None, 16, 16, 256) | output: (None, 16, 16, 256)
conv2d_4: Conv2D | input: (None, 16, 16, 256) | output: (None, 8, 8, 512)
leaky_re_lu_4: LeakyReLU | input: (None, 8, 8, 512) | output: (None, 8, 8, 512)
flatten_1: Flatten | input: (None, 8, 8, 512) | output: (None, 32768)
dropout_1: Dropout | input: (None, 32768) | output: (None, 32768)
dense_1: Dense | input: (None, 32768) | output: (None, 1)

Discriminator


dense_2_input: InputLayer | input: (None, 100) | output: (None, 100)
dense_2: Dense | input: (None, 100) | output: (None, 8192)
leaky_re_lu_5: LeakyReLU | input: (None, 8192) | output: (None, 8192)
reshape_1: Reshape | input: (None, 8192) | output: (None, 8, 8, 128)
conv2d_transpose_1: Conv2DTranspose | input: (None, 8, 8, 128) | output: (None, 16, 16, 512)
leaky_re_lu_6: LeakyReLU | input: (None, 16, 16, 512) | output: (None, 16, 16, 512)
conv2d_transpose_2: Conv2DTranspose | input: (None, 16, 16, 512) | output: (None, 32, 32, 256)
leaky_re_lu_7: LeakyReLU | input: (None, 32, 32, 256) | output: (None, 32, 32, 256)
conv2d_transpose_3: Conv2DTranspose | input: (None, 32, 32, 256) | output: (None, 64, 64, 128)
leaky_re_lu_8: LeakyReLU | input: (None, 64, 64, 128) | output: (None, 64, 64, 128)
conv2d_transpose_4: Conv2DTranspose | input: (None, 64, 64, 128) | output: (None, 128, 128, 64)
leaky_re_lu_9: LeakyReLU | input: (None, 128, 128, 64) | output: (None, 128, 128, 64)
conv2d_5: Conv2D | input: (None, 128, 128, 64) | output: (None, 128, 128, 1)

Generator

Figure B.3: DCGAN Models

Appendix C

GAN Training Images

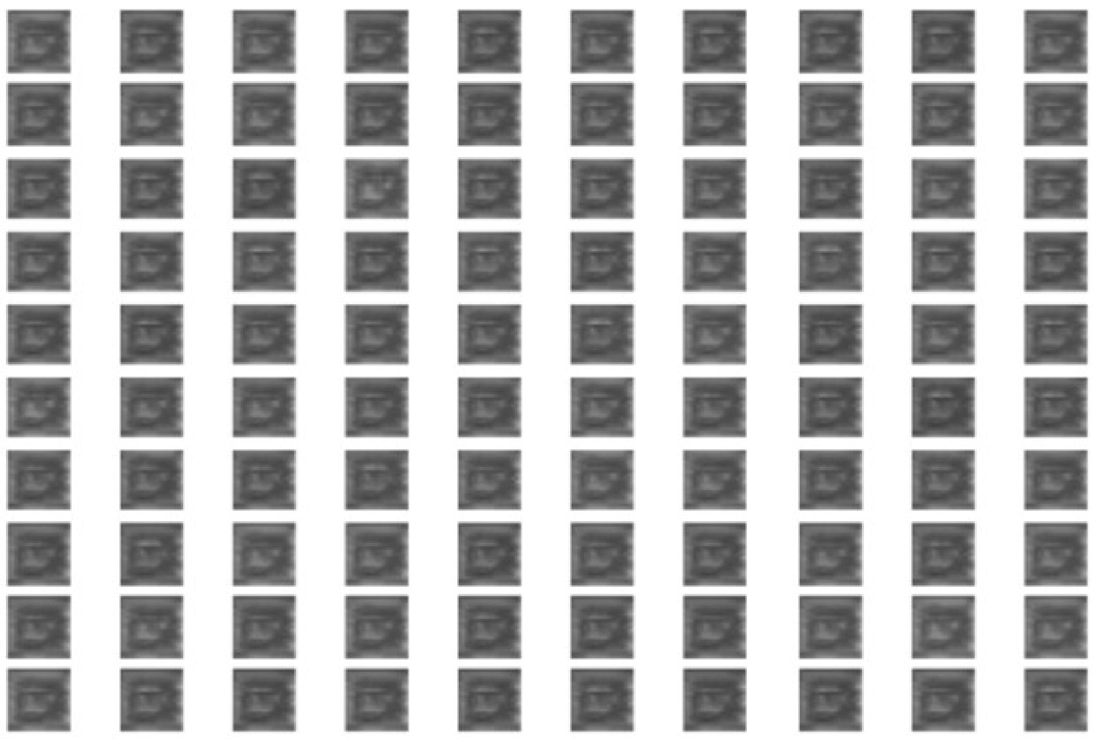


Figure C.1: DC GAN 1000 epochs

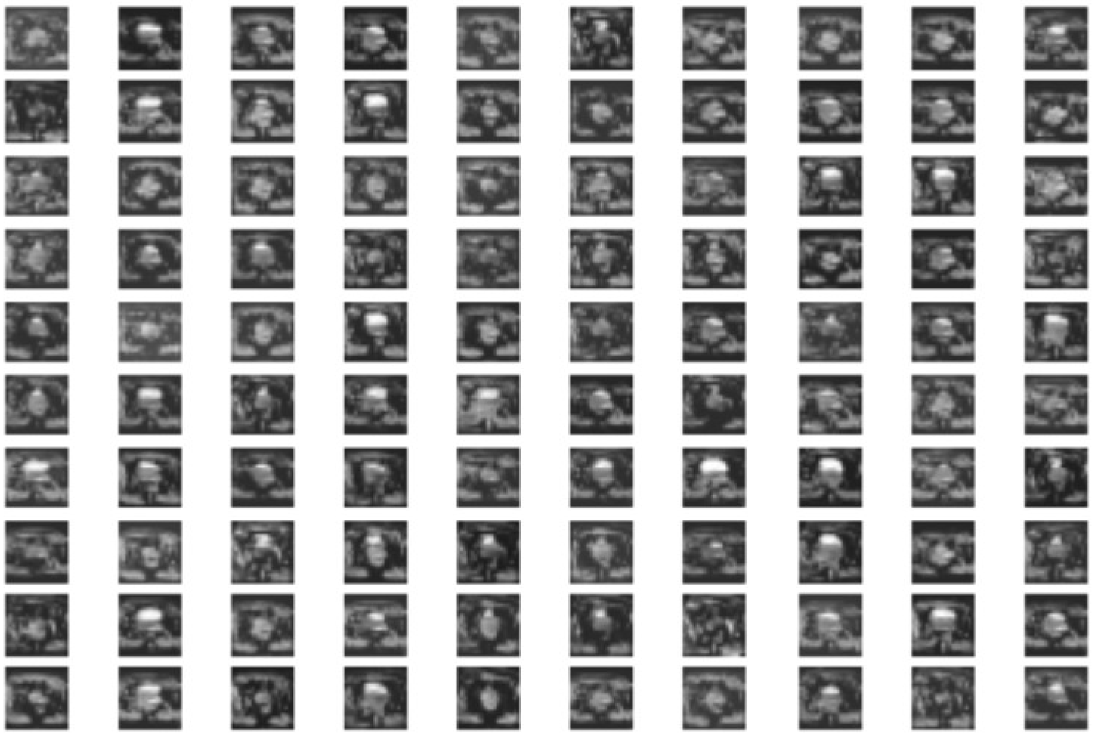


Figure C.2: DC GAN 2000 epochs

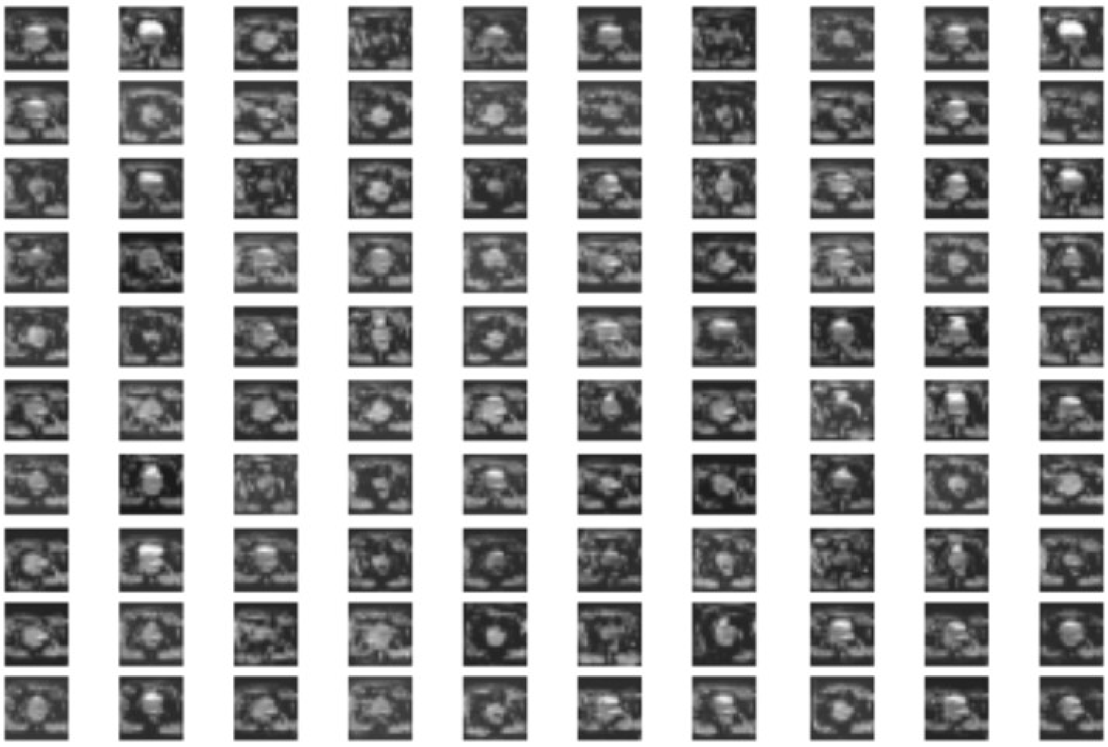


Figure C.3: DC GAN 3000 epochs

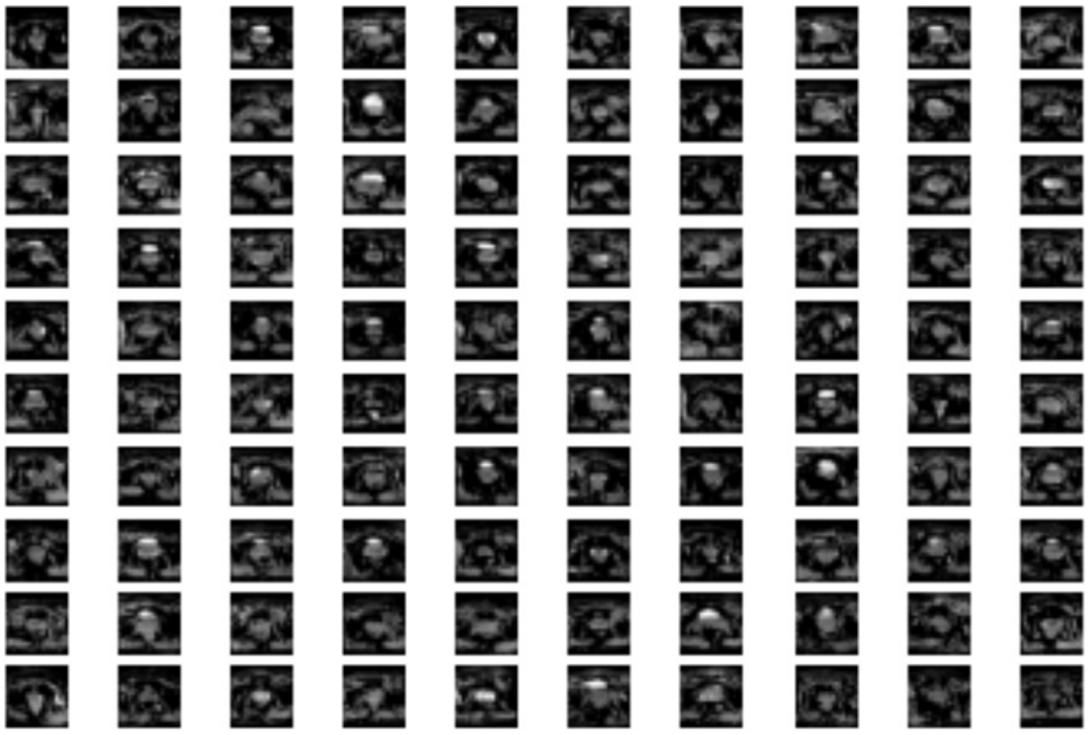


Figure C.4: cGAN 1000 epochs

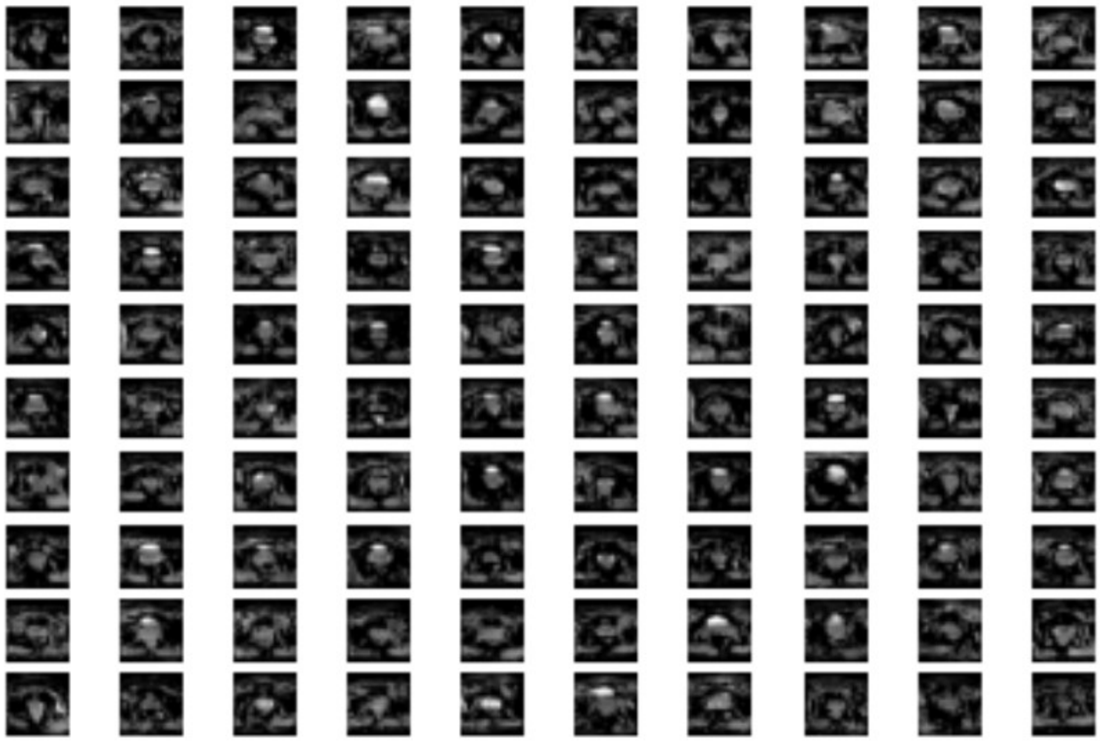


Figure C.5: cGAN 2000 epochs

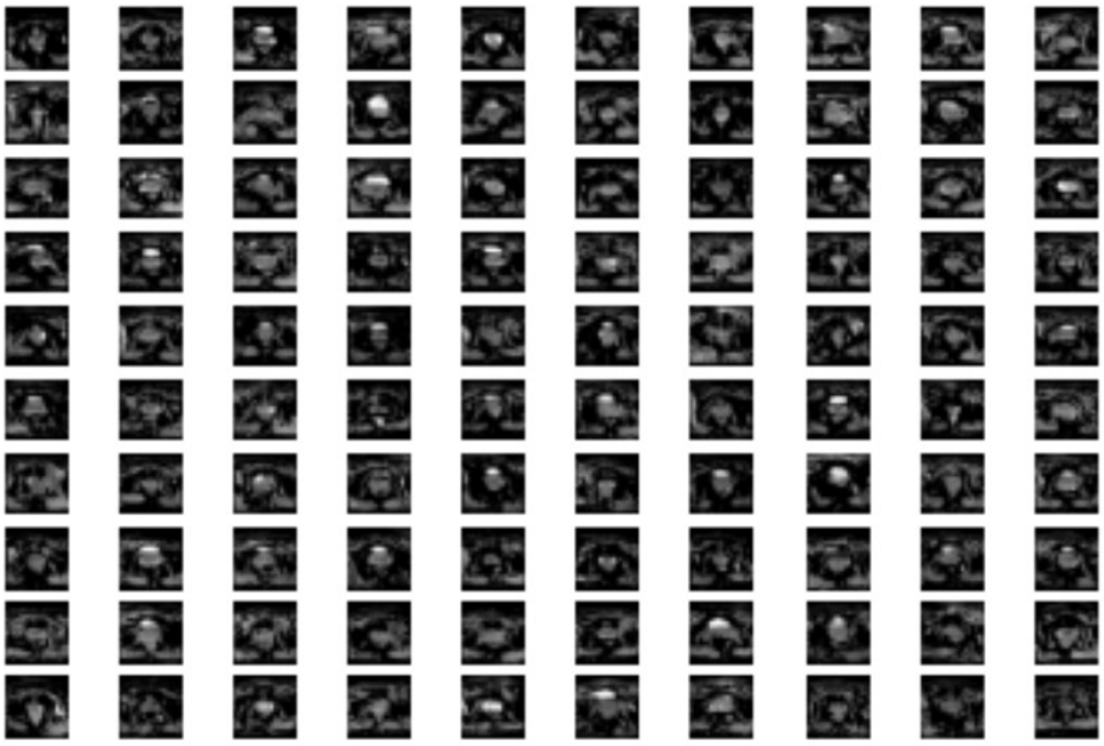


Figure C.6: cGAN 3000 epochs

Appendix D

Code of the models

The relevant code for the models used in this thesis is uploaded on // `https://github.com/omer-parvez/master_thesis` and will be available after 1_{st}, September, 2021. Due to copyright issues and the research work of this thesis will be presented to a conference. Explanation about the used libraries will be available in **README.txt**