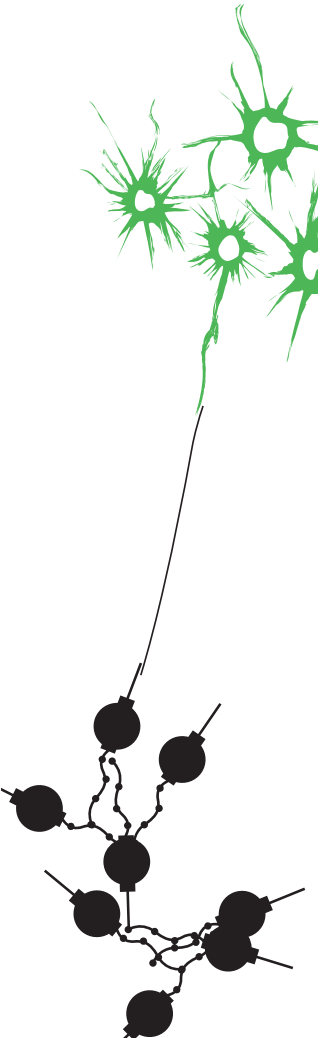


ROBOTICS AND MECHATRONICS

MACHINE LEARNING AND DEEP LEARNING FOR BETTER ASSESSMENT OF THE BIG-3 DISEASES IN HEALTH CARE

E.I.S. (Elfi) Hofmeijer

MSC ASSIGNMENT

Committee:
prof. dr. ir. C.H. Slump dr.
ir. F. van der Heijden
prof. dr. ir. R.N.J. Veldhuis

December, 2019

053hofmeijer2019
Robotics and Mechatronics
EEMathCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Summary

The research project B3CARE aims to advance the technology readiness level for screening of the three diseases, Lung Cancer, Chronic Obstructive Pulmonary Disease (COPD) and Cardiovascular Disease (CVD), by accelerating software development for data post-processing of CT-images. To realise widespread implementation of screening for these three diseases, it is necessary to inform/train (technical) physicians on how to exploit computer assistance based on post-processing software like machine- and deep learning. In this thesis, there will be a focus on the training and education work package of the B3CARE project to prepare (technical) physicians to be able to understand and use advanced software tools in clinical care. A focus is put on one particular task in this work package: *The development of a simulator of patient cases*. Since both the training of physicians and the training of the advanced software tools rely on the amount and variability of patient cases, it is desirable to be able to create them. The objective of this thesis will be to investigate to what extent a generative model, especially a Generative Adversarial Network (GAN), can be created to simulate chest CT patient cases with benign or malignant nodules.

This will be accomplished through answering three consecutive sub-questions. It will first be considered in what way images can be created and how the type of image and its difficulty can be influenced. Following this it will be explored in what way the quality of the generated images can be quantitatively assessed and if this corresponds to human judgement. Both these topics will be handled using an image data set of hands with six different classes. Every class will correspond to the number of fingers that is raised by the hand. This data set is chosen, considering that the average person should be an "expert" in hands, and it will therefore be easier to qualitatively assess the generated images. Finally, it will be explored to what degree lung nodules can be recognised and classified using an algorithm. To accomplish this, first of all lung nodules are created using the knowledge obtained in the first two topics.

It could be concluded that conditional GANs (cGANs) are a good method to generating new images of which the type can be influenced. However, influencing this type is only possible if the data set used to train with, is already classified according to some measure of type. It proved to be difficult to quantitatively assess the generated images. Metrics to do so exist and three of them have been explored to see to what extent they correlate with human judgement. The biggest problem with two of these metrics is that they are dependent on the performance of the classifier that they use. Also, two of the metrics are influenced by the ratio of classes occuring in the generated image data sets. Using a classifier trained on a real nodules image data set, the generated nodule image test set received an accuracy of 73%, whereas the real nodule image test set received an accuracy of 70%. This difference might be because the cGAN has only learned some basic features of benign and malignant nodules which makes it easier for the independent classifier to classify them.

Samenvatting

Het onderzoeksproject B3CARE beoogt de technologische paraatheid te verhogen voor het screenen van de drie ziekten, longkanker, chronische obstructieve longziekte (COPD) en cardiovasculaire ziekte (CVD). Dit probeert het project te bereiken door de softwareontwikkeling te versnellen voor de gegevensverwerking van CT-beelden. Om een brede implementatie van screening voor deze drie ziekten te realiseren, is het noodzakelijk om (technische) artsen te informeren/trainen in het gebruik van computerhulp op in nabewerkingssoftware zoals Machine- en Deep Learning. In deze thesis zal de nadruk liggen op het training- en opleidingspakket van het B3CARE-project om (technische) artsen voor te bereiden op het begrijpen en gebruiken van geavanceerde software tools in de klinische zorg. In dit werkpakket wordt aandacht besteed aan één specifieke taak: *De ontwikkeling van een simulator van patiëntcasussen.* Het is wenselijk om deze te kunnen maken, omdat zowel de opleiding van artsen als de geavanceerde softwaretools afhankelijk zijn van de hoeveelheid en de variabiliteit in patiëntcasussen. Het doel van deze thesis is om te onderzoeken in hoeverre een generatief model, in het bijzonder een Generative Adversarial Network (GAN), kan worden gemaakt om CT-patiëntcasussen met goedaardig of kwaadaardige nodules te simuleren.

Dit wordt bereikt door drie opeenvolgende sub vragen te beantwoorden. Eerst zal worden bekeken op welke manier afbeeldingen kunnen worden gemaakt en hoe het type afbeelding en de moeilijkheidsgraad daarvan, kunnen worden beïnvloed. Daarna zal worden onderzocht op welke manier de kwaliteit van de gegenereerde afbeeldingen kwantitatief kan worden beoordeeld en of dit overeenkomt met een menselijk oordeel. Beide onderwerpen worden behandeld met een data set van afbeeldingen waarop handen afgebeeld zijn in zes verschillende categorieën. Elke categorie komt overeen met het aantal vingers dat door de hand wordt opgestoken. Deze data set is gekozen, aangezien de gemiddelde persoon een 'expert' in handen is. Het zal daarom gemakkelijker zijn om de gegenereerde afbeeldingen kwalitatief te beoordelen. Ten slotte zal worden onderzocht in welke mate long nodules kunnen worden herkend en geclassificeerd met behulp van een algoritme. Om dit te bereiken, worden allereerst long nodules gegenereerd met behulp van de kennis die is verkregen in de eerste twee onderwerpen.

Het kan worden geconcludeerd dat conditionele GANs (cGANs) een goede methode zijn om nieuwe afbeeldingen mee te genereren waarvan het type afbeelding kan worden beïnvloed. Het beïnvloeden van dit type is echter alleen mogelijk als de data set waarmee wordt getraind, al is gecategoriseerd op de gewenst types. Het bleek moeilijk te zijn om de gegenereerde afbeeldingen kwantitatief te beoordelen. Er bestaan metrieken om dit te doen en drie daarvan zijn onderzocht om te zien in hoeverre ze verband houden met het menselijke oordeel. Het grootste probleem met twee van deze metrieken is dat ze afhankelijk zijn van de prestaties van de classificator die ze gebruiken. Ook worden twee van de metrieken beïnvloed door de verhouding van categorieën die voorkomen in de gegenereerde data set van afbeeldingen. Met behulp van een classificator, die is getraind op de echte data set met long nodule afbeeldingen, behaalde de gegenereerde data set met long nodules een nauwkeurigheid van 73%. De echte data set met long nodule afbeeldingen behaalde een nauwkeurigheid van 70%. Dit verschil kan komen, omdat de cGAN slechts enkele basiskenmerken van goedaardige en kwaadaardige nodules heeft geleerd. Hierdoor wordt het voor de onafhankelijke classificator gemakkelijker om de nodules te classificeren.

Contents

1 Introduction

Lung Cancer, Chronic Obstructive Pulmonary Disease (COPD) and Cardiovascular disease (CVD) are called the "Big-3" in health care. The general population is frequently encountered with the Big-3 and the diseases are expected to be the leading causes of death by 2050. Early detection and treatment is vital in these diseases in order to reduce morbidity and mortality. (Heuvelmans et al., 2019) Early detection can be accomplished by screening. Its effectiveness for lung cancer has already been proven (The National Lung Screening Trial Research Team, 2011). However, even though the United States has adopted lung cancer screening, lung cancer is still type of cancer that is the leading cause of death. Furthermore, lung cancer is the most common cancer worldwide. According to the World Health Organization (WHO), lung cancer had 2.09 million cases in 2018 and had 1.76 million deaths that same year. (International Agency for Research on Cancer, 2018) Even though precaution might be taken in the form of screening, lung cancer remains very difficult to recognise and categorise.

The diagnosis, or staging, of lung nodules is done by radiologists. Radiologists realise this by using low-dose Computed Tomography (CT) scans from the chest area. Computed tomography is the most commonly used technique for analysing lung nodules and is non-invasive. It creates multiple cross-sectional slices that make it possible to scan through a 3D volume and analyse the internal organs and tissue. Radiologists mostly develop their skills through residency in order to implement their acquired knowledge into daily practice. Skills are hereby transferred from an experienced radiologist to the resident radiologist. After this period, they may extend this through peer-to-peer training. Life-long learning is an essential aspect for each radiologists, especially with the rapid evolving of modern radiology and the technology involved. (Reiner, 2008; Nguyen et al., 2019)

The learning process of a residence radiologist, but also of a more experienced radiologists, relies on stored and new patient cases that they come across. In other words, their skills rely heavily on their experience. This could potentially cause a radiologist (in training) to not have seen certain cases and this may increase the probability of misdiagnosis if such a new case presents itself. The radiologists in general are skilled in performing diagnosis with impressive efficiency and accuracy. However, they are never immune to making errors, no matter how experienced or skilled they are. (Pinto and Brunese, 2010) A possible step to increase their efficiency and accuracy would be if they work together with a Computer-Aided Diagnosis (CAD) device. (Jorritsma et al., 2015) The device could for instance be used as a second opinion or to make a pre-selection in the cases, ordering them from high to low priority. However, radiologists are mostly not familiar with the workings of the CAD and therefore can utilise more of their potential.

1.1 Motivation

Lung cancer has two broad histological subtypes: Non-Small Cell Lung Cancer (NSCLC) or Small Cell Lung Cancer (SCLC). The cells in SCLCs are smaller than in NSCLCs, as the name already suggests. Next to this, SCLCs can metastasise faster to other areas of the body than NSCLCs. (Purandare and Rangarajan, 2015) Lung cancer signifies itself by the presence of lung nodules. A lung nodule can be either malignant or benign. Whereas the benign variant is mostly harmless and does not grow, the malignant variant grows fast and can metastasise to other regions in the body. Characteristics correlating with a lung nodule being either malignant or benign, show a wide spectrum of image appearances. These range from centrally located masses to invading structures, and from smooth masses to spiculated. Moreover, the size and growth rate is also important to determine if the nodule is benign or malignant. (Purandare

and Rangarajan, 2015) For instance, a nodule with size $> 3cm$ has a high probability of being malignant. (Peter B. O'donovan, 1997)

Lung nodules can also be categorised depending on their location; well-circumscribed, juxtapleural, pleural tail and juxtavascular. The well-circumscribed nodules have no extensions into neighbouring structures and are independent, whereas the juxtapleural nodules are connected to neighbouring pleural surfaces and juxtavascular nodules show attachment to proximal vessels. Pleural tail nodules have, as the name suggests, tails connected to the nodule. They are not, however, adherent to the pleural walls. These categorisations are all location dependent. Next to this, it is also possible to categorise nodules as solid nodules, which are the most common type, or subsolid nodules (SSNs). The latter category, SSNs, have so-called partial ground-glass opacity (GGO), and compared to solid nodules, they do not conceal underlying structures. Figure 1.1 shows some examples of these different categories. (Detterbeck et al., 2013; MacMahon et al., 2017)

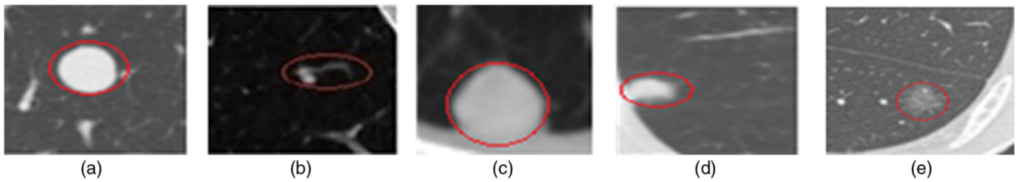


Figure 1.1: Samples of different categories of lung nodules. (a) A solid, well-circumscribed nodule (b) A subsolid, juxtavascular nodule (c) A juxtapleural nodule (d) A pleural-tail nodule and (e) A GGO nodule. Pictures taken from Shaukat et al. (2019).

One important aspect in the interaction between the radiologist and an automated aid is trust. The CAD and the radiologist can be viewed as a diagnostic team. Therefore, the standalone team members must have good performance, but they also have to be able to work together. It has been shown that such a diagnostic team can improve the accuracy in diagnosis compared to the stand alone members (White et al., 2008). However, there are also studies which claim otherwise (de Hoop et al., 2010). Jorritsma et al. (2015) showed that if there is not enough trust in an automatic aid there will be an under-reliance and the full potential of the aid will not be used. However, if there is too much trust in the automatic aid there will be over-reliance which may cause the radiologist to make mistakes he or she would not have made without the device. To assist in this trust issue, the authors suggest to provide an explanation for each specific CAD decision and to educate radiologists of the general methods of CAD algorithms and what their strengths and weaknesses are. (Jorritsma et al., 2015)

To familiarise the radiologists with CAD systems, it is beneficial to incorporate these systems in their education. During this period, the radiologist already learns the basics of a CAD algorithm. However, the CAD can also support the radiologist during and after their training by generating new patient cases. These cases can vary in difficulty and can therefore ensure the radiologist has seen a broad spectrum of cases. The difficulty of a case is defined by the type of nodule and the location it occurs. Nodules in so-called "hiding places" remain unnoticed more often for instance. The CAD may even learn which cases a certain radiologist finds difficult and give supplementary cases to train with. In this way, radiologists would not have to rely on the patient cases they encounter to gain more experience.

1.2 Relation to other work

Generation of new data is also called generative modelling and one of the most popular generative models is called a Generative Adversarial Network (GAN) (Goodfellow et al., 2014). Exploring the possibilities of generative modelling, and especially GANs, can open up doors to new or improved applications. These are also applicable in the field of Medical Imaging. Gen-

erative modelling is applicable in several operations. Noise in medical images can be regarded as "missing" data and GANs could be used to fill in this missing data, i.e. denoise the images. Also, features can be highlighted that were not or only slightly noticed before. In medical image synthesis we can differentiate between unconditional synthesis, i.e. simulating medical images from scratch (a noise vector), and cross modality synthesis, i.e. creating a CT image from an MRI image.

In the first category, unconditional synthesis, Chuquicusma et al. tested if radiologists were able to distinguish generated nodules from real nodules. Using a *Visual Turing Test* one radiologist thought generated nodules were real 100% of the time and a second radiologists thought this was the case for 67% of the time. A major limitation that was encountered concerned the generation of samples. Generated nodules could present features of both malignant and benign nodules. Next to this, more variation and a higher quality of the images would be needed. (Chuquicusma et al., 2018) Mirsky et al. had a similar approach using a 3D conditional GAN to change malicious CT scans to healthy and vice versa. A rectangular cuboid was cut out off the original scan, interpolated and inserted into the GAN. After alteration from healthy to malicious or malicious to healthy, the cuboid was put back into the original scan. The results were tested on three radiologists and they diagnosed 94% of cases where the cancer was removed as being healthy and diagnosed 99% of injected cancer cases as being malignant cancer. After informing the radiologists about the alterations, the scores lowered to 87% and 60% respectively. (Mirsky et al., 2019)

In the second category, cross modality synthesis, Nie et al. used the adversarial strategy to overcome limitations in medical image synthesis from MRI to CT. The GAN takes patch-based MRI images as input and generates a CT image based on this. (Nie et al., 2017) More examples of cross modality synthesis can be found using different modalities, e.g. CT to MR and CT to PET. (Jin et al., 2019; Bí et al., 2018) These two examples give a short summary of the state of the art of GANs in medical imaging. This summary is, however, far from exhaustive and only highlighted topics that might be relevant for this thesis. A more extensive overview can be seen in the review article of Yi et al. (2019), which also discusses areas as segmentation, reconstruction, detection, and registration. A general problem in this area of research is the availability of a reliable way to compare performance, except for judgement by human observers. Next to this, no reference data is available to compare the different GAN-based methods. (Yi et al., 2019)

1.3 Objective

B3CARE is a research project that aims to advance the technology readiness level for screening of the three diseases, lung cancer, COPD and CVD, by accelerating software development for data post-processing of CT-images. To accomplish this, a large scale, high quality research data biobank is necessary with which a robust B3 screening method can be created. Furthermore, this method needs to be implemented in current health care. To realise widespread implementation of B3 screening, it is necessary to inform/train (technical) physicians on how to exploit computer assistance based on post-processing software like machine learning and deep learning designed for assessment of these diseases. Taking this into consideration, the research project is divided into several work packages. In regards to this thesis, there will be a focus on the training and education work package to prepare (technical) physicians to be able to understand and use advanced software tools in clinical care. This work package is divided into several tasks. A focus will be put on task 2: *The development of a simulator of patient cases.* Also, the scope will be narrowed to only focus on lung cancer. At the end of this thesis, in the Appendix, a small contribution to task 1 of the work package will also be presented. Task 1 is concerned with *Development of a post-academic course*, that aims at insight in Machine Learning and its applicability to medical image data.

As discussed previously, automatic aid can already be incorporated in an early stage. Since both the training of physicians and the training of most CADs rely on the amount and variability of patient cases, it would be desirable to be able to create them. If new patient cases can be created, it should be possible to realise an infinite database of varying cases. Therefore, it should also be realised that there is influence on the type of cases that are generated, i.e. to vary the degree of difficulty. In this way, it is always possible to train on new data without having to search or wait for new patient cases. This would be convenient for both radiologists and (technical) physicians in training as well as for the training of CADs.

To aid in the aforementioned problem, the objective of this thesis is to investigate to what extent a generative model, specifically a Generative Adversarial Network (GAN), can be created to simulate chest CT patient cases with benign or malignant nodules. Ideally, these generated patient cases should be indistinguishable from real cases by both radiologists and existing CADs for lung nodule detection. The contributions specifically will be to answer the following subquestions:

- In what way can images be created and how can the type of image and its difficulty be influenced?
- How can the quality of the generated images be quantitatively assessed and does this correspond to human judgement of the quality?
- To what degree can the nodules be recognised and classified using an algorithm and/or an experts opinion?

At the end of this thesis, it will hopefully be clear what the means are to generate an image and how its difficulty can be influenced. Using this, it would be possible to do a quality assessment of the image using both human judgement and existing quantitative metrics. Both results could be compared to come to a conclusion as to what metrics are best to use. This is preferable to know, because hence one is not reliant on people to assess the quality of image. Collecting human judgement can be time intensive and difficult, especially if there is a very limited expert pool. Finally, using the knowledge previously gained, lung nodules will be created and assessed.

1.4 Document structure

This thesis is organised in six chapters, as follows. The current chapter introduced the research project B3CARE and the focus of this thesis. The necessary background to support this objective is also given and some additional information to realise the objective. Chapter 2 will give all the necessary background information to understand the remainder of the thesis. It will explain the workings of a GAN, the main limitations they have, and how they can be evaluated. Chapter 3, Chapter 4, and Chapter 5 will be concerned with each sub-question as described above respectively. Each of these chapters will be divided into a small introduction, methods, results and a discussion. Chapter 6 will contain a final discussion about the methods and will give a relation to other research. Chapter 7 will conclude this thesis and give insight in future work that can be explored. Finally, Appendix A contains a short exploratory research into the creation of an assignment to support (technical) physicians in the understanding of Convolutional Neural Networks (CNNs) as part of task 1. This is specifically done by creating an assignment that aids in visualising the decisions made by a CNN in order to understand the decision better. This does not directly correlate to the main objective of this thesis, but it is part of the training and education work package of the B3CARE research project. It is therefore placed in the Appendix and can be read by everyone interested.

2 The Generative Adversarial Network

One of the first papers published that can be regarded as dealing with the subject of Artificial Intelligence (AI) was in the 1950's. The paper was called *"Computing Machinery and Intelligence"* and suggested a test for determining whether or not a computer can be viewed as intelligent. It was written by A.M. Turing and the test later became known as 'the Turing test'. (Turing, 1950) In 1956 the first neural networks were described and in 1961 Rosenblatt published a paper about the principles, the motivation and the accomplishments of the perceptron. (Allanson, 1956; Rosenblatt, 1961) Also, the first machine learning algorithm was written in 1963 that could overcome the problems of the limitations of this perceptron. (Uhr and Vossler, 1961) Due to financial and psychological reasons, the research into AI had reached a low point in the 1970's, but because of advances in neuroscience and in computer technology the interest in neural networks was renewed in the 1980's. With the increase of computational power in the 1990's AI began to be used in fields as logistics and medical diagnosis. The first computer that could beat the world reigning chess champion was developed in 1997 and was called DeepBlue. (McCorduck, 2004) From the 2000's onward, development in the field of AI grew rapidly. Google begins developing autonomous cars, Apple's Siri is introduced and more research is conducted in various fields; economics, logistics, medical imaging, etc. A search for "Artificial Intelligence" in 2018 alone on PubMed gives 8,203 hits, which is almost double the amount of hits 10 years before.

As one can understand, AI is still a growing field of interest with a wide variety of domains in which it can be applied. It is applied for translation of languages, automation of routine labour tasks and also to diagnose medical images. Nowadays, AI is divided into several sub-fields of which one is Machine Learning. Machine Learning specifically focuses on acquiring patterns from raw data. This is realised using so-called features that are pieces of information from the entire representation of the data. Using these techniques, AI is, in a way, able to gain its own knowledge and make decisions. It is however, not always possible to define or extract these features. At this point, Deep Learning can offer a solution, since it can build complex concepts out of simpler ones. (Goodfellow et al., 2016) Figure 2.1 shows a schematic of the relationships between these different disciplines of AI.

From this point onward, this chapter shall continue to explore the field of Deep Learning. Specifically it will focus on Deep Learning Networks that can generate data, generative models. It will highlight one generative model in particular; the Generative Adversarial Network (GAN). This chapter will describe the basic concept of a GAN, the mathematics behind the GAN and what problems the technique still faces. Before explaining the principle of operation of a GAN, it will be shortly outlined why GANs are chosen over other popular generative models.

Generative models

The term 'generative model' can be used in different contexts. In this case, a model that is able to learn to represent an estimate of a data distribution using a training set containing samples of that data distribution. Two types of generative models exist; explicit models and implicit models. Explicit models are trying to learn the parameters of the distribution of data. They take a training set of examples and try to return an estimation of the data distribution from which the samples came. Implicit models do not learn this distribution, instead they focus on a stochastic procedure that directly creates the data. The three most popular generative models are Fully Visible Belief Networks (FVBNs), Variational Autoencoders (VAEs) and GANs. FVBNs and VAEs belong to the first category, explicit models, whereas GANs belong primarily to the latter category, implicit models. (Goodfellow, 2016) Since FVBNs and VAEs are also popular

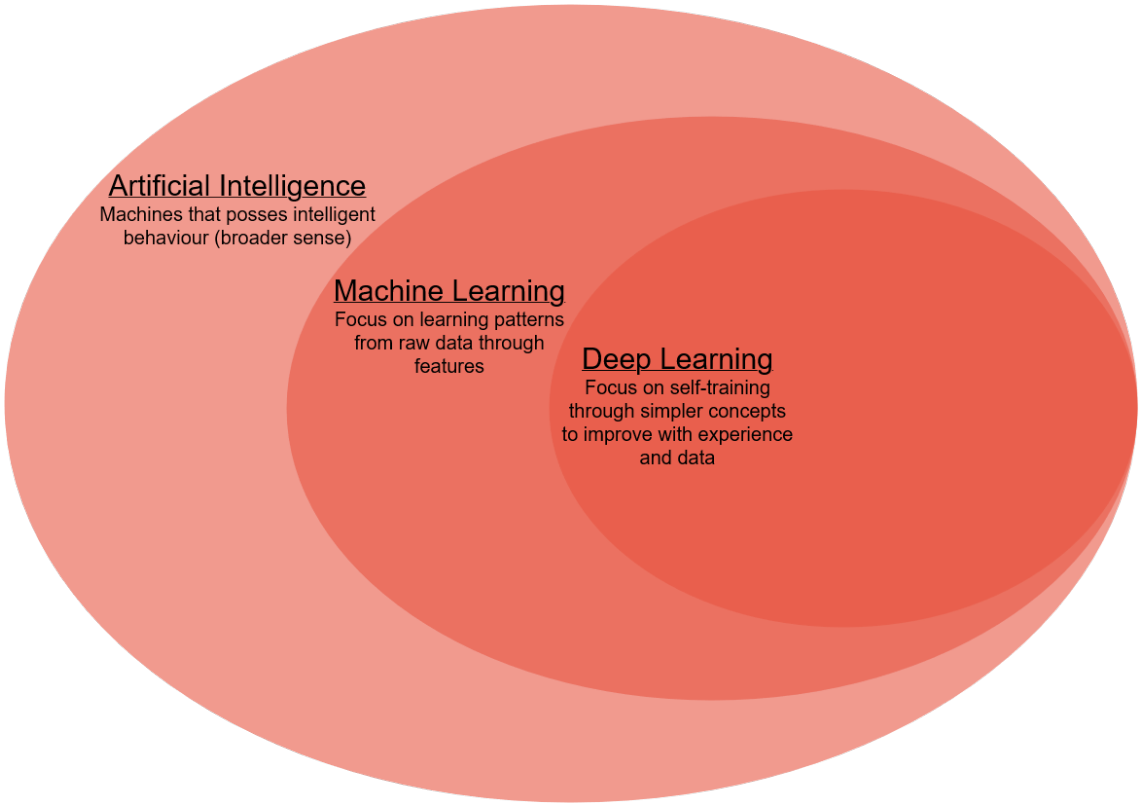


Figure 2.1: Schematic representation showing that Deep Learning is a kind of Machine Learning, which is used for many approaches of Artificial Intelligence.

generative models, a short comparison will be made to explain why GANs are chosen as the model of choice.

As mentioned before, FVBNs (Dayan et al., 1996) belong to the category of generative explicit models. To be more specific, FVBNs define an explicit density model that is computationally tractable. A type of FVBN that is used for image generation is the so called pixelCNN, first introduced by van den Oord et al. (2016). In the case of an image, the joined probability distribution of pixels is decomposed over an image x. This creates a product of conditional distributions with x_i a single pixel. This is accomplished using the chain rule of probability:

$$p_{model}(x) = \prod_{i=1}^{n^2} p_{model}(x_i | x_1, ..., x_{i-1}) \tag{2.1}$$

Each pixel x_i is conditioned on the previous pixels. Therefore, each pixel has a probability distribution that is a function of all the pixels before it. Only the first pixel is independent. Pixels are generated from left to right and from top to bottom. In the deep CNN, filters of the convolutional layers are masked during training to make sure only information from pixels before is used. (Van Den Oord et al., 2016; van den Oord et al., 2016) A drawback in this method is that samples must be generated one entry at a time and can therefore not be parallelised. GANs can generate multiple samples in parallel, which results in a greater generation speed. (Goodfellow, 2016) When keeping in mind the final goal, it is not preferable if the model needs much time to generate each new case.

Contrary to FVBNs, the VAEs (Kingma, 2013; Rezende et al., 2014) use an explicit density model that is intractable. As the name suggests, VAEs contain an auto-encoder network. This network consists of an encoder and a decoder. The encoder maps real data to a latent space and the decoder maps the latent space to the output. The output in an auto-encoder is the same as the

input. This means that the auto-encoder is trying to recreate an image from its compressed version. The amount of nodes at the end of the encoder is very important. If there are too little, recreation of the image is limited. VAEs use this auto-encoder architecture to represent a data generating distribution. Random samples from this encoded distribution can be decoded by the decoder to generate new images. The posterior distribution is approximated by the encoder and the distribution is typically intractable. The VAE, as the name suggests, uses variational inference to approximate this posterior distribution. Variational inference is a technique to approximate complex distributions (Jordan et al., 1999). A parametrised family of a distribution is set and the best approximation for the target distribution has to be found among this family. This family could for instance be a family of Gaussians. If the prior distribution or the approximate posterior distribution is not good enough, it might cause the encoder to not learn the target distribution but something else. This can occur even given enough training data and a perfect optimisation algorithm. GANs on the other hand, retrieve this true distribution with infinite data and a large enough model, without any priors. (Goodfellow, 2016; Rezende et al., 2014)

Finally, comparing the generated samples of all three methods, the produced samples by GANs are subjectively regarded as being better. Unfortunately all metrics currently in use to assess the quality of the images, still have weaknesses and are not universally applicable. (Goodfellow et al., 2016) Therefore, it is difficult to compare different generative models. On the other hand, GANs also bring a new disadvantage with them. Keeping the final goal of this thesis in mind, it would be preferable to able to create samples that have the best quality possible. Time needed for the generation of these samples should also be as short as possible in order to be able to efficiently assess multiple samples. However, GANs do present one big disadvantage, as will be further explained in the following sections, GANs require a Nash equilibrium (Ratliff et al., 2013).

2.1 Principle of Operation

A GAN consists of two convolutional neural networks (CNNs); a discriminator (D) and a generator (G). As the name suggests, the generator is burdened with generating new data which resembles a given real data distribution. The discriminator has as its goal to determine if the data, that is presented to it, originated from the real data distribution or from the synthesised data distribution. This results in the generative model competing with the discriminator model, since each has a different objective. The discriminator tries to classify generated data as fake and real data as real, whereas the generator tries to let the discriminator classify the generated data as being real too. The basic concept is also schematically shown in Figure 2.2.

Consider a data distribution p_{data} with samples x and a random distribution p_Z with samples z. The goal of G will be to convert p_Z into a distribution which resembles p_{data} as close as possible. This conversion results in a distribution p_g with samples $\hat{x}$. Distribution p_Z serves as a prior on the input noise variables. Both G and D will be considered as differentiable functions represented by a multilayer perceptron. The conversion of p_Z to p_g is realised by $G(z;\theta_g)$, with θ_g the parameters of G. The second multilayer perceptron $D(x;\theta_d)$, with θ_d the parameters of D, serves as a classifier and will output a value between 0 and 1. An output of 1 corresponds to a certain decision that the input is real and an output of 0 that the input is fake. (Goodfellow et al., 2014)

2.1.1 Mathematics behind training GANs

To train a GAN we have to consider two situations: updating the discriminator D and updating the generator G. These situations occur one after the other, so D and G are not trained simultaneously. Furthermore, both function D and G are considered differentiable with respect to its inputs as well as to its parameters. The cost functions of both networks are influenced by para-

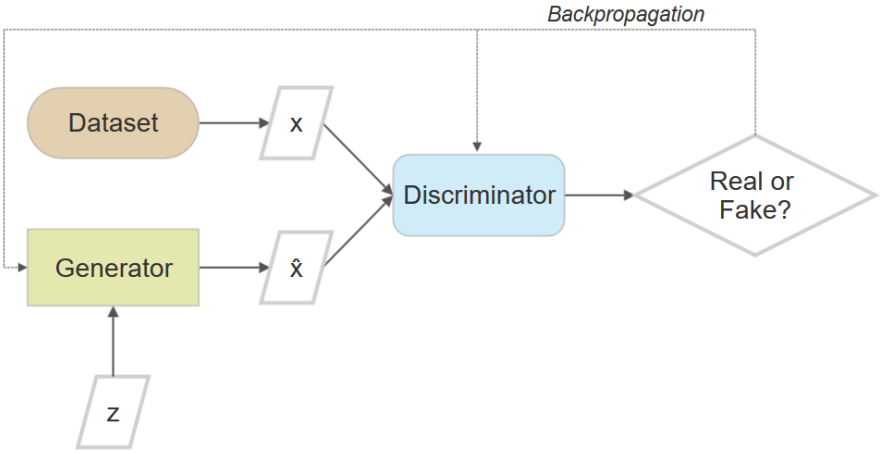
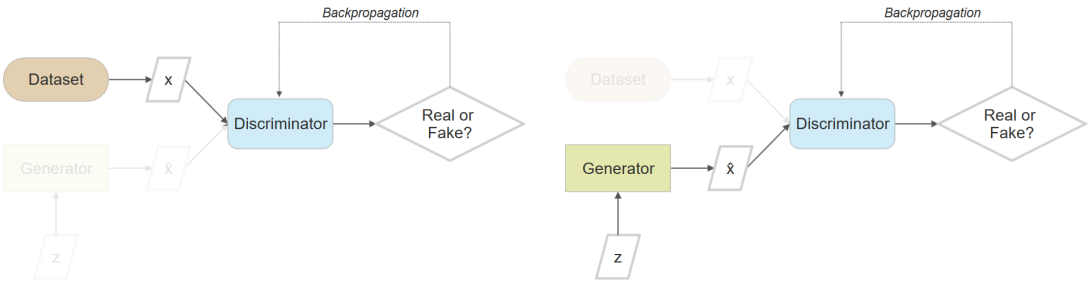


Figure 2.2: Schematic of the basic concept of GANs. The generator takes as input a noise vector (z) and has as an output a generated image ($\hat{x}$). The discriminator takes this generated images as input, but also has images from the real dataset as input (x). The discriminator has to decide if the image it receives is either a real image or a fake/generated image. Through backpropagation both the discriminator and the generator improve.

meters from both networks. They are however, only able to adjust their own parameters. This situation is therefore sometimes described as a game instead of an optimisation problem. The solution to a game is a Nash equilibrium, which is defined as the state in which none of the two players can improve its state while the state of the other is remained fixed (Ratliff et al., 2013). Following, it will be described how to reach such an equilibrium when training a GAN. To get to this (theoretical) solution, first the general training process of D and G will be described and this will be followed by the derivation of the loss function for both networks. Gradients can be calculated from these loss functions with respect to the parameters of the networks. After describing the theoretical solution, also the algorithm for the theoretical training process will be described.

First of all, the situation of D will be considered. Its cost function resembles $J^{(D)}(\theta_d, \theta_g)$, but it will only be able to control θ_d. When updating D also two situations present themselves. In the first situation, which is also showed in Figure 2.3a, $D(x;\theta_d)$ is updated using samples x from p_{data}. The output is the probability that x came from $p_{data}(x)$ with $D(x) \in [0,1]$. In the second situation, which is also showed in Figure 2.3b, $D(x;\theta_d)$ is updated using samples $\hat{x}$ from p_g. These samples are mapped from p_Z by $G(z;\theta_g)$ into samples from p_g. These samples coming from p_g are therefore also noted as $G(z)$. The output is the probability that $G(z)$ came from $p_{data}(x)$ with $D(G(z)) \in [0,1]$. (Goodfellow et al., 2014; Goodfellow, 2016)



(a) Update with real data.

(b) Update with fake data.

Figure 2.3: Situation when D is trained. To train D it requires one update with real data and one update with the generated data.

Secondly, the situation of G will be considered. Its cost function will look like $J^{(G)}(\theta_d, \theta_g)$, but it will only be able to control θ_g. This situation is very similar to the situation in which D is updated with generated data. In this case however, $G(z;\theta_g)$ is updated and $D(x;\theta_d)$ is kept fixed. The situation is also shown in Figure 2.4. The output is the probability that $G(z)$ came from $p_{data}(x)$ with $D(G(z)) \in [0,1]$.

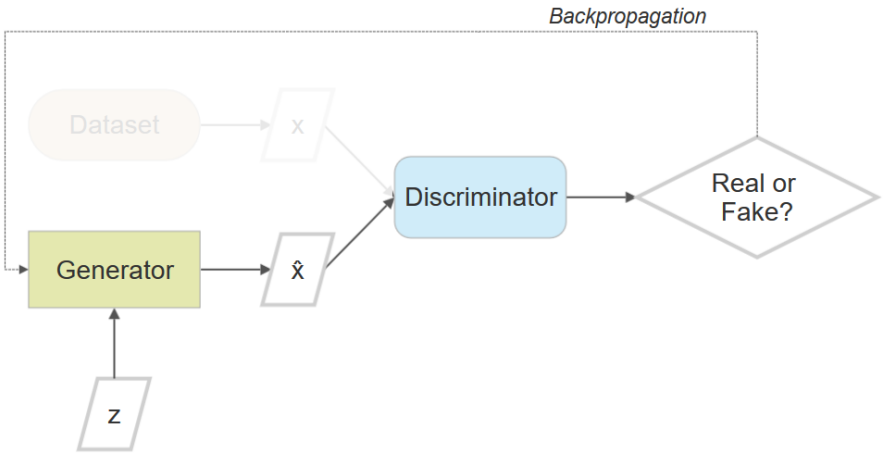


Figure 2.4: Situation when G is trained.

Loss function

The cost functions comes from the binary cross-entropy loss function:

$$L(\hat{y}, y) = [y log\hat{y} + (1-y) log(1-\hat{y})] \tag{2.2}$$

With y the label associated with the input data, which is either real ($y = 1$) or fake ($y = 0$). The label $\hat{y}$ is associated with the probability output of D. If we take the situation of Figure 2.3a and fill in eq. 2.2, we will get a loss function as shown in eq. 2.3. A sample that comes from p_{data} will have input label $y = 1$, since it comes from the real data distribution. It will have a probability output $\hat{y} = D(x)$ and this will result in:

$$L(D(x), 1) = logD(x) \tag{2.3}$$

Updating the parameters θ_d results in $\hat{\theta}_d = \underset{\theta_d}{argmax}\ logD(x)$. With $\hat{\theta}_d$ the updated parameters of D. It is maximised since D needs to classify data from $p_{data}(x)$ as real. If we take the situation of Figure 2.3b or Figure 2.4 and fill in eq. 2.2, we will get a loss function as shown in eq. 2.4. A sample that comes from p_g will have input label $y = 0$, since it comes from the generated data distribution. It will have a probability output $\hat{y} = D(G(z))$ and this will result in:

$$L(D(G(x)), 0) = log(1 - D(G(z))) \tag{2.4}$$

Updating the parameters θ_d results in $\hat{\theta}_d = \underset{\theta_d}{argmax}\ log(1 - D(G(z)))$. With $\hat{\theta}_d$ the updated parameters of D. It is maximised since D needs to classify data from $p_g(x)$ as fake. Updating the parameters θ_g results in $\hat{\theta}_g = \underset{\theta_g}{argmin}\ log(1 - D(G(z)))$. With $\hat{\theta}_g$ the updated parameters

of G. It is minimised since D needs to classify data from $p_g(x)$ as real. The situation described above is applicable when one sample is used. When looking at a data set with N samples the objective function of D results in eq. 2.5. Where y_i is considered the label of image x^i from p_{data} or image z^i from p_g.

$$J^{(D)}(\theta_d,\theta_g) = -[\frac{1}{N}\sum_{i=1}^{N/2} y_i\ log(D(x^i)) + \frac{1}{N}\sum_{i=N/2}^{N}(1-y_i)\ log(1-D(G(z^i)))] \tag{2.5}$$

The size of the data set is represented by N. Since N consists of samples from p_{data} and p_g, both parts of eq. 2.5 has $N/2$ samples. It is a 50% probability if x^i or z^i is sampled. All y^i are likely to occur and the sums of eq. 2.5 can be converted into expectations:

$$J^{(D)}(\theta_d,\theta_g) = -[\frac{1}{2}\{\mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \frac{1}{2}\mathbb{E}_{z\sim p_g(z)}[log(1-D_(G_(z)))]\}] \tag{2.6}$$

In this function $\mathbb{E}_{x\sim p_{data}(x)}$ denotes the expectation over the real data distribution $p_{data}(x)$. Therefore $\mathbb{E}_{z\sim p_g(z)}$ denotes the expectation over the generated distribution $p_g(z)$. Mostly, the factor $\frac{1}{2}$ is ignored, since it's constant.

The objective of D, through training, is to minimise the loss. This can also represented by maximising the opposite of the loss and this results in:

$$\max_{\theta_d}[\{\mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_g(z)}[log(1-D(G(z)))]\}] \tag{2.7}$$

The objective of G is to make D produce $D(G(x)) = 1$, in other words, it wants to make D classify a generated sample as a real sample. To achieve this, it wants to maximise the uncertainty of D. When looking at a data set with N samples the objective function of G results in eq. 2.8. Where y_i is considered the label of image x^i from p_{data} or image z^i from p_g.

$$J^{(G)}(\theta_d,\theta_g) = [\frac{1}{N}\sum_{i=1}^{N}(1-y_i)\ log(1-D(G(z^i)))] \tag{2.8}$$

The size of the data set is represented by N. In this case N consists of samples from p_g. The sum of eq. 2.8 can be converted into an expectation:

$$J^{(G)}(\theta_d,\theta_g) = \frac{1}{2}\mathbb{E}_{z\sim p_g(z)}[log(1-D(G(z)))] \tag{2.9}$$

The objective of G, through training, is to minimise this cost in order to increase uncertainty of D. This results in:

$$\min_{\theta_g}[\mathbb{E}_{z\sim p_g(z)}[log(1-D(G(z)))]\}] \tag{2.10}$$

Also in this case the constant $\frac{1}{2}$ is mostly left out. The cost of D and G together can be summarised through a value function $V(\theta_d,\theta_g) = -J^{(D)}(\theta_d,\theta_g)$ (Goodfellow et al., 2014):

$$\min_G \max_D V(D,G) = \{\mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_g(z)}[log(1-D(G(x)))]\} \tag{2.11}$$

In practice, this objective function is rarely used. This is because the loss function ($min\ log(1-D(G(z)))$) of G does not provide a strong or steep gradient signal when training starts. It saturates and the numerical derivative is difficult to calculate. Therefore, in practice, $J^{(G)}(\theta_d,\theta_g) =$

$-\frac{1}{2}\mathbb{E}_{z\sim p_g(z)}[logD_(G_(z))]$ is used instead of eq. 2.9, which does not saturate at the beginning of training.

Theoretical solution

In a situation when G is held fixed, the optimal D_G^* can be determined. This result is obtained by maximising eq. 2.11 with respect to D. The derivation of this can be found in Appendix B and the result can be seen in eq. 2.12.

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \tag{2.12}$$

As is known, the objective of G is the opposite of that of D. Therefore, if we want to obtain the optimal G we must minimise eq. 2.11. This solution should also satisfy $p_g = p_{data}$.

Using $D = D_G^*$, the optimal G^* can be expressed using the Jensen-Shannon Divergence (Majtey et al., 2005) $JS(p,q)$:

$$G^* = -log(4) + 2JSD(p_{data}||p_g) \tag{2.13}$$

The derivation of eq. 2.13 can be found in Appendix C. The Jensen-Shannon divergence is defined as:

$$\int JS(p,q) = \frac{1}{2}[D_{KL}(p||\frac{p+q}{2}) + D_{KL}(q||\frac{p+q}{2})] \tag{2.14}$$

with $D_{KL}(p||q)$ the Kullback-Leibler (Kullback and Leibler, 1951) divergence:

$$\int KL(p,q) = \int dx p(x) log\frac{p(x)}{q(x)} \tag{2.15}$$

Minimising the KL divergence will result in the model distribution getting closer to the data distribution, which is the same as maximising the log-likelihood of data under a model. As can be seen form eq. 2.13, it can only be minimised if $p_g(x) = p_{data}(x)$.

Using this information, it is possible to state that the goal of the training of a GAN is to have $p_g(x) = p_{data}(x)$. Substituting this into eq. 2.12, it is possible to see that the optimal discriminator results in $D_G^* = 0.5$. (Goodfellow et al., 2014)

The loss functions for the discriminator and generator associated with the objective function in eq. 2.11 are considered to be used in the so-called minimax GAN (or MM GAN). The non-saturating version of the generator in a GAN is called Non-Saturating GAN (NS GAN). Next to these two, there are multiple other losses possible like the Wasserstein GAN (WGAN), DRAGAN or BEGAN (Arjovsky et al., 2017; Kodali et al., 2017; Berthelot et al., 2017). A summary of all the different types is beyond the scope of this thesis. Although the type of loss used affects the behaviour of the GAN, Lucic et al. (2018) found that with enough hyperparameter optimisation and random restarts the different models can reach comparable scores. This suggest that computational cost is more important.

Theoretical training process

The main algorithm on which training is based was proposed by Goodfellow et al. (2014) and can be seen in Algorithm 1. This algorithm, and the explanations accompanying it, are pro-

posed in a non-parametric setting. This indicates that the convergence of the model is studied in the space of probability density functions.

Algorithm 1: Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for *number of training iterations* **do**

  for *k steps* **do**

    Sample minibatch of m noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from noise prior $p_g(z)$.;

    Sample minibatch of m examples $\{x^{(1)}, ..., x^{(m)}\}$ from data generating distribution $p_{data}(x)$.;

    Update the discriminator by ascending its stochastic gradient:

$$\Delta_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(x^{(i)}) + \log(1 - D(G(z^{(i)})))].$$

  end

  Sample minibatch of m noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from noise prior $p_g(z)$.;

  Update the generator by descending its stochastic gradient:

$$\Delta_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(z^{(i)}))).$$

end

The gradient-based updates can use any standard gradient-based learning rule.

On a finite data set the optimisation of D to completion in the inner loop will result in overfitting. Instead, alternating between k steps of optimising D and one step of optimising G is proposed in the algorithm of Goodfellow et al. (2014). As long as G changes slow enough, D will remain near its optimal solution. The training process is also visualised in Figure 2.5.

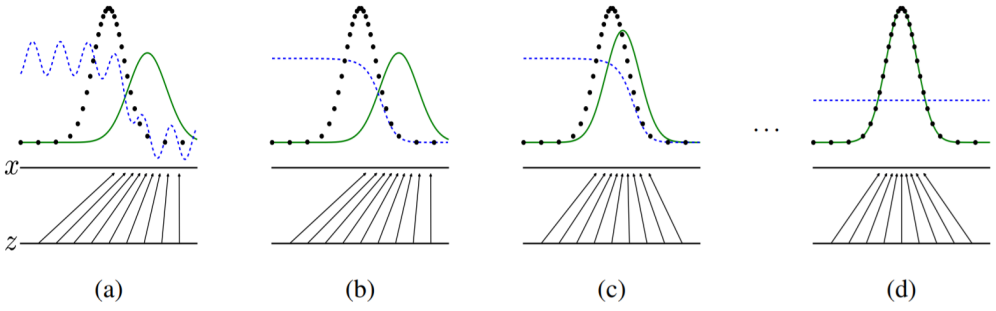


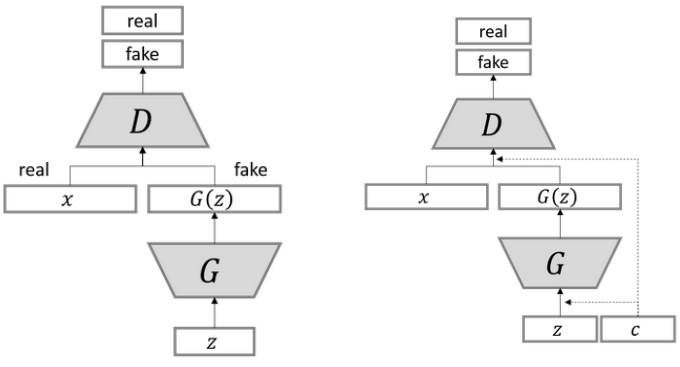
Figure 2.5: Learning process of a GAN. The lower horizontal line represents the domain from which z is sampled. This domain is mapped to the domain of x by $x = G(z)$ and imposes the non-uniform distribution p_g (green solid line). The distribution of the real data p_x is visualised as the black dotted line. The blue dashed line represents the discriminitive distribution D. (a) Shows a near convergence scenario with D the partially accurate classifier. (b) With G fixed, D is trained to discriminate generated samples from real samples. (c) Now keeping the updated D fixed, G is trained and has moved to regions that are more likely classified as real data. (d) At this point both G and D cannot be improved anymore, i.e. $D^*_G = 0.5$ has been reached. This point is only reached if both networks have enough capacity. (Goodfellow et al., 2014)

2.1.2 GAN architecture

Next to variation in loss function, GANs can also vary in architecture. A type of GAN that will be mentioned quite often is the Deep Convolution GAN (DC-GAN) and was introduced by Radford et al. (2015). Before this introduction, GANs were already deep and convolutional, but the DC-GAN has some specific architectural constraints. It is useful to use the name, in order to refer to this specific style of architecture. DC-GANs were the first architectures that enabled the models to learn to generate high resolution images at the first try. (Goodfellow, 2016) The constraints applied to the architecture contained the following (Radford et al., 2015; Goodfellow, 2016):

- Strided convolutions replaced (un)pooling layers in D and for G they were replaced by strided (transposed) convolutions.
- In both G and D Batch Normalization was introduced, except for the last layer of G and the first layer of D.
- In G also Rectified Linear Units (ReLU) activations were used except for the output, which used a hyperobolic Tangent (Tanh).
- In D LeakyReLU is used.
- For deeper architectures, the fully connected hidden layers are removed.
- Instead of Stochastic Gradient Descent (SGD), the Adaptive Moment Estimation (Adam) optimiser is used.

This normal GAN structure is also sometimes referred to as "Vanilla GAN" and can be seen in Figure 2.6a. Before the introduction of DC-GANs, LAPGANs were the only one that could scale to images with a high resolution. (Goodfellow, 2016) The LAPGAN was introduced by Denton et al. (2015). It uses the normal GAN architecture, but integrates it into the framework of a Laplacian pyramid. The idea behind this is to capture the multi-scale structure of natural images by building multiple GANs in series, to ensure that the image structure is captured at each particular scale of a Laplacian pyramid. From the bottom to the top of the pyramid the sample synthesised will have more coarse and more fine features respectively. (Denton et al., 2015)



(a) Vanilla GAN **(b)** conditional GAN

Figure 2.6: Two different GAN architectures. The vanilla GAN can is schematically represented on the left and the cGAN on the right. (Mino and Spanakis, 2018)

As mentioned at the beginning of this thesis, it will be necessary to able to control the type of image that is generated. An extension of the normal GAN is called the conditional GAN (cGAN). As an addition to the general architecture, both D and G receive an extra input, e.g. it might

contain information about the class of the training example. (Mirza and Osindero, 2014) The structure of the network can be seen in Figure 2.6b.

Besides the architecture mentioned above, there is one more architecture that should be highlighted. The Style-GAN was proposed by Karras et al. (2019) and is one of the best performing GANs at the moment. As the name suggests, it is able to adjust the "style" of an image at each convolutional layer based on the latent code. With images of faces for instance, this would mean influence on high-level attributes like pose and identity, and stochastic variation, like hair and freckles. In this way the strength of image features can be controlled at different scales. These modifications only influence the input in G and do not directly influence D or the loss functions. Through this architectural change, intuitive scale-specific mixing and interpolation operations are enabled.

Since GAN research has increased rapidly the last few years, one can imagine that there are more architectures developed than the ones mentioned. This thesis is limited to the architectures mentioned above and further research is beyond its scope. The reader is referred to the papers of Nguyen et al. (2017) and Brock et al. (2018) for examples of two other GANs that generated high quality images. These both contain new architectures, the Plug & Play Generative Network (PPGN) and the BigGAN.

2.2 Main problems and research areas

Since the first GAN network has been proposed, several limitations have been encountered that still require attention (Goodfellow, 2016). In this sections, the most important limitations will be illustrated.

The vanishing gradient problem. As indicated earlier in $G(z;\theta_g)$, θ_g is optimised rather than p_g itself. Due to the discriminator being trained first, and the initial performance of the generator being poor, the derivative of the loss function of the generator ($min\ log(1 - D(G(z)))$) will be almost 0. To avoid this, the loss function can be changed into $max\ log(D(G(z)))$. This will result in the derivatives being very high, which in turn results in better training.

Nash equilibrium. The Nash equilibrium is difficult to achieve because of the non-cooperative game that is played between the generator and the discriminator. They both update their costs without informing the other. Therefore the convergence of their gradients cannot be guaranteed. As the amount of iterations increase, the network might become more unstable instead of going into a Nash equilibrium. Another possibility is that both players might be undoing each others work.

Mode collapse. During training, after a certain amount of epochs, the generator may collapse into a setting where it always produces the same output. The cause for this is an effect from having a multi-model distribution. If the discriminator improves at recognising a real distribution sample from a single mode, two things can happen:

- The generator might focus on the other modes as to avoid failing in the particular mode the discriminator performs very good on.
- The generator might focus on that particular mode to beat the discriminator in this mode.

Both scenarios will give the same reward to the generator. However, in this case, it is beneficial to master one mode, instead of performing average on multiple modes. Complete mode collapse is rare in practice, but partial mode collapse does happen. One can imagine this as the same dog that is reproduced but from different views or with a different fur colour. This can also be explained using the optimal solution of the GAN game $G^* = \min_G \max_D V(G,D)$. In this case G^* samples from the real data distribution. However, gradient descent, in this case simultan-

eously, does not prioritise $min\ max$ over $max\ min$. If it behaves like $G^* = \max_D \min_G V(G,D)$ then G^* will learn to map every sample z to a single sample x that D would consider real rather than fake. A proposed solution is the use of mini-batch features. In short, mini-batch features allows D to compare a sample to both a mini-batch of real samples and of generated samples.(Salimans et al., 2016)

Problems with counting, perspective and global structure. The network does not understand that, for instance, a monkey only has two eyes and one nose. This can be observed in Figure 2.7a. Next to this, the network cannot differentiate between front and back and between 2D and 3D. This can be observed in Figure 2.7b. Furthermore, the network does not understand the concept of holism. As a result of this, a cow might be produced which is both standing on its hind legs, and standing on all fours. This "cow" can be observed in Figure 2.7c. There is no solution to these problems yet.

(a) Counting problem **(b)** Perspective problem **(c)** Structure problem

Figure 2.7: Three results of a GAN network that encounters a couple of problems: (a) Problems with counting. (b) Problems with perspective. (c) Problems with structure. Pictures taken from Goodfellow (2016).

Quantitative evaluation. An approach to quantitatively evaluate GANs without limitations does not exist yet. Theis et al. (2015) describe some difficulties that arise when evaluating a model, e.g. models that have visually pleasing samples may have bad log-likelihood and vice versa. An extensive overview of GAN evaluation techniques can be found in the review paper of Borji (2018). A few metrics will be described in the next section.

2.2.1 Tips & Tricks

Several tips and tricks are developed to improve GAN training. However, these should all be considered as techniques that are good to experiment with, but not as a guarantee of improving the GAN. These techniques are mostly well thought out, but most of the time it is complicated to indicate how effective they are. In some situations, they might have proven their worth, whereas in other situations they might have worsened the performance. (Goodfellow, 2016) Some techniques are also easier to implement than others.

One of the first tricks is to normalise your input images in the range of $[-1,1]$. In this way, D will always get input images in the same range. Another trick that seems to work is to let G sample from a Gaussian distribution, which is spherical like, instead of from a uniform distribution, which resembles a hypercube. Also, the use of the Adam optimiser seems to improve results (Radford et al., 2015). The same holds for making the last activation layer of G a hyperbolic Tangent (Tanh) activation layer.

Balancing G and D. Balancing D and G and reaching convergence is one of the most difficult problems. An intuitive approach to reach this convergence, might be to make sure both networks are equally strong. However, as described in the sections before, theoretically it is only possible to get an optimal G if there is an optimal D. In this, on the other hand, hides another

trap. One might think that it would be better to update D multiple times before updating G. In practice, however, this does not work out very well most of the time. (Goodfellow, 2016)

Training with labels. The idea of using labels was first tried by Denton et al. (2015) who created the cGAN. In this experiment it was proven that the cGAN outperformed the vanilla GAN. However, it is not entirely clear why training with labels increases the performance. It might be because it gives the GAN a more information, causing it to perform better. However, it might also be possible that it does not perform better at all, and that the images that are created are more biased to look like what the visual perception of humans would expect them to look like. Humans have labelled the images because of what they see in these images and have therefore classified the images according to their visual perception. The cGAN receives this information and might perform better because of this when assessed by human judgement through visual assessment. Depending on the goal, it is arguable if it matters which scenario is actually the case. (Goodfellow, 2016)

One sided label smoothing. As explained earlier, D needs to reach its optimal version before G can be optimised. However, in this process, the predictions of D might get too confident. In practice it is not a good idea to make G stronger or D less strong. Before doing any of that, one-sided label smoothing can be tried which was reinvented by (Szegedy et al., 2016). This technique is used to regularise the probabilities of D when they get too extreme. The labels for real data ($y = 1$) and generated data ($y = 0$) are replaced by values like 0.9 or 0.1. If we apply label smoothing, the optimal D would look like eq. 2.16.

$$D_G^* = \frac{\alpha p_{data}(x) + \beta p_g(x)}{p_{data}(x) + p_g(x)} \tag{2.16}$$

With α the smoothed value for real data and β the smoothed value for generated data. This adjustment does not encourage D to start making the wrong predictions, but it does make the correct choice less confident. In the case of GANs, the β parameter should always be zero. If it would be non-zero the optimal function of D would change. If p_{data} would be really small and p_g would be really large, than there would be no stimulus to move the model any closer to the real data. (Salimans et al., 2016)

Figure 2.8: Result of what happens if Batch Normalisation is used with small size mini-batches. The upper and the lower set are both a mini-batch produced by a generator. The variation in the mean and standard deviation of feature values in a mini-batch can have an influence that is more dominant than the images individually in that mini-batch. Figure taken from Goodfellow (2016).

Virtual Batch Normalisation. In practice, GANs mostly make use of small mini-batch sizes to decrease computational costs. A batch normalisation layer in the network is used to scale the

data. It may however, cause the GAN to fall into mode collapse. The mean and standard deviation in a batch normalisation layer are calculated for every mini-batch. The small batch sizes may result in large fluctuations in the mean and standard deviations between mini-batches. These fluctuations between mini-batches can even become large enough that they have more influence on the image being produced by G than one sample z has. In this case the output of a network for such an input z will be influenced by the other inputs z' from that same mini-batch. Salimans et al. (2016) have found a solution to this problem, by making use of a reference batch. This reference batch is used to normalise the mean and standard deviation by averaging each sample in a mini-batch with this reference batch.

2.3 Evaluation of GANs

A problem previously mentioned, is that there does not exist a technique that can quantitatively and qualitatively compare the performance of different GANs. However, multiple different performance evaluation techniques have been developed. Each of which focus in varying amounts on either quantitative are qualitative assessment. (Borji, 2018) Using quantitative or qualitative metrics is a difficult question. Regarding the generation of images, one could want to focus on qualitatively good images in order to convince observers into thinking that the images are real. This could be regarded as the ultimate test, but in this case it could be possible that the model focuses on only a small part of the data. An expert in one mode would probably be favoured over an average performance in all modes. On the other hand, one could want to focus on quantitatively assessing the models to make sure the assessment is less subjective. This could, however, result in images that do not correlate with human judgement on the generated images. According to Borji (2018), an efficient GAN metric should:

1. favour models that generate high fidelity images.
2. favour models that generate a high diversity of images (therefore the metric will be sensitive to over-fitting, mode drop and mode collapse).
3. favour models that can control the sampling (that have disentangled latent spaces).
4. have well-defined bounds.
5. be invariant to image transformations and distortions. The semantic meaning of an image does not change when the image is rotated by a few degrees and therefore the metric should not be affected by this.
6. agree with judgement according to human visual perception.
7. have a low computational complexity and low sample complexity.

For an extensive review the reader is referred to Borji (2018). Three evaluation metrics will be further discussed below that meet the above criteria to varying degrees; the Gan Quality Index (GQI) (Ye et al., 2018), the Frèchet Inception Distance (FID) (Heusel et al., 2017) and Number of Statistically-Different Bins (NDB) (Richardson and Weiss, 2018). Table 2.1 gives a short overview of which criteria the three different metrics meet.

2.3.1 Classification Performance

The GAN Quality Index (GQI) was introduced by Ye et al. (2018) and uses a form of classification performance to evaluate GANs. Next to training a generator on the real data, also a classifier (C_{real}) is trained on the real data. Labels for the generated data are obtained by classifying them according to C_{real}. During this classification process, a threshold is set in order to discard images with low pseudo label scores. Therefore, images with low quality or images that do

Table 2.1: Overview of the different seven criteria and the performance of the three different metrics, GQI, FID and NDB on these criteria. The ratings are relative and "-" indicates that it is unknown. (Borji, 2018)

	GQI	FID	NDB
Fidelity	high	high	low
Diversity	low	moderate	high
Control Sampling	-	-	-
Well-defined bounds	$[0,100]$	$[0,\infty]$	$[0,\infty]$
Distortions	-	high	low
Human judgement	low	high	-
Computationally efficient	-	high	-

not belong to one of the existing classes are removed. In turn another classifier ($C_{generated}$) is trained using the generated data. Both these classifiers are evaluated using the same test set that consists of real data. Using these two classifiers the GQI of the GAN can be calculated (Ye et al., 2018):

$$GQI = \frac{ACC(C_{generated})}{ACC(C_{real})} * 100\% \tag{2.17}$$

The accuracy of $C_{generated}$ is described as $ACC(C_{generated})$ and the accuracy of C_{real} as $ACC(C_{real})$. The GQI is a number between 0 and 100. When a higher value is obtained, the generated distribution matches the real distribution better. Advantages of this metric are that it has well defined bounds and that it is able to capture the fidelity of generated images well. However, it is necessary to train two classifiers in order to evaluate the generated images of the GAN, which can be time intensive. Also, the result of this score is therefore dependent on the performance of the trained classifiers. Lastly, the score is an indirect technique to assess the generated images, since the generated images are not used directly. (Borji, 2018)

2.3.2 Frèchet Inception Distance

The Frèchet Inception Distance (FID) was first introduced by Heusel et al. (2017) and converts a set of generated images into feature space using a specific layer of the Inception Network (Szegedy et al., 2014). The Inception Network is a deep CNN that set the state-of-the-art for classification and detection in 2014 on the ImageNet data set. The network is still being used because of its good performance on the ImageNet data set. It is also possible to use a specific layer of another trained CNN. The probability $p_1(.)$ of observing real images and the probability $p_2(.)$ of observing generated images should be the same ($p_1(.) = p_2(.)$) if and only if $\int p_1(.)f(x)dx = \int p_2(.)f(x)dx$. A basis $f(.)$ should span the function space where $p_1(.)$ and $p_2(.)$ exist. Polynomials of the data x are described by $f(x)$. When used for calculating the FID, the specific layer mentioned earlier (a coding layer of the CNN) is used instead of x to generalise the polynomials in order to obtain features that are relevant to human visual perception. Also, only the first two polynomials are considered, the mean and the covariance, since this is more practical. The resulting distributions are assumed to be Gaussian. The quality of the generated

images can be quantified by calculating the Frèchet distance between these two Gaussians, shown in eq. 2.18. (Heusel et al., 2017)

$$FID = ||\mu_1 - \mu_2||^2 + Tr(C_1 + C_2 - 2\sqrt{C_1 C_2}) \tag{2.18}$$

With (μ_1, C_1) and (μ_2, C_2) the mean and covariance of the generated and real data respectively. The expression $Tr()$ is considered to be the trace linear algebra operation, which is the sum of elements along the diagonal of the square matrix. A smaller distance between the two distributions will result in a smaller FID and therefore a close resemblance of the generated distribution to the real distribution. Advantages of this metric are that it is able to capture the fidelity of the images well, it is computationally efficient, able to detect mode dropping and it is also consistent if increasing disturbances in the images are applied (Heusel et al., 2017). On the other hand, this technique is also dependent on the quality of the used classifier and the metric assumes a Gaussian distribution for the coding units. (Borji, 2018)

2.3.3 Number of Statistically-Different Bins

Richardson and Weiss (2018) have proposed another metric to measure the diversity of generated data and mode collapse. The metric is based on the idea that if two sample sets of the same distribution are arranged in bins, the number of samples that fall into such a bin should be the same. Suppose there exists a real distribution p with N_p samples and a generated distribution q with N_q samples. Every sample in distribution p can be described by s_i^p and every sample in distribution q is described by s_j^q. A bin B can be described by an indicator function $I_B(s)$. If a sample s falls into a certain bin B this results in $I_B(s) = 1$. If this does not fall into that bin, the bin will stay empty. If it is the case that $p = q$ then we can expect that

$$\frac{1}{N_p}\sum_i I_B(s_i^p) \approx \frac{1}{N_q}\sum_j I_B(s_j^q)$$

In other words, both distributions will fill the same bins B with the same amount of samples s. If this is not the case, the classic two-sample problem for Bernoulli variables can be used to determine whether the number of samples are statistically different for a given bin (Daniel and Cross, 2018). The z-score can be used as a testing statistic $z = \frac{P_p - P_q}{SE}$ in which P_p and P_q are the proportions of a sample that belong to a bin. The testing statistic also requires the standard error $SE = \sqrt{P(1-P)[1/N_p + 1/N_q]}$, which uses the pooled sample proportion P (the proportion that belongs to the bin when the sets are joined). This test can be performed on all bins and following the NDB can be calculated. Richardson and Weiss (2018) propose to use Voronoi cells to perform the binning, to ensure that every bin contains at least some samples. To do this, K-means clustering of the N_p training samples is realised. This will separate the N_p samples into an arbitrary K clusters ($K << N_p, N_q$), in which each sample s_i^p belongs to the cluster with the nearest mean. The L_2-distance is used to determine the nearest bin for the samples in N_q.

The main advantage of this metric is that it is applied directly on the image pixels and does not use deep-representations, like the other two described metrics do. This might make this metric sensitive to image properties that the other two metrics are not sensitive to. One of the biggest disadvantages of this technique is that the images are treated as vectors in pixel space, because Voronoi cells are used as bins. It is not known if using the L_2-distance in pixel space is meaningful. (Borji, 2018)

Document structure

In the next three chapters, the three sub-questions asked in the Introduction will be treated consecutively. All three chapters will begin with explaining the methods, followed by the results and will end with a small discussion. The thesis will then continue with a final discussion, after which a conclusion and recommendations will be given.

3 Generation of Images

This chapter is concerned with the first research question posed in the introduction: *In what way can images be created and how can the type of image and its difficulty be influenced?*. The chapter will be subdivided in four sections. The first chapter will explain what data is used and why. The second section will treat the type of network that will be used to accomplish the previously mentioned research question and will elaborate on how exactly this is implemented. The following section will show final results and the chapter will be ended with a small discussion.

3.1 Used Data

This thesis is interested in exploring to what extent CT-scan data of the chest can be simulated. In particular, lung nodules in the lungs. However, it is difficult for a non-expert to assess generated images in this area, since a CT-scan can be quite complicated. Identification of abnormalities would therefore be difficult, let alone the assessment of what kind of nodule is present. Therefore, at first another image data set is used, one that contains images of hands raising a number of fingers. Every possible situation, varying from zero to five fingers raised, is represented in the data set and therefore the data set can be considered as having six classes; Class '0', Class '1', Class '2', Class '3', Class '4' and Class '5'. If it is possible to influence the class of the image that is being generated, difficulty could also be influenced by assigning labels to different difficulty levels. In the case of lung nodules, this difficulty could be concerned with their location, "hiding places" for instance, or the type of nodule. In this example of the hands the difficulty is in that sense not adjusted, but it does investigate if it is possible to influence the kind of image generated. An example of each class of the fingers data set can be seen in Figure 3.1. It is assumed that the average observer is an "expert" in the assessment of the classes in this image data set, since the average observer should be familiar with the concepts of hands and fingers.

(a) Class '0' **(b)** Class '1' **(c)** Class '2' **(d)** Class '3' **(e)** Class '4' **(f)** Class '5'

Figure 3.1: Examples images from the *Fingers* image data set. Image (a) to (f) show an example image of Class '0' to '5' respectively. The classes correspond to the amount of fingers the hand raises in the image.

Every class in the image data set has right hands, as well as left hands. Also, the images show that there is variation in the hands that are represented. They are tilted under different angles. In Figure 3.2 six pictures are shown form one class, Class '2', to indicate the variation of images present for one particular class. In total, the data set consists of $21,600$ images of 128×128. The pixels of the hand are centred by the centre of mass and there is a noise pattern on the background. The images are normalised in the range between $[-1,1]$ before they are used. Before use and for visualisation, the images are in the range of $[0,255]$.

3.2 GAN architecture & Training Specifications

Now that an image data set is chosen of which the generated images will be interpretable, it is necessary to create a network in which the type of image to be created can be influenced. As has been explained in Chapter 2, using the cGAN architecture it is possible to also train the network on the class labels. Therefore, it is also possible to, after the training is finished, indicate which

Figure 3.2: The six images are examples of images from Class '2' of the image data set. It can be seen that the images show variety within the same class. The images show right, as well as left hands and from different angles. They do, however, always raise the same two fingers.

class image the generator should generate. If different difficulty levels could be included in the class labels, than it would also be possible to influence the level of difficulty.

The cGAN turns the objective function of eq. 2.11 described in Chapter 2 of the vanilla GAN into:

$$\min_G \max_D V(D,G) = \min_G \max_D \{\mathbb{E}_{x\sim p_{data}(x)}[logD(x|y)] + \mathbb{E}_{z\sim p_g(z)}[log(1-D(G(z|y)))]\} \quad (3.1)$$

As can be seen, the objective function is now also dependent on the class label y. To train the cGAN network, a separate generator network and discriminator network are necessary. The architectures for both the generator and the discriminator are based on the DC-GAN network, which was described in the previous chapter. The architectures are adjusted to generate and use the size of the images in the fingers data set. In the discriminator network, also Drop Out layers have been added, since they proved to have a positive effect on the performance of GANs in literature. The architecture for the generator network can be seen in Figure 3.3.

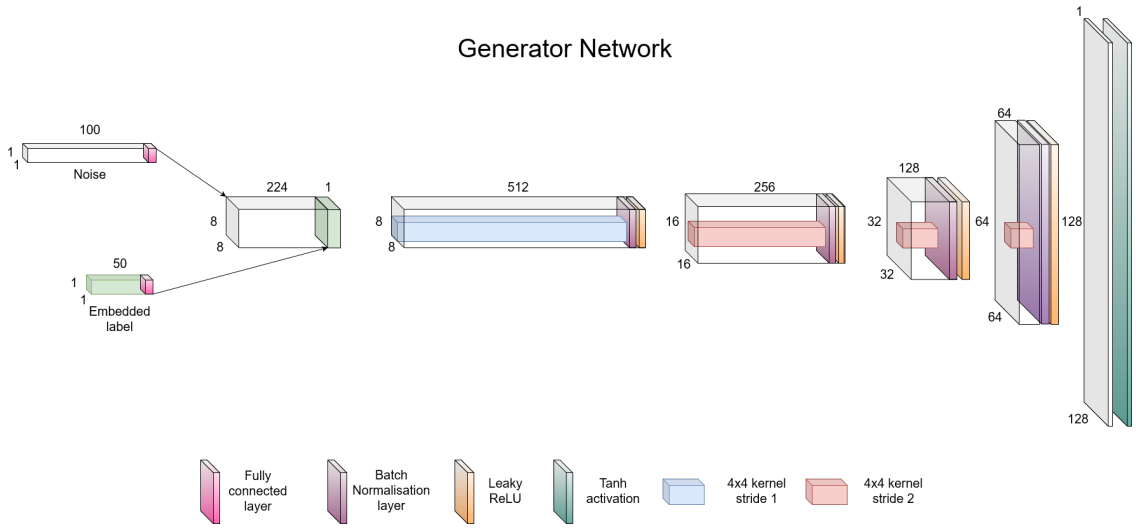


Figure 3.3: Network architecture of the used generator. It takes a noise vector of dimension 100 and an embedded label of dimension 50 as input. The network has five transposed convolutional layers, each followed by a batch normalisation layer and a leaky ReLU, except for the last transposed convolutional layer, which is followed by a hyperbolic tangent (tanh) activation layer. The network outputs an image of 128×128.

The generator takes as input a noise vector with a dimension of 100 and the label of the class to be generated. The noise vector is used as input for a fully connected layer which, after reshaping, turns it into 224 feature maps of 8×8. The label is embedded in vector with an arbitrary dimension of 50 and is also used as input for the fully connected layer. This layer, after reshaping, creates a feature map of 8×8 that is concatenated with the feature maps of the noise vector.

One transposed convolutional layer with stride 1 is used with 512 filters respectively. After this, three transposed convolutional layers with stride 2 are used to up-sample the feature maps to 16×16, 32×32 and 64×64. They use 256, 128 and 64 filters respectively. Each transposed convolutional layer is followed by a batch normalisation layer and a leaky ReLU layer. Finally, a transposed convolutional layer with stride 2 and 1 filter is used to create an image of 128×128, which is in turn passed through a hyperbolic tangent (tanh) activation layer.

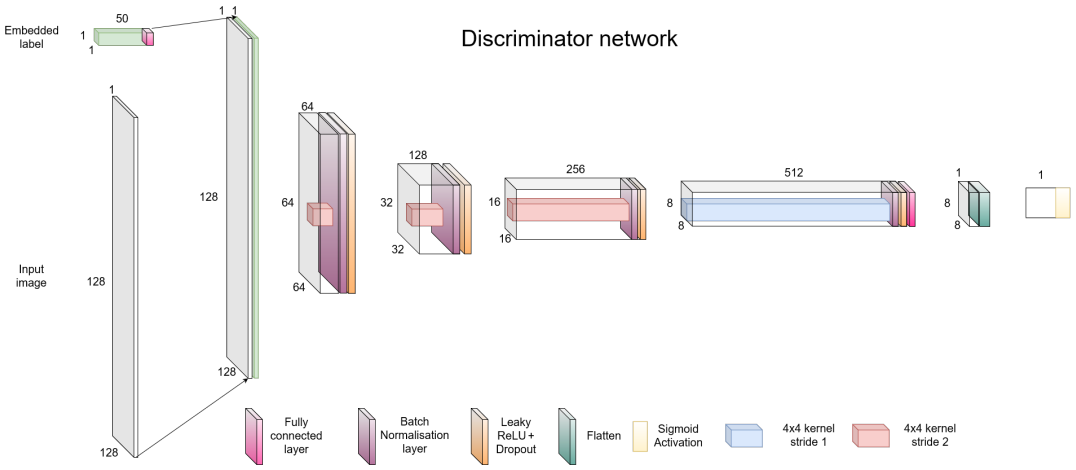


Figure 3.4: Network architecture of the used discriminator. It takes an image of 128×128, which can either be generated or real, and an embedded label of dimension 50 as input. The network has five convolutional layers, each followed by a batch normalisation layer and a leaky ReLU, except for the last convolutional layer, which is followed by a fully connected layer. Finally a 'sigmoid' activation layer is applied to give a probability as output.

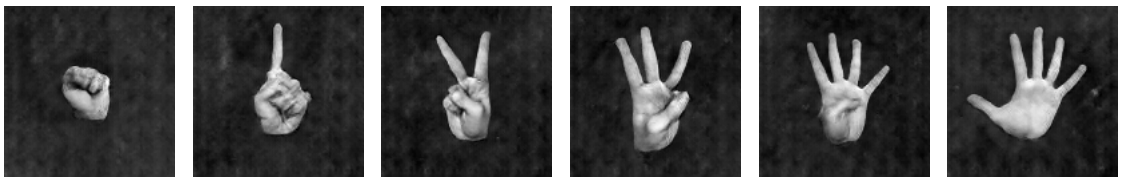
The architecture for the discriminator network can be seen in Figure 3.4. The discriminator takes as input the output of the generator, which is the 128×128 generated image, or a 128×128 image from the real image data set. This image is concatenated with the embedded label that was passed through a fully connected layer and was reshaped to match the size of the input image. Four convolutional layers are applied with stride 2 and 64, 128, 256 and 512 filters respectively. This results in down-sampled feature maps of 64×64, 32×32, 16×16 and 8×8. Each convolutional layer is followed by a batch normalisation layer and a leaky ReLU layer with a 0.5 Drop-Out layer. After this, one convolutional layer is applied with stride 1 and 1 filter. This finally results in one feature map of 8×8 which is flattened and passed trough a fully connected layer with a 'sigmoid' activation to get one probability output. This output is in the range of $[0, 1]$. If the discriminator outputs 0, it is very confident that the input image is fake. If it outputs 1, it is very confident that the input image is real.

The discriminator and generator are trained separately. To be more exact, the discriminator is trained and the entire GAN network is trained in which the weights and biases of the discriminator are held constant. This therefore corresponds to only updating the generator. The training process is similar to the process described in Algorithm 1 in Chapter 2. First the discriminator is trained on a half batch size of real data and than it is trained on a half batch size of generated data. Next, the generator is trained using one batch size. This is done with a batch size of 32 and for 50 epochs. Larger batch sizes required more memory which was not available. The network was trained for 50 epochs, because the network did not seem to improve significantly anymore after this. Also, the amount of time to train the network 50 epochs on one GPU ($14hours$) allowed for multiple tests within a reasonable period of time. With 21,600 images $\frac{21{,}600}{32} = 675$ updates are necessary to complete one epoch. The learning rate of both the discriminator and the generator is equal to 0.0002. Also, non-saturating binary cross-entropy loss and the Adam optimiser are used with $beta_1 = 0.5$. The choices of learning rates and optimiser were depend-

ent on literature and seemed to produce good results in this case as well. The code is written in Python and the Keras library is used with Tensorflow 2.0 as back end.

3.3 Results

After the training has been implemented and completed as described above, the generator is able to produce images as can be seen in Figure 3.5. The generator was asked to create one image of every class. When looking at these images, it can clearly be seen that each image looks like a hand. Also, the network seems to have captured the concept of each class correctly as well. However, when looking very closely to the images, it is possible to see some inconsistencies which indicate that the images are not real.



(a) Class '0' **(b)** Class '1' **(c)** Class '2' **(d)** Class '3' **(e)** Class '4' **(f)** Class '5'

Figure 3.5: Examples images from the generated image data set using the trained GAN. Image (a) to (f) show an example image of a generated image asked by the generator to produce Class '0' to '5' respectively.

It is also possible to look at the loss and accuracy after every epoch during the training of the GAN. In Figure 3.6 these are visualised in a graph. The left graph shows the loss of the generator and of the discriminator, separating the loss of training on a real image data and of generated image data for the discriminator. It can be seen that the loss of the generator shows a decreasing trend and that the losses of the discriminator keep the same trend.

In the right graph the accuracy of the generator and of the discriminator can be seen, which shows the amount of correctly classified labels. Also in this case, the accuracy of the discriminator is divided in the situation when real images are used and when generated images are used. It can be seen that the accuracy of the discriminator in both cases fluctuates around 0.8 which is to be expected since the loss has not reached 0 yet. Next to this, it can be seen that the accuracy of the generator is really low. This can be explained by the fact that when the generator is trained, the corresponding label with a generated image is set to 1, which is the label to indicate that an image is real. This is done to prevent the gradient from saturating as explained in Chapter 2 about the objective function of the GAN. However, this results in a low accuracy, because the discriminator will classify the image mostly as being fake or 0. Also, it is possible to see that the accuracy is slightly increasing at the end, which corresponds to the decreasing loss in the left graph. This may indicate that the generator was still improving. The interpretation of these graphs are, however, not always intuitive.

Even though it is possible to qualitatively judge the generated images by comparing them to the real images with the human eye, this does not provide quantitative assessment of the performance of the network. The assessment can be quantified in this, but this would require assessment by human observers. Quantitative assessment in this way is very time intensive. Next to this, the loss and accuracy do not provide a clear assessment, since the interpretation of them is not always intuitive. In the next section, it will be evaluated how the performance of the network can be more quantitatively assessed.

3.4 Discussion

There are several aspects that need to be considered during the discussion about this chapter. Even though the results seems satisfactory, not every parameter has been changed to inspect

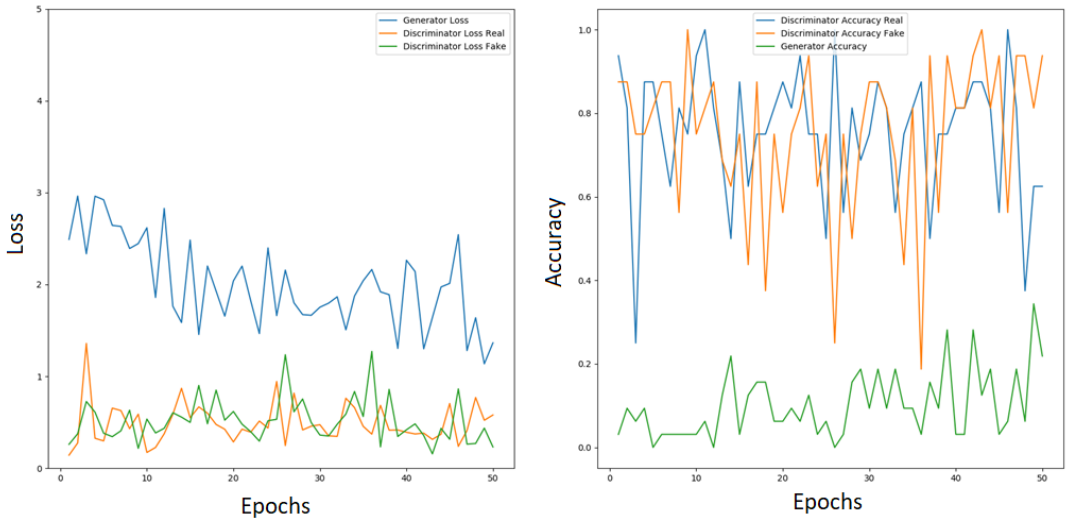


Figure 3.6: Graphs of loss and accuracy of both generator and discriminator. The left graph shows the loss of the generator and the discriminator after every epoch. The loss of the discriminator is divided in the loss after updating with a real image batch and after updating with a generated image batch. The y-axis shows the loss and the x-axis shows number of epochs. The right graph shows the accuracy of the generator and the discriminator after every epoch. In this case also, the accuracy of the discriminator is divided in the accuracy after updating with a real image batch and after updating with a generated image batch. The y-axis shows the accuracy and the x-axis shows the number of epochs.

the influence on the outcome. To get to this result, various parameters have been changed to see their effect; batch size, the learning rate of the discriminator and the generator, amount of epochs, and the size of the noise vector that the generator takes as input. Besides this, other parameters may have been changed as well; optimiser function and their accompanying terms (like $beta_1$ for the Adam optimiser), only the non-saturating loss function has been used and no changes have been applied to the training ratio of the discriminator and the generator. Finally, the architecture of both the generator and discriminator have not been changed. Therefore, the effect of depth and the effect of some layers, like the DropOut layer, has not been explored.

4 Metric Comparison for Quality Assessment

This chapter will be concerned with the second research question posed in the Introduction: *How can the quality of the generated images be quantitatively assessed and does this correspond to human judgement of the quality?* In order to assess the quality of the generated images, it is necessary to know when a generated image is considered as "good" or "bad" and when it is considered as "better than" the other. Since the desired result is used for visual assessment by humans, human judgement should also be considered as the golden standard of assessing the generated images. However, it is time and energy intensive to get an expert opinion on the generated images every time the GAN is trained somewhat differently. Especially if the expert is not very common and has little time to perform such tasks. Therefore, the same image data set as in the previous chapter will be used. This ensures a larger expert pool, compared to immediately working with lung nodules in CT-scans. A questionnaire is set out to gather the average human judgement on generated images of differently trained GANs and these results are compared with three different quantitative metrics. This should give some insight into which quantitative metrics most accurately resemble the human judgement.

In the first section of this chapter it will be explained how differently performing networks are realised to create generated images of differing quality. In the second section it will be explained how the average human judgement on the generated image sets is gathered. The third section will explain how this average human judgement is correlated with three different quantitative metrics. In the following section results will be shown and the chapter will be ended with a short discussion.

4.1 Generating Image Sets

As well as the same image data set, also the same network will be used as described in Chapter 3. To get differently performing GANs, several training specifications are adjusted. The GAN architecture however, is kept the same. Five different training specifications are considered, with which five differently performing GANs and therefore also five generated image data sets can be created. Next to these five, also the original image data set is considered, Image set 1. In Table 4.1 the varying training specifications for the generated image data sets can be seen. Image set 1 is left out in this Table, since it is the real image data set and therefore no GAN had to be trained to obtain it. In every scenario the batch size was set to 32 and the Adam optimiser was used with $\beta_1 = 0.5$.

Table 4.1: Five generated image sets have been obtained from five differently performing GANs by adjusting the training specifications. The changed parameters to train the GAN can be seen in this Table.

Image set	Learning rate D	Learning rate G	Epochs	Latent Dimension
Image set 2	0.0002	0.0006	50	100
Image set 3	0.0002	0.0002	50	100
Image set 4	0.0002	0.0002	50	1
Image set 5	0.0006	0.0002	50	100
Image set 6	0.0002	0.0006	5	100

4.2 Human Judgement Assembly

To find out which generated image data set is considered as best and which as worst, a questionnaire will be put together. This questionnaire will exist of three different parts. In the first

part, it will be considered if the generated images are classified as the same class the generator was asked to create, i.e. if the generator is asked to create Class '1', the image should also look like Class '1'. This is an important criteria, since the goal is to be able to influence the type of image that is created. In the second part the variety of images is assessed. Besides creating realistic images, it is also preferable that the generator is able to create multiple images of the same class that are not exactly the same. If it generates the same image every time it is asked to create a certain class, this would not be very useful. In the third part, the images of every image data set will be ranked according to how realistic they look, since it should also be assessed which image looks the best according to the visual perception of human judges. Each part will be elaborated further below.

4.2.1 Classification

Ten images of each image data set are given to classify, i.e. observers are asked to select which class the images represent. This results in a total of 60 images being classified. The images are chosen at random and therefore it is possible to get multiple images of Class '3', whereas only one image of Class '1'. To increase the amount of images per data set being classified, the participants are divided into five different groups. Each group is given ten different images from the image data sets. This results in a total of 50 images per data set that are classified.

4.2.2 Variety

To assess the variety in a certain image data set, participants will be asked to give a score between $1-4$. They are given six images from one particular class from every image data set. If the participants thinks the six images show high variety, it will be given a 4 and if it shows low variety it will be given a 1. The five groups, as described before, are each given a different class to judge. This means that one class will not be assessed, Class '5'. It is, however, assumed that the other classes will give a generalised opinion about the whole image data set.

4.2.3 Ranking

In the last part of the questionnaire, participants will be asked to rank six pictures, each from a different image data set but from the same class. They will be asked to do so for every class. A 1 will be given to the image that is most realistic and a 6 will be given to the image that appears least realistic. All five groups will receive the same images for this part.

4.3 Correlation of Metrics & Human Judgement

This section describes which metrics are used and how they are applied. After this it will described how these will be correlated with human judgement.

4.3.1 Metrics used

The metrics described in Chapter 2 will be used to quantitatively assess the image data sets; the Frèchet Inception Distance (FID, the Gan Quality Index (GQI) and the Number of statistically Different Bins (NDB). These three were chosen, because they meet the criteria of an efficient GAN metric to varying degrees. Table 2.1 in Chapter 2 described the FID as being a good metric in general, but as having a moderate assessment on diversity of images. The NDB metric in turn has a high assessment on the diversity, but lacks good assessment of fidelity and distortions. The GQI discern itself from the other two, because it has both lower and upper bounds.

For both the FID and the GQI it is necessary to create a CNN classifier trained on the real image data set. Also, for the GQI it is also necessary to create a classifier trained on each generated image data set. From the real image data set, Image set 1, $10,000$ images are taken as training set and $3,600$ images for the test set. Also, each generated image data set consists of $10,000$ images. For all image data sets the same architecture is used which can be seen in Appendix E.

The classifier[1] is trained for 5 epochs and with a batch size of 12. A larger batch size was computationally too intensive for the used hard- and software. After the 5 epochs the performance did not improve anymore.

4.3.2 Correlation

The three metrics and outcome of the questionnaire can not directly be compared since they are all measured on a different scale. It is possible however, to normalise their score on a scale from 1 to 10. The image data set with the best score will get a score of 10 and the lowest score will correspond to a score of 1. In this way, it is easier to compare the assigned scores. However, when comparing this way, it is only possible to compare the ranking of each image data set. Since the outcomes of the metrics all have a different range, the normalised scores can not be compared directly.

These rankings can also be used to determine a correlation coefficient between the ranking of human judgement (HJ) and a ranking of one of the metrics. Specifically, the 'Kendall Tau Rank Correlation Coefficient' can be used (Kendall, 1938). It is defined as:

$$\tau(\sigma_1,\sigma_2) = \frac{2}{n(n-1)} \sum_{i,j \in S \wedge i<j} sign(\sigma_1(i)-\sigma_1(j)) * sign(\sigma_2(i)-\sigma_2(j)) \tag{4.1}$$

With σ_1 and σ_2 two different permutations with length n, which are both assumed to be a permutation of the integers in a set $S = 1,2,...,n$. In this case σ will correspond to the ranking of the image data sets according to a metric. Therefore, $\sigma(i)$ will represent the rank of image data set i in that ranking. The outcome τ is scaled between $[-1,1]$ with 1 occurring when $\sigma_1 = \sigma_2$ and -1 occurring when σ_2 is the reverse of σ_1. The correlation coefficient can also be define as in eq. 4.2, where C corresponds to the amount of concordant pairs and D corresponds to the amount of discordant pairs.

$$\tau(\sigma_1,\sigma_2) = \frac{2}{n(n-1)}(|C|-|D|) \tag{4.2}$$

The Kendall Tau distance is derived from the Kendall Tau Rank Correlation Coefficient and is equal to the number of discordant pairs, $K(\sigma_1,\sigma_2) = |D|$. It is also possible to normalise this value which results in the following equation:

$$K(\sigma_1,\sigma_2) = \frac{2|D|}{n(n-1)} \tag{4.3}$$

In this case $K(\sigma_1,\sigma_2) = 0$ only when $\sigma_1 = \sigma_2$, because K lies in the interval of $[0,1]$. When σ_2 is the reverse of σ_1 the maximum of 1 occurs.

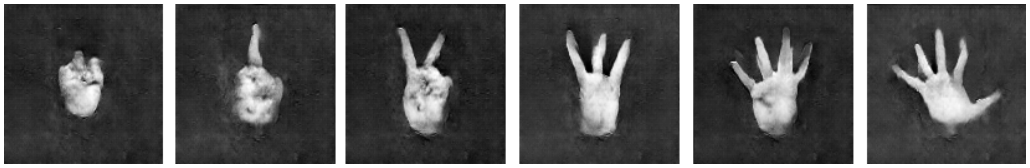
4.4 Results

To give an indication of what the images from the five different generated image data sets look like, an example of each class per image data set is shown in Figure 4.1. It can be seen that for instance Image set 5 and Image set 3 look far more realistic than Image set 6. Next to this, it can be seen that not every Image set was able to capture the semantic features of each class. In Image set 2 an image of Class '2' also appears as an image of Class '1'. In Image set 6 Class '2' and Class '4' are also very unclear.

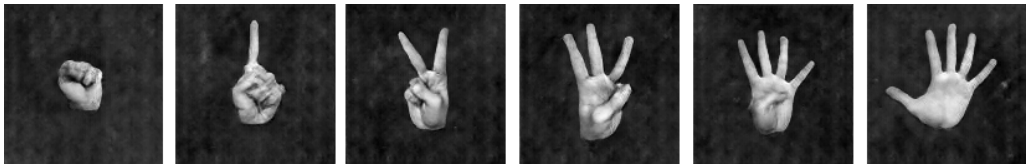
[1] The architecture was taken from https://www.kaggle.com/nassimyagoub/cnn-finger-counter-100-accuracy

(a) Image set 2

(b) Image set 3



(c) Image set 4



(d) Image set 5

(e) Image set 6

Figure 4.1: Examples of each Class in each generated image data set. From left to right there is an example image from Class '0' to Class '5' and six images from (a) to (e) correspond to image data set 2 to 6 respectively.

4.4.1 Questionnaire

In total there were 57 respondents to the questionnaire. The average results will be discussed on the basis of every part in the questionnaire.

Classification

The images that are used for classification are similar to the images that can be seen in Figure 4.1. The 57 respondents have each classified 10 images from each image data set. In Table 4.2 the amount of correctly classified image as are shown as well as the amount of wrongly classified images and the resultant accuracy. In the last column, the resulting ranking according to the accuracy can be seen. As would be expected, the real image data set has achieved the highest accuracy. Next to this, it would also be expected that Image set 2 and Image set 6 would be ranked quite low, since they had some difficulty with capturing the right semantic features for each class.

Variance

In Figure 4.2 six images from each generated image data set of the same class, Class '2', are shown. This can give an indication of the degree of variance in each image data set. It can

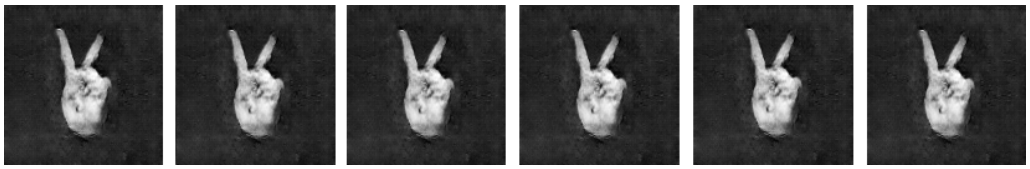
Table 4.2: Results of the classification of the images in each image set in the questionnaire. The amount of correctly classified, as well as wrongly classified images from each image data set are shown. In the last to columns the corresponding accuracy is given and the ranking according to the accuracy.

Image set	Correctly classified	Wrongly classified	Accuracy	Ranking
Image set 1 (Real)	565	5	99.12%	1
Image set 2	525	45	92.11%	5
Image set 3	533	37	93.51%	4
Image set 4	555	15	97.37%	3
Image set 5	562	8	98.60%	2
Image set 6	361	209	63.33%	6

already be seen that the images from Image set 2 and Image set 3 do not show a high variance. Figure 4.2 is an example of what one group received to judge, together with six images from Image set 1 (Figure 3.2). Each group assessed the variance on one class, therefore, each class was assessed except for one.

(a) Image set 2

(b) Image set 3



(c) Image set 4

(d) Image set 5

(e) Image set 6

Figure 4.2: Six examples of images from Class '2' in each generated image set to give an indication of the variety the generator can produce. The images from (a) to (e) correspond to images from image set 2 to image set 6 respectively.

The results within one group per class were averaged, followed by averaging the results for each class over all groups. This resulted in one final average judged variance, which can be seen in Table 4.3. Also in this case, the image data sets were ranked according to the final score on this part.

Table 4.3: Result of the average judged variance and the standard deviation of the images in each image set. The last column shows the rank according to this average.

Image set	Average Judged Variance	Standard Deviation	Ranking
Image set 1 (Real)	2.82	0.73	3
Image set 2	1.07	0.41	6
Image set 3	2.87	0.74	1
Image set 4	1.18	0.59	5
Image set 5	2.84	0.79	2
Image set 6	2.10	1.09	4

As expected Image set 2 and Image set 4 are ranked at the bottom. Interesting to see is that the real image data set, Image set 1, is ranked in third place. However, as can be seen, the values are very close to one another. If the standard deviation is taken into consideration these three image sets might be considered to have the same rank.

Ranking

In the final part, participants were asked to rank images of the same class from different Image data sets. In Figure 4.3 an example can be seen for Class '2'. In this case Picture 1, Picture 3 and Picture 4 do not seem to show finer features, whereas Picture 2, Picture 5 and Picture 6 do. This would give an indication that Image set 2, Image set 4 and Image set 6 will be considered as having less realistic images.

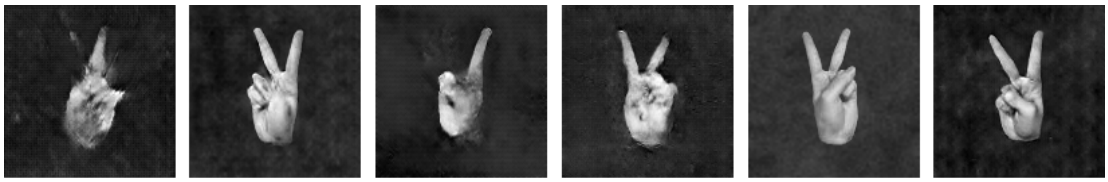


Figure 4.3: Example of images from each image data set of Class '2'. Picture 1 = Image set 6, Picture 2 = Image set 3, Picture 3 = Image set 2, Picture 4 = Image set 4, Picture 5 = Image set 1 and Picture 6 = Image set 5.

Each participant, regardless of group, had to rank these images for each class. This would result in an average position for each image data set according to each class. This is in turn averaged to obtain one average judged rank for each image data set. In Table 4.4 the results can be found.

As expected, it can be seen that Image set 2, Image set 4 and Image set 6 are ranked lower. Also, as would be expected, Image set 1 is ranked highest. However, when taking the standard deviation into account, especially the three lowest ranking image sets to not differ that much in their judgement.

Total

The ranking from each different assessment, classification, variance and ranking, are shown together in Table 4.5. Also, these three rankings are averaged and this results in one final ranking according to human judgement. As was to be expected, the real image data set is ranked in

Table 4.4: Result of the final average ranking of each image data set (+ standard deviation) according to human judgement. The last column shows the ranking on this part.

Image set	Average Judged Rank	Standard Deviation	Ranking
Image set 1 (Real)	1.47	0.69	1
Image set 2	4.61	0.79	4
Image set 3	2.61	0.80	3
Image set 4	4.82	0.84	5
Image set 5	2.08	0.86	2
Image set 6	5.41	0.87	6

the highest position. Furthermore, it can be seen that Image set 5 seems to contain the images which are considered to be most realistic. The images in Image set 6 are considered to be least realistic.

Table 4.5: Final Ranking of human judgement of each of the image data sets. The ranking of each of the three different assessments are shown, classification, variety and ranking, as well as the average of these three. The last column shows the final ranking according to the average.

Image set	Classification	Variety	Ranking	Average	Final Ranking
Image set 1 (Real)	1	3	1	1.7	1
Image set 2	5	6	4	5.0	5
Image set 3	4	1	3	2.7	3
Image set 4	3	5	5	4.3	4
Image set 5	2	2	2	2.0	2
Image set 6	6	4	6	5.3	6

4.4.2 Metrics correlation

In Table 4.6 the results of applying the three different metrics, FID, GQI and NDB, can be seen. The first thing that can be seen is that every metric has a different final ranking according to the calculated scores. Also, every metric scores perfectly on the real image data set, as would be expected. In Appendix D also a small experiment can be found in which the metrics are also applied to the real images test set and to Image set 3 (half data). In the latter case the GAN is trained under the same circumstances as in the case of Image set 3, only half the size of the training data is used.

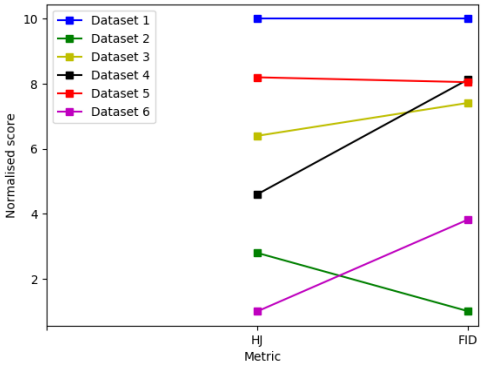
The scores for each metric and the ranking of human judgement are normalised onto a scale of $1-10$. These results are shown in Figure 4.4. In Figure 4.4a the normalised score of the FID metric compared to HJ is visualised. It can be seen that the scores of the metric and HJ do not results in the same ranking of the image data sets. In Figure 4.4b the same can be seen for the GQI metric. The ranking according to the scores seems to be more similar when compared to the FID and only two image data sets are switched. In Figure 4.4c it can also be seen that the ranking according to the scores of NDB does not correspond to the ranking of HJ, but it seems to correspond better than the FID.

In Table 4.7 the calculated Kendall Tau Rank Correlation and Kendall Tau distance can be seen for each of the metrics compared to HJ. From these results it can be seen that the FID performs worst of the three metrics when compared to HJ. The GQI and NDB recieve almost the same

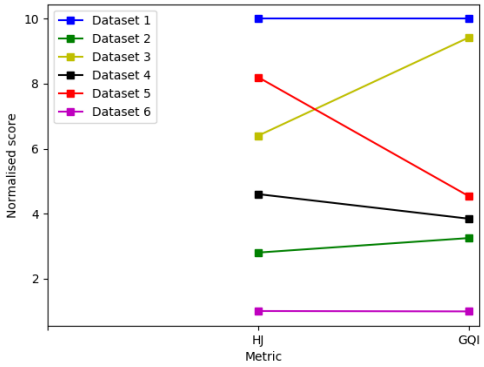
Table 4.6: Scores of each image data set according to the three different metrics: FID, GQI & NDB. Each metric shows the score that was obtained with the image set, followed by the corresponding rank according to all the scores.

Image set	FID (score - rank)	GQI (score - rank)	NDB (score - rank)
Image set 1 (Real)	0.000 - 1	100.0 - 1	0 - 1
Image set 2	569.060 - 6	53.750 - 5	20 - 5
Image set 3	163.819 - 4	96.056 - 2	12 - 3
Image set 4	118.381 - 2	57.833 - 4	20 - 5
Image set 5	123.352 - 3	62.583 - 3	8 - 2
Image set 6	391.081 - 5	39.278 - 6	19 - 4

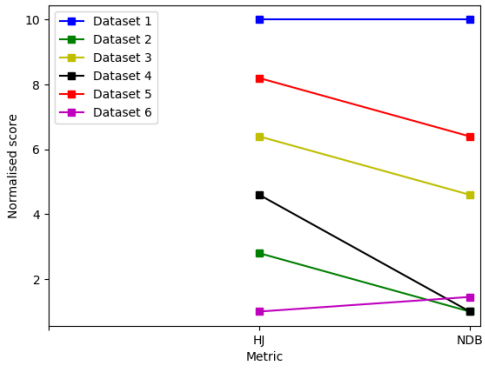
scores, only the Kendall Tau distance is in the case of the NDB a little higher. This is because, in the NDB scores, there is a tie present and, when calculating the Kendall Tau distance, this has to be compensated for.



(a) Correlation FID with HJ



(b) Correlation GQI with HJ



(c) Correlation NDB with HJ

Figure 4.4: Correlation of the three different metrics with human judgement. Subfigure (a) to (c) show graphs of the correlation with metric FID, GQI and NDB respectively. In each of the graphs the y-axis shows the normalised score and the x-axis shows the metric. The different colours represent the different Image data sets.

Table 4.7: Kendall Tau rank correlation coefficient and Kendall Tau distance for all metrics compared to human judgement.

Correlated Metric	Coefficient	Distance (normalised)
HJ vs. FID	-0.2	0.6
HJ vs. GQI	0.6	0.2
HJ vs. NDB	0.6	0.23

4.5 Discussion

The discussion will be covered in three parts; human judgement assessment, metric assessment and the correlation of those. To begin with, it is very difficult to get a reliable human judgement on images. It has not been looked into if the amount of images that have been assessed and the amount of respondents are enough to make a reliable conclusion. Next to this, human judgement is also dependent on the expertise and motivation of the person at task.

Considering the first part of the questionnaire, the classification, the classes of the images that were used, were generated randomly. Therefore, certain classes might have been represented more than others for a given image data set. If the GAN had more difficulty with a class that was represented a lot, this can have an unfair influence on the final accuracy of the classification. Regarding the second part of the questionnaire, the average variance had quite high standard deviations. This, in combination with the question about the number of respondents, raises the question what happens to this average if for instance 10 more respondents would have filled in the questionnaire. This same might be asked for the the third part of the questionnaire, the ranking. Next to this, only one image of each class is used for ranking. If the variance is high, then it might also be possible that one image is more realistic than another one. Finally, to get to the final ranking it is assumed that every part of the questionnaire has an equal contribution. It might be considered that, for instance, one part might be more important than the other. This can also be dependent on the goal the GAN is designed for.

When looking at the application of the three metrics also some discussion points arise. Both the FID and GQI are dependent on the classifier that is being used. Although in this case the classifier seems to perform very good, in another scenario where different data and therefore a different classifier are being used, this ranking might not correspond in the same way to human judgement as it does now. This also has a relation to how much each class is represented in the training set. In this case it appears that the classifier was able to learn particular features of each class quite well, since the classifier performs very good. However in another scenario, where the distribution of classes is not equal, the classifier might not perform as good on each class. This directly affects the performance of the FID and GQI metric.

The NDB seems to correlate quite well with human judgement in this case and can be applied directly to the generated images. However, the NDB is very prone to the amount of images that represent certain classes in the generated image data set. One class should be presented as much in the generated image data set as it is presented in the real image data set. This makes the NDB metric very suitable to establish mode dropping, since one class, or multiple, would then be absent in the generated image data set. However, if in the generated image data set one class is present with a lot more images, compared to the class in the real image data set, this also greatly influences the outcome of the NDB metric. This raises the question if a high variance in each class of the generated image data set, which would be preferable, also influences this metric in a negative way.

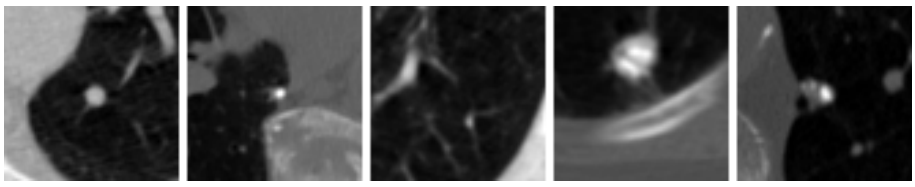
When correlating the metrics it is not taken into account how close each data set was ranked to the next. As mentioned before, in some cases a small change in tested data set or observer opinion might have given a different ranking and therefore another outcome. It can also be questioned if it is fair to use this correlation between the metrics and human judgement in other scenarios. All the same, it is interesting to see that the FID seems to correlate worse than the other two metrics, as the FID is said to be the best metric thus far in literature. Especially when correlated with human judgement. Finally, only three metrics are considered whereas there are a lot more that could have been tested.

5 Generation & Assessment of Lung Nodules

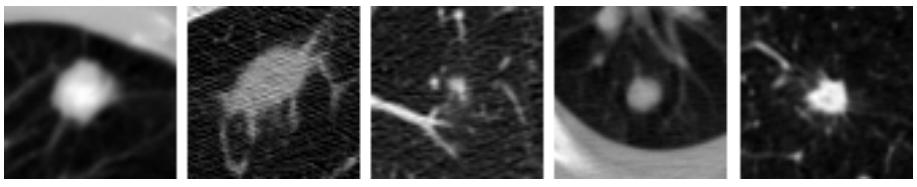
This chapter will be concerned with the final sub-question posed in the introduction: *To what degree can the nodules be recognised and classified using an algorithm and/or an experts opinion?* Using the knowledge of the previous chapters it will hopefully be easier to train a GAN architecture to generate lung nodules and to evaluate the quality of these images. In the first section it will be explained which data is used and what the GAN architecture looked like together with its training specifications. In the second section it will be explained in what way the results will be evaluated, followed by the section which shows the results. Finally, in section four a small discussion will take place.

5.1 Used Data & GAN training

To train the GAN network, the LIDC-IDRI database is used (Armato III et al., 2015; McLennan et al., 2011; Clark et al., 2013). This Lung Image Database Consortium consists of 1,018 cases of CT-scans with marked annotated nodules of diagnostic and lung cancer screening. The database is created with the help of seven academic centres an eight medical imaging companies. Because of this not every case is acquired using the same equipment, slice thickness, etc. Most cases are evaluated by four radiologists. As such, information is available on if there is a nodule present, in which slice and on what position, but also what the opinion is about this nodule. Unfortunately, it was not indicated as to what category the nodule belongs, i.e. a juxtavascular nodule or well-circumscribed nodule. Only the degree of malignancy was indicated and from this it could be concluded if the nodule was considered to be benign or malignant. Using this information nodules were cut out of their slices in a 64×64 image and it was noted whether the nodule was found to be benign or malignant. This resulted in a total of 2,638 images with a nodule present of which 2,097 are considered benign and 541 malignant. The images in the data set were assigned one of two possible classes; benign or malignant. The images are normalised in the range between $[-1,1]$ before they are used. It would have been preferable if the images could have been used with Hounsfield Units instead of pixel values. However, information about this was not found in the LIDC-IDRI data set. Because this information could not be recovered, it was decided to use the $[-1,1]$ range since this is recommended in literature and worked well in the previous chapter. In Figure 5.1 some examples of the used nodules can be seen. Pixel values have been scaled to the range of $[0,255]$ to visualise the images. Figure 5.1a shows five examples of benign nodules and Figure 5.1b shows five examples of malignant nodules.



(a) Examples of benign nodules



(b) Examples of malignant nodules

Figure 5.1: Examples of nodules extracted from the LIDC-IDRI database. In subfigure (a) benign nodules are shown and in subfigure (b) malignant nodules are shown.

Also, in this case a cGAN has been used as architecture. The architecture of both the generator and the discriminator is quite similar to the ones explained in Chapter 3. The architecture of the generator differs on two aspects. The generator needs to output images of 64×64, since the images from the real data also have this size. To achieve this, the second transposed convolutional layer is changed from having stride 2 to stride 1. This ensures that up-sampling is performed until the image size of 64×64 is reached instead of 128×128. The adjusted architecture of the generator can be seen in Figure 5.2.

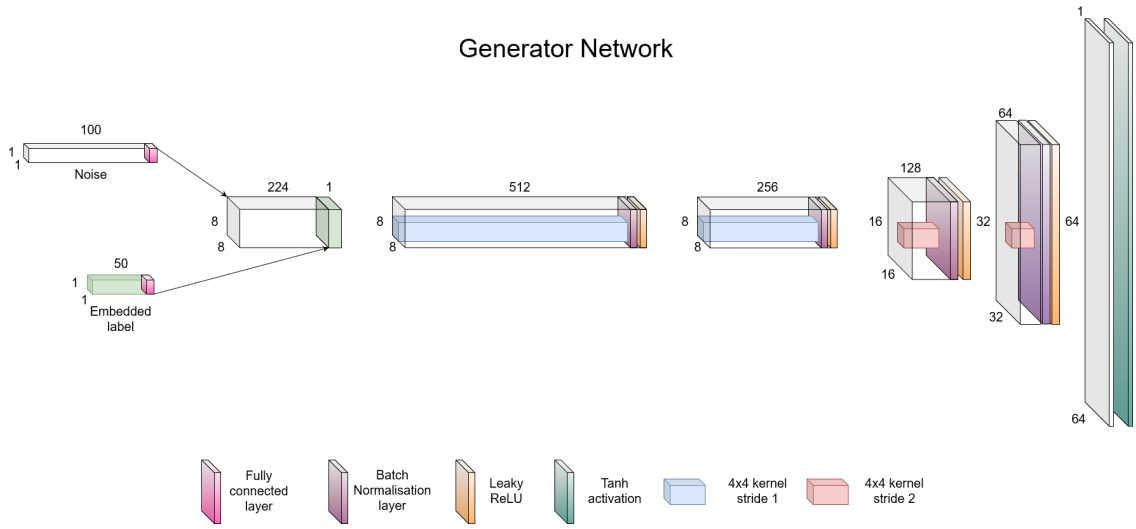


Figure 5.2: Network architecture of the used generator. It takes a noise vector of latent dimension 100 and an embedded label of dimension 50 as input. The network has five transposed convolutional layers, each followed by a batch normalisation layer and a leaky ReLU, except for the last transposed convolutional layer, which is followed by a hyperbolic tangent (tanh) activation layer. The network outputs an image of 64×64.

The architecture of the discriminator is also adjusted on two aspects. The input size is adjusted to 64×64 and also one convolutional layer is adjusted to have stride 1 instead of 2. This is done to ensure that the generator and discriminator have a similar architecture. The adjusted architecture of the discriminator can be seen in Figure 5.3.

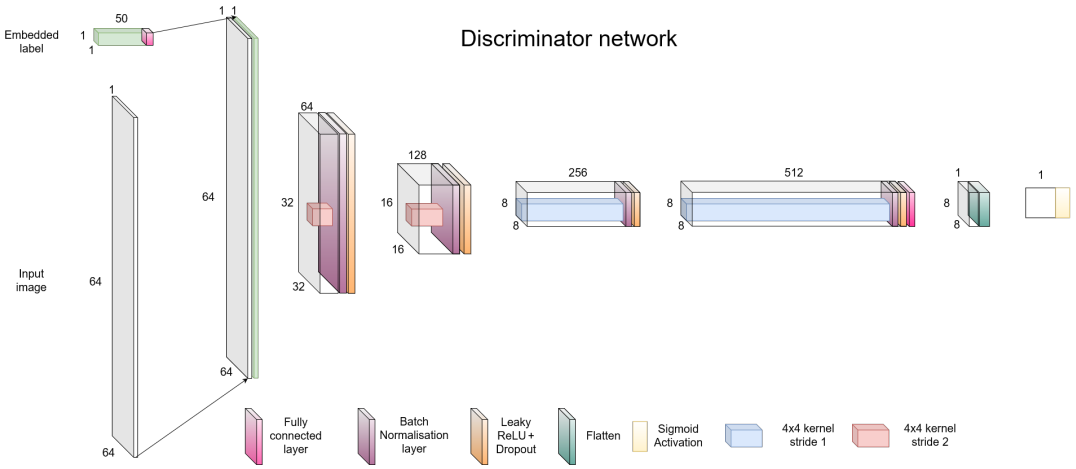


Figure 5.3: Network architecture of the used discriminator. It takes an image of 64×64, which can either be generated or real, and an embedded label of dimension 50 as input. The network has five convolutional layers, each followed by a batch normalisation layer and a leaky ReLU, except for the last convolutional layer, which is followed by a fully connected layer. Finally a 'sigmoid' activation layer is applied to give a probability as output.

The training process is similar to the process described in Algorithm 1 in Chapter 2. A batch size of 32 is used and the network is trained for 10,000 epochs. The amount of epochs has been increased compared to the previous chapter, because the image data set is much smaller. Next to this, training on nodules posed to be more difficult and realistic images were only created after training for more epochs. The learning rate of both the generator and the discriminator is equal to 0.0002. Both networks use binary cross-entropy loss and the Adam optimiser with $beta_1 = 0.5$. Next to this, one-sided label smoothing is added for real images. This changed their label from 1 to 0.9 and is an attempt to make sure the discriminator does not overpower the generator.

5.2 Evaluation of generated nodules

To evaluate the generated lung nodules, two different types of assessment will be used. The metrics described in the previous chapter will be used; FID, GQI and NDB. All three metrics will be applied, but the knowledge of their correlation with human judgement can now be used to interpret the metric scores. Next to this, the generated images will be tested by a separate nodule classification system, which is trained on (real) nodule CT-scan data. If the generated images are realistic enough, the classification system would assign the same probabilities on average to the images. However, it is expected that the probabilities that are assigned to the generated nodules will be more weak. As independent classifier, the ResNet50 network is used, which is more often used for image classification purposes (He et al., 2016). The network architecture can be seen in Appendix E.

As mentioned earlier, 2,638 images of nodules were obtained of which 2,097 are benign and 541 are malignant. For training of the classifier 2,110 images were used of which 1,710 were benign and 400 were malignant. To balance this ratio between malignant and benign, the malignant nodules have been flipped horizontal and flipped vertical to extend the number of malignat nodules to 1,200. This results in a total amount of 2,910 nodules. Of this training set 20% was used as validation set. The generated training nodule data set was created to have the same ratio of malignant and benign and the same amount of nodules. This left the real test set with 528 nodules of which 387 benign and 141 malignant. Also the generated nodules test set was created using the same ratio and amount of nodules. The architecture was in both cases trained for 10 epochs and a batch size of 32, which gave the best results in terms of accuracy. When trained longer, the network would only increase its accuracy through overfitting. Therefore the training was stopped after 10 epochs. To compare the probabilities of the classification of the real and generated images, the probabilities will be plotted in a histogram and a density plot.

5.3 Results

In Figure 5.4 some examples can be seen of generated lung nodules, both benign and malignant. It appears as if they are part of a CT-scan slice of lung nodule, however, they are not entirely realistic. Figure 5.1a shows five examples of generated benign lung nodules. It can be seen that nodules are present in both image 1,2 and 4. However, in image 3 it seems to be absent and in image 5 it is almost not visible. Figure 5.1b shows five examples of generated malignant lung nodules. In image 1, 2 and 5 also nodule like spots can be seen. These three images also resemble each other. Image 3 however, does not seem to make much sense. Also, image 4 seems to lack a nodule like spot. Also the variance in the images is not very high, it is especially low for the malignant category. A better example of this can be seen in Appendix E Section E.

In Figure 5.5 the changing loss function over the 10,000 epochs can be seen. The loss of the discriminator is separated into the loss when trained on real data and the loss when trained on generated data. It can be seen that both the generator and discriminator loss have a sudden drop around epoch 7,000, after which they increase again. It seems as if the GAN has gone into

(a) Examples of generated benign nodules

(b) Examples of generated malignant nodules

Figure 5.4: Examples of nodules generated nodules using a GAN. In subfigure (a) generated benign nodules are shown and in subfigure (b) generated malignant nodules are shown.

a failure mode, which may be caused by a discriminator that is too strong or a generator that is outputting garbage (or both). However, it also seems to recover again.

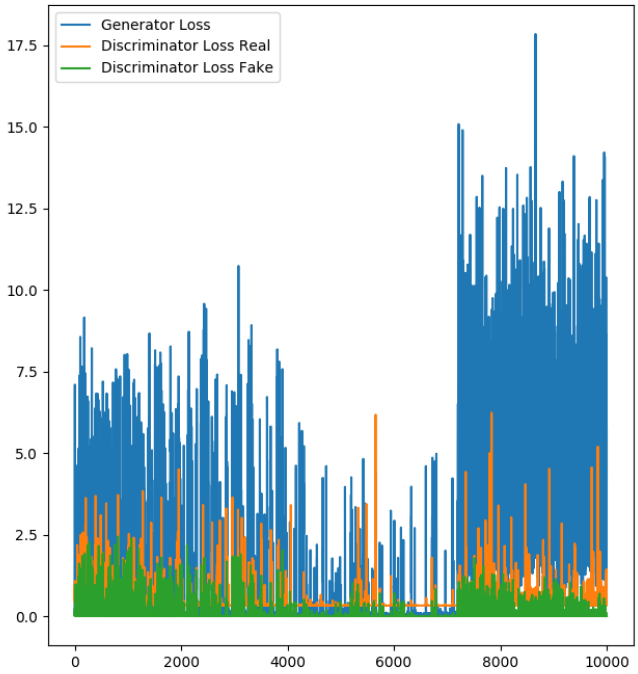


Figure 5.5: Graph of loss of both generator and discriminator. The graph shows the loss of the generator and the discriminator after every epoch. The loss of the discriminator is divided in the loss after updating with a real image batch and after updating with a generated image batch. The y-axis shows the loss and the x-axis shows number of epochs.

In Table 5.1 the results of the three metrics applied on the generated data set can be seen. As explained before, the FID ideally reaches 0, the GQI reaches 100 and the NDB also ideally should reach 0. If this is the case, the generated image data distribution resembles the real data distribution. In Appendix D a small experiment can be found, that shows the NDB metric is not very robust.

The results of the FID and GQI are also dependent on the classifier that is used. The classifier trained on real data reached an accuracy of 70% and the classifier trained on the generated

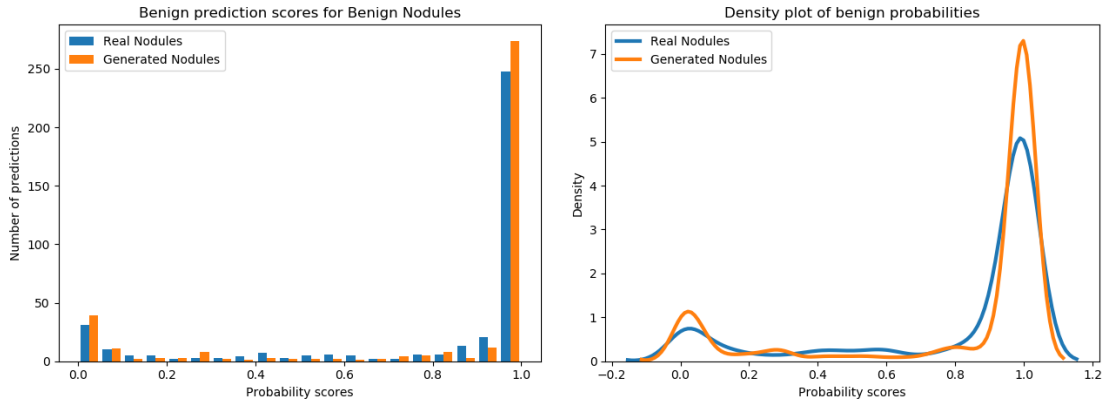
Table 5.1: Scores of each image data set according to the three different metrics: FID, GQI & NDB. Each metric shows the score that was obtained with the image set, followed by the corresponding rank according to all the scores.

	FID	GQI	NDB
Generated Nodules	0.016	105.18	4

data reached an accuracy of 73%. Both the accuracies were tested on the real nodule test set. In Appendix E Section E the accuracies and loss over the epochs can be seen. The classifiers were not trained for a longer period than 10 epochs, because the network started to over-fit. Looking at the accuracies, the classifiers are not trained perfectly and this therefore has some influence on the FID and GQI score. In Appendix D also a small experiment can be found in which the metrics are applied using the same generated images data set, only with a flipped ratio of benign and malignant nodules. It shows that this ratio also has an influence on the metrics scores.

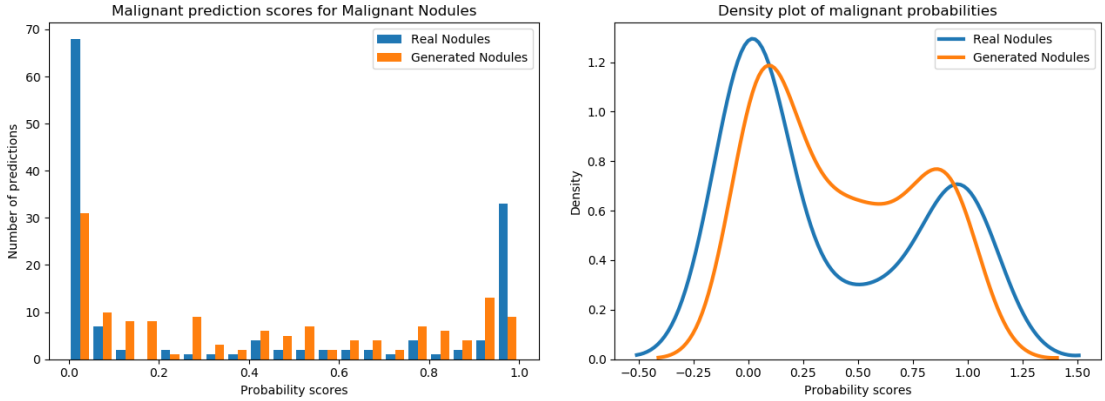
To expand the evaluation of the generated images, also a look at the probability scores has been taken. These scores are assessed using the real classifier tested on both the real test set and the generated test set. When evaluating only the malignant images in the test sets, the real classifier scored 37.6% on the real test set and 41.1% on the generated test set. When evaluating only the benign images in the test sets, the real classifier scored 81.1% on the real test set and 80.8% on the generated test set.

The benign probability scores of the benign images for both test sets can be seen in Figure 5.6. In Figure 5.6a they are plotted in a histogram. A score of 1.0 means the image has been assigned 100% probability that the image contains a benign nodule. Regardless if the probability given is correct or not, it can be seen that the generated nodules are more often assigned a high, confident probability. This can also be seen a little clearer in Figure 5.6b, which shows the density plots of the probabilities. This could be because the GAN created benign nodules which represent a certain feature of benign nodules clearly, or it could be a coincidence.



(a) Histogram **(b)** Density plot

Figure 5.6: Graph showing the benign probability scores assigned to benign test images from both the real nodules test set and the generated nodules test set. Figure 5.6a shows a histogram in which the benign probability scores are plotted. The x-axis represents the probability scores and the y-axis represents the number of predictions. Figure 5.6b shows the density plot of these probability scores. The x-axis shows again the probability scores and the y-axis shows the density.



(a) Histogram **(b)** Density plot

Figure 5.7: Graph showing the malignant probability scores assigned to malignant test images from both the real nodules test set and the generated nodules test set. Figure 5.7a shows a histogram in which the malignant probability scores are plotted. The x-axis represents the probability scores and the y-axis represents the number of predictions. Figure 5.7b shows the density plot of these probability scores. The x-axis shows again the probability scores and the y-axis shows the density.

The malignant probability scores of the malignant images for both test sets can be seen in Figure 5.7. In Figure 5.7a they are plotted in a histogram. A score of 1.0 means the image has been assigned 100% probability that the image contains a malignant nodule. Regardless if the probability given is correct or not, it can be seen that the real nodules are more often assigned a high, confident probability. This can be seen more clearly in Figure 5.7b, which represents the density of the probabilities. When compared to Figure 5.6b, it can be seen that for both test sets the classifier is less sure of the classification which results in a probability more towards 0.5. In the case of the generated nodules, more probabilities are closer to this point. This could be, because the GAN was trained using only the 400 malignant nodules. This number of images is much smaller than the number of benign nodules. This can make it difficult to capture the features of a malignant nodule.

5.4 Discussion

For the purpose of this thesis, all the annotated nodules in the LIDC-IDRI data base were used. A final decision for the nodule, benign or malignant, was included. However, this final decision had not been verified in a later stage in most of the cases. Therefore it is possible that not all benign annotated nodules were benign and the same goes for the malignant nodules. The cases were gathered among eight different imaging modalities and seven different institutes. There may be variations in the acquiring of these cases, i.e. different slice thickness. Ideally all cases would have been made under the same circumstances with the same imaging modality to ensure no variations arise in this area. The images were normalised to a range of $[-1, 1]$, which for a CT scan is not good. Through this normalisation, a lot of information has been lost. If this information could have been preserved, the GAN and the independent classifiers might have performed better.

It was chosen to use images of 64×64 to give the nodule more context of the surroundings. However, it might also be that this was just enough context to make it confusing of what exactly was represented in the images, especially with this classification which does not depend on the position of the nodule per se. Next to this, because of this frame size, it might also have occurred that multiple nodules were present in one image. Especially if one of those nodules is benign and the other malignant, it complicates capturing the right features for the GAN.

Next to this, it was difficult to train a classifier on the data set. If this independent classifier had trouble with extracting correct features, it might also have been more difficult for the GAN to do this. For the independent classifier, an extended data set with flipped images was used. This could also have been done for the GAN. As it was seen that the accuracy increased for the independent classifier after this data augmentation, it would be fair to assume that the GAN would also have performed better. It is also interesting to see that the independent classifier classifies the generated benign images with more confidence than the real benign images. In the case of the malignant nodules this is the other way around. It is possible that the GAN did not have time or enough reference images to clearly produce malignant features, which results in more uncertain classification. It could be that using the flipped data or training for more epochs, might have improved this.

Regarding the outcome of the metrics, the FID is close to 0, which suggests that the activations of the generated nodule data resemble the distribution of the real nodule data. The GQI score has exceeded 100 which indicates that the classifier trained on generated nodules performed better on the real nodule test set than the classifier trained on real nodule data did. The difference in accuracy is not very high (73% and 70%) so it might be that this difference is caused by good guessing. On the other hand, it might also be that the classifier trained on generated nodules has learned a bit more important features of benign and malignant nodules that are represented in the real nodule test set. The NDB also shows a value close to 0. Only 4 different bins are considered to be statistically different. However, the NDB metric seems to vary a lot.

Finally, even with use of the metrics, it is difficult to assess the generated nodule images. It is questionable how reliable the metrics are in this case, especially the FID and GQI since the classifier they use performs much worse than in the case of the fingers data set. Also, the NDB metric did not appear to be robust and is affected by the ratio of image classes in the image set that is used.

6 Discussion

In the previous chapters a short discussion after each chapter has already taken place. In this general discussion the general method will be discussed and relation to other work will be considered. In hindsight it would have been better to dedicate more research into the quantitative assessment of the generated images. After carrying out the questionnaire, applying the metrics and correlating them. It became clear that this is not enough information give a clear conclusion. It did become clear that the three metrics used were not entirely satisfactory. It was therefore still difficult to assess the generated nodule images.

As explained in the Introduction of this thesis, previous work has been conducted in this area as well. Chuquicusma et al. (2018) also used a GAN architecture to create lung nodules. They also used the LIDC-IDRI data base, but had 1145 nodules, because they excluded some. They used a vanilla DC-GAN with a learning rate of 0.0001 and 0.0002 for discriminator and generator. The GAN was trained on benign samples for 114,000 iterations and trained on malignant samples for 110,000 iterations. Next to this, it was also trained on a mixture of samples for 99,000 iterations. The image sizes that were used were much smaller than in this work, but they do not indicate what the precise size was. The authors managed to generate lung nodules, benign as well as malignant and their quality was evaluated by two radiologists. The visual Turing test resulted in one radiologist thinking generated nodules were real 67% of the time and for the other radiologist this was 100% of the time.

In another work by Mirsky et al. (2019), they used a GAN architecture to inject and remove lung nodules in 3D CT scans. They used two cGANs of which one was trained on unhealthy samples and the other on healthy samples. Voxels of $32 \times 32 \times 32$ were used, of which a cube of $16 \times 16 \times 16$ was cut out and filled in by the GAN. Therefore, this experiment can be considered as image-to-image translation instead of generation from scratch, as was the case in this thesis. The authors also used the LIDC-IDRI database, but also applied data augmentation to get to 11,323 nodules for injection and 58,089 for removal. Both cGANs were trained for 200 epochs. Their results were also evaluated by three radiologists. They were asked to evaluate whole CT-scans of which 70 were tampered with and 30 were authentic. Altogether, 99% of injected cases were diagnosed as malignant and 94% of removed cases were diagnosed as being healthy. After informing radiologists of this tampering, the percentages decreased to 60% and 87% respectively.

It is difficult to compare their work and the work in this thesis since there was no quantitative evaluation of the generated nodules. Also, due to lack of time, there was no evaluation by radiologists in this thesis. Also, in the case of Mirsky et al. (2019), it is very difficult to compare the work to the work in this thesis. A big difference already is that image-to-image translation is different from training from scratch. Next to this, in the work of Chuquicusma et al. (2018) a different GAN type was used which therefore also has a different objective function. It can be questioned to what extent different GAN types should be compared to another since their goal can be different.

7 Conclusions and Recommendations

To conclude this thesis, it will be assessed to what extent the goal described in the Introduction is met. The overall goal was to be able to create new patient cases. If this were to be realised, the training of physicians would not be restricted by the availability of patient cases. In order to ensure that an infinite database of varying cases can be realised, it has to be possible to influence the type of cases that are generated. Therefore, it was investigated to what extent a generative model, specifically a GAN, could be created to simulate chest CT patient cases with benign or malignant nodules. Ideally, these patient cases should be indistinguishable from real cases by both radiologists and existing CADs for lung nodule detection. This led to three sub-questions which will be answered separately below.

First of all it needed to be investigated: *In what way can images be created and how can the type of image and its difficulty be influenced?*. It can be concluded that cGANs are a good way to create new images and that their difficulty can be influenced. However, this last part is only possible if the data set, that is used to train with, is already classified according to some measure of difficulty. In this case, it was not possible to classify the nodules in the LIDC-IDRI data set according to their position and category. Also, it should be noted that the training of the GAN is not straightforward and can require some amount of time.

Following this, the next question needed to be researched: *How can the quality of the generated images be quantitatively assessed and does this correspond to human judgement of the quality?*. This question has proven to be difficult to answer. Multiple metrics exist to quantitatively assess the generated images. Of these, three metrics have been applied and correlated with human judgement. In this case it could be concluded that the GQI correlated best with human judgement. However, one of the biggest problems is that it is reliant on the classifier that is used. This does, therefore, not imply that it is the best metric to use on a different data set. Especially if the classifier does not perform as good as in the tested case.

Finally, the question was asked: *To what degree can the nodules be recognised and classified using an algorithm and/or an experts opinion?*. In this case the generated nodules were classified with a similar accuracy as the generated nodules. Benign nodule classifications even had a higher probability of being benign, but malignant nodule classifications had a lower probability being malignant. Therefore it can be concluded that, with this independent classifier used, the generated nodules are classified as good as real nodules. Unfortunately, the generated nodules were not assessed by an expert, which does not allow for a conclusion on this part.

To conclude the overall research, a generative modelling technique, GAN, has been applied to investigate to what extent it is possible to create new chest CT patient cases. These patient cases should either contain benign or malignant nodules. The type of nodule to be generated was successfully influenced. However, this influence is only possible if the network is trained using labels that indicate the type of nodule. The generated cases were generated in 2D and with a size of 64×64. This does not represent a full CT-scan which is 3D and has multiple slices of 512×512.

7.1 Recommendations

Looking at the discussion and the conclusion, a few recommendations can be made for future studies into this area. It is recommended to develop a systematic way to train a GAN. Even though not all adjustments have the same effect in every scenario, it would still be a valuable support to have a systematic way to train the GAN. In this way, hopefully, training possibilities can be explored more efficiently.

Maybe most importantly, it is desirable to have a good quantitative metric to evaluate the generated images. It is very time intensive to ask experts about the quality of the generated images and this opinion may also be influenced by several factors like motivation. All available metrics could be critically assessed under different scenarios to see which is most robust and applicable to most scenarios. On the other hand, it might also be possible to create a new metric that satisfies the conditions better. Ideally this metric would also be applicable to different scenarios; different types of GANs, using different hyperparameters, different optimisation techniques and different types of data sets. Therefore it should probably be a metric that is directly applied on the images and not through for instance a classifier. Next to this, it should not be negatively affected by the variance of images and the ratio in which possible several classes are present. The metric should at least assess on fidelity, diversity of samples and should be able to assess the degree of sampling that can be controlled. Next to this, it should have well-defined bounds. It may be a metric that is specifically designed for lung nodules, since in other situations the goal might be different, but ideally it would be a universal metric.

When continuing on the overall goal it would be interesting to see if the network would perform better when creating full slices of 512×512. This might give the network more context to work with, but it will also be more computationally intensive to realise. The nodule generation might also be enhanced if the used data set has annotations on location categorisations, e.g. juxtapleural. However, in order to realise this, such a data set with accompanying annotations is necessary.

It might also be interesting to follow an approach like the one of Mirsky et al. (2019). The image-to-image translation appears to be a more effective and less time intensive approach, since it takes less epochs to obtain a good result. The idea of inserting a nodule on a desired place can be extended by training on different categories of nodules, e.g. juxtavascular, well-circumscribed. A basic CT-scan can be used, in which multiple positions might be defined. These positions might especially include the so-called "hiding places", which are positions that are easy to miss for radiologists if a nodule is present. The degree of difficulty can then be defined and influenced according to its position and according to the category the generated nodule belongs to. This concept of image-to-image translation might also be tried by using a 2D slice without a nodule as input and training the network on creating a nodule in the output image. In this way, the network is not burdened with generating the correct anatomy, but only with putting a type of nodule in a realistic place.

A Appendix 1 - Task '1' Development of a post-academic course

As part of the course 'Machine Learning for Medical Application', an assignment about the visualisation of deep learning networks was created. As this suggests, the assignment had as its goal to give the person working on it, insight into what aspects the network focuses on in order to be able to classify an image. In order to do this, a short literature research was conducted into visualisation techniques and one technique was chosen to be incorporated into the assignment. In this literature research two different categories were distinguished: Perturbation based visualisations Fong and Vedaldi (2017) and backpropagation based visualisations Simonyan et al. (2013). Because it is a small literature research, this list is not exhaustive. The technique should not be too difficult to execute and should be easily interpretable. In the remainder of this Chapter, first techniques in both the field of perturbation visualisation and backpropagation visualisation will be described. After each description a note is given on possible downsides of the technique. After this, it is described in what way one of the techniques is incorporated into an assignment.

Perturbation based visualisation

Perturbation based visualisation techniques originated from the idea of observing the change of prediction probability when perturbing the pixel intensity of the input image. In all techniques described below, it is assumed that a trained classification network C is available. Also, a testset is available with images I, that can be used as an input image into the network.

Occlusion

As the name occlusion suggests, this perturbation technique uses occlusion of parts of an input image I_0. The occluded image is inserted in network C to observe the change of prediction probability and change in activations of the feature map in the last convolutional layer. (Samek et al., 2016; Zeiler and Fergus, 2014)

A big drawback of this method is that one occlusion largely affects the prediction of a network. Next to this, the occlusions are not very representative for the used data, for instance, an image of a tree with a black square in it (the occlusion) is not encountered in real life.

Super-pixel

A group of adjacent pixels in an image I_0 that have similar intensities, is called a super-pixel component. In super-pixel perturbation, the input image is divided into these super-pixel components, instead of systematically and regularly occluding patches.(Ribeiro et al., 2016) The procedure of super-pixel perturbation can be described in five steps:

1. Segment an input image I_0 into super-pixel components.
2. Generate n samples by:
    - Activating a random amount of super-pixel components. In this description "activation" implies maintaining the intensities in the super-pixel component to it's original values.
    - The super-pixel components that are not activated will be replaced by a corresponding average intensity of all pixels in the super-pixel component.
3. For each n, generate prediction scores.

4. Using the n points, fit a simple regression. The activations of super-pixels in a sample and their corresponding prediction score for the sample can be used for this purpose.

5. Finally, the weights of each super-pixel feature can be used to generate a heatmap.

Also in this technique, a downside is that occlusion of these super-pixels largely affects the prediction scores.

Integrated gradients

The integrated gradients method is in a way a synthesis of perturbation based methods and gradient based methods. As for perturbation, instead of occluding the image directly, it is perturbed over a continuous domain. That is to say that the image goes from a baseline image, for instance all zeroes, to the original image. As for being gradient based, the sensitivity of each pixel with respect to the prediction is integrated over the spectrum to give an approximate attribution score for each pixel. (Sundararajan et al., 2017) The method can be described in 3 steps using input image I_0 with pixel intensities x_{ij}:

1. Generate n samples by varying the pixel intensities linearly from 0 to x_{ij}.

2. Generate the sensitivity for each pixel x in each n:

$$\frac{\partial F(x)}{\partial x_{ij}}\Big|_n \tag{A.1}$$

   With F the function that represents the deep classification network.

3. Obtain a heatmap by integrating the sensitivity of each pixel over the n samples.

The main drawbacks of perturbation methods are that they are mostly computationally expensive and that the perturbations are not realistic. A trained network is not familiar with abnormalities like a black or grey square appearing in an image of for instance a tree. The next section will describe visualisation techniques based on backpropagation, which are mostly more computationally efficient.

Backpropagation based visualisation

Backpropagation

Normal backpropagation, also called vanilla backpropagation, is one of the techniques that can be used to visualise what the neural network "sees". For a trained convolutional network C with input image I_0, class score c and class score function $Sc(I)$, a heatmap can be calculated as an absolute of the gradient of $Sc(I)$ at I_0 using Equation A.2. (Simonyan et al., 2013)

$$\frac{\partial Sc}{\partial I}\Big|_{I_0} \tag{A.2}$$

This Equation is the derivative of Sc with respect to image I at point I_0. Using this, gradients of neurons with negative inputs will be suppressed. This is also schematically represented in Figure A.1. One of the disadvantages of this technique occurs when layers like MaxPooling and BatchNormalisation are used. The partial derivatives are calculated with respect to each pixel. This will however result in a gradient heatmap that is inherently discontinuous.

Guided backpropagation

Guided backpropagation is similar to vanilla backpropagation, but it also suppresses the flow of gradients through neurons of which the incoming gradients are negative. (Springenberg et al., 2014) This is also schematically shown in Figure A.1.

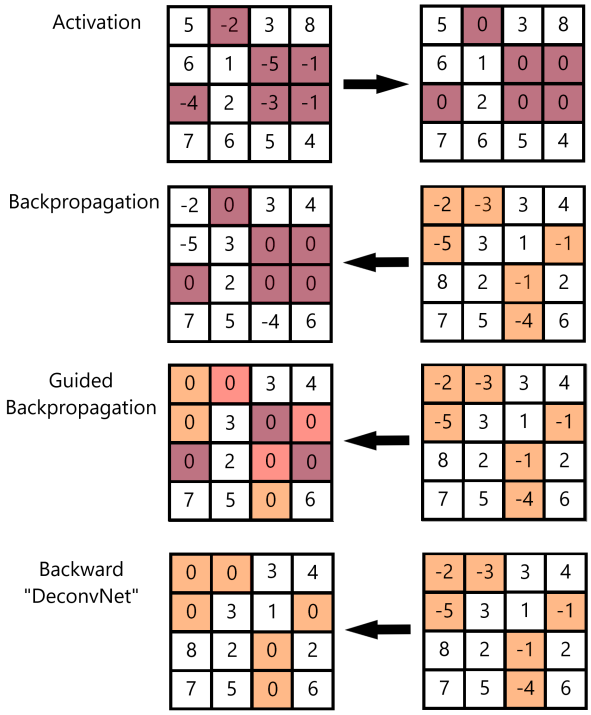


Figure A.1: Activation and different types of backpropagations. From top to bottom respectively, the figure visually shows activation, backpropagation, guided backpropagation and backpropagation using a deconvolution network.

DeconvNet visualisation

The main difference between visualisation using a deconvolutional network (DeconvNet) or visualisation using saliency, is the way backpropagation is realised through non-linear layers, like a ReLU layer. As can also be seen in Figure A.1, in vanilla backpropagation the gradient of neurons that relate to a negative input are suppressed. In DeconvNet backpropagation the gradients of neurons that have an incoming negative gradient are suppressed. (Simonyan et al., 2013)

Grad-CAM

Class Activation Mapping (CAM) uses the activation maps of the last Convolutional layer directly to derive a down sampled relevance map of the input pixels. This down sampled relevance map will be up sampled to the size of the original input image. (Selvaraju et al., 2017)

The procedure can be described in a few steps using again input image I_0, class c, class score function $Sc(I)$ and the feature maps in the final convolution layer $A_1, A_2, ..., A_k$:

1. For each pixel $(i\,j)$ in A_k the weights $w_1, w_2, ..., w_k$ are calculated based on the gradient of class c with respect to each feature map. These weights represent a partial linearisation:

$$w_k^c = \frac{1}{Z}\sum_i\sum_j \frac{\partial Sc}{\partial A_k^{ij}} \tag{A.3}$$

   In which Z is equal to the number of pixels in the feature map A.

2. Weighted activations $A_1, A_2, ..., A_n$ of each pixel in a feature map are obtained using their weights and corresponding feature map activations and a ReLU is applied to obtain the Grad-CAM map $L_{Grad-CAM}$:

$$L^c_{Grad-CAM} = ReLU(\sum_k \alpha^c_k A_k) \tag{A.4}$$

3. The ReLU is applied to ensure only the positive values are left. Important features will result in positive values.

One of the limitations of this technique is that the up sampling of the map will result in artefacts and loss of the signal. Next to this, this technique also has problems with Pooling and Normalization layers.

SmoothGrad

Another method for visualisation is described by Smilkov et al. (2017) and is referred to as SmoothGrad. The method is different from vanilla backpropagation in that it visually sharpens the gradient-based sensitivity maps. This is achieved by adding Gaussian noise to the original image, calculating gradients multiple times and averaging the resulting sensitivity maps. The whole process can be described in multiple steps:

1. Take input image x and sample n similar images by adding Gaussian noise $N(0, \sigma^2)$. The standard deviation of the noise can be described by σ.

$$X_n = x + N(0, \sigma^2) \tag{A.5}$$

   With X_n the n sampled images with Gaussian noise.

2. Create sensitivity maps for every sample and average these maps to get the SmoothGrad map.

$$\hat{M}_c(x) = \frac{1}{n} \sum_1^n M_c \times X_n \tag{A.6}$$

With M_c the sensitivity map of image x and $\hat{M}_c$ the resulting average sensitivity map.

Filter visualisation

Visualisation of filters in a hidden unit can be achieved by maximising the activation in the given hidden layer. To do this, input patterns of bounded norm should be examined to maximise the sum of the input plus the bias in a given hidden unit. This could give a good representation of what the unit is doing. (Courville et al., 2009)

1. Initialise an input image x to a vector of the same size as the images in the training and/or test set.

2. The gradient of the activation of a given hidden layer is computed with respect to x and a step should be made into the gradient direction (gradient ascent).

3. These gradient updates should be continued until the activation function does not increase anymore. This can be approximated by the following formula:

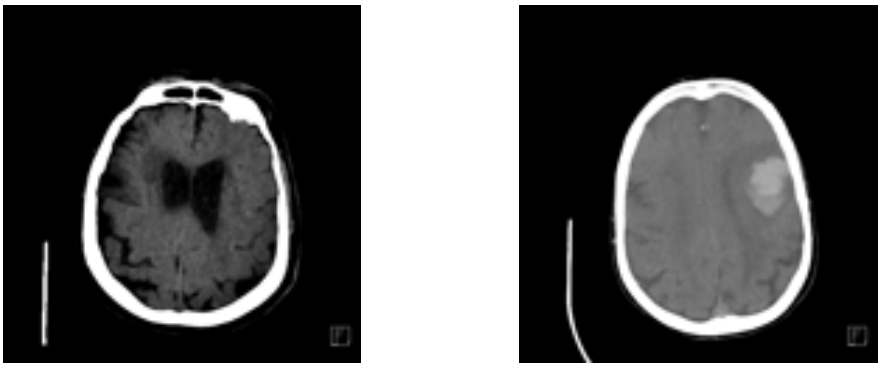
$$x^* = argmax_{||x||=\rho} h_{ij}(\theta, x) \tag{A.7}$$

With θ being the deep network parameters (its weights and biases) and x^* being the image with maximum activation. For a unit i from a layer j in the network, $h_{ij}(\theta, x)$ is the activation.

In this way, the local minimum or minima will be examined using gradient ascent. The gradient of $h_{ij}(\theta, x)$ is computed and x is moved in its direction to create x^*.

Visualisation Assignment

To create the assignment, visualisation based on the occlusion technique was chosen. For the assignment Dual Energy CT-images of slices of the brain with and without the presence of haemorrhage are used. In Figure A.2a an example of such an image without haemorrhage can be seen and in Figure A.2b an example of such an image with haemorrhage.



(a) Without haemorrhage **(b)** With haemorrhage

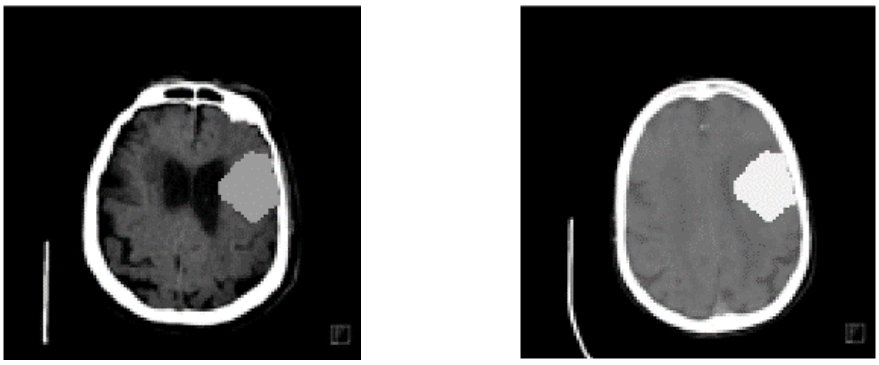
Figure A.2: Dual Energy CT-images of the brain

Using a pre-trained classification network (a CNN), the images in Figure A.2 were classified as haemorrhagic or not. The prediction scores can be found in Table A.1. To see if the visually haemorrhagic area is responsible for this, the haemorrhage in Figure A.2b has been occluded, by applying a mask on the haemorrhage area. This can be seen in Figure A.3b and the corresponding prediction scores in Table A.1. The same has been done for Figure A.2a, in which a haemorrhagic area has been inserted as can be seen in Figure A.3a. Also the prediction scores of this image with inserted haemorrhage can be found in Table A.1.

Table A.1: Prediction scores of several CT brain images. The scores vary between 0 (not certain) and 1 (absolutely certain).

	Figure A.2a	Figure A.3a	Figure A.2b	Figure A.3b
Haemorrhage	0.0509	0.4207	0.9714	0.4986
No Haemorrhage	0.9491	0.5793	0.0286	0.5014

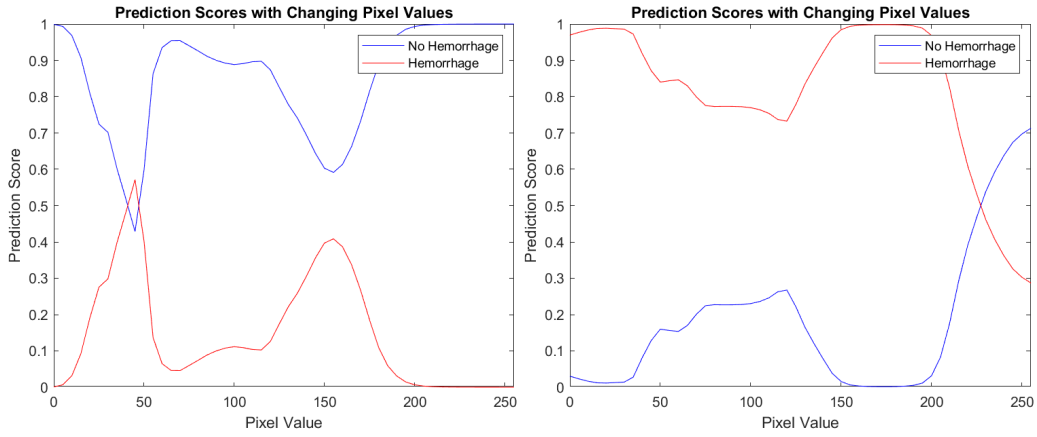
As can be seen from Table A.1, the predictions scores do change a lot when the haemorrhagic area was covered in a haemorrhagic image, or when such an area was inserted when it was not present before. This suggests that this area is of interest for the classification network and that changing the pixel values has an influence on the decision it makes. In fact, the decision changes with changing pixel value. In Figure A.4 it can be seen in what way these prediction scores change, when changing the pixel values from 0 to 255. When looking at these graphs it seems as if the network might also be sensitive to texture and/or contrast and not only to pixel value since the scores deviate much when the pixel value is changed.



(a) Without haemorrhage **(b)** With haemorrhage

Figure A.3: CT-images of the brain with masks added

To investigate what locations have an influence on the outcome of the prediction scores, a square mask is moved over different areas of the image. In this way, features that are unique to an image might be found. Or, in other words, the saliency map is created. In the context of visual processing, saliency refers to features that are unique in an image. A saliency map is in turn the topographical representation of these features. In a deep learning network, for a particular class given an image, they are a way to measure the spatial support. (Simonyan et al., 2013)



(a) Image with inserted haemorrhage **(b)** Image with covered haemorrhage

Figure A.4: The effect of changing the pixel values on a covered or inserted haemorrhage area on the prediction score.

We want to rank pixels in an image, based on their effect on the prediction scores. Given an image i in vectorised form with class c and the prediction scores of the classification network for each class $P_c(i)$, this would look as follows in a linear network:

$$P_c(i) = w_c^T i + b_c \tag{A.8}$$

With w_c the weight vector and b_c the bias of the model. Equation A.8 can also be written in terms of pixels, which result is $P_c(i) = \sum w_c(n,m)i(n,m) + b_c$. If one of the pixels in the image is changed by an amount of i_0 than the prediction score changes into:

$$P_c(i) + \frac{\partial P_c(i)}{\partial i(n,m)} i_0 \tag{A.9}$$

Since the expression is linear, this would result in $P_c(i) + w_c(n,m)i_0$. The sensitivity of the score for a pixel $i(n,m)$ can be described by the rate at which the score changes:

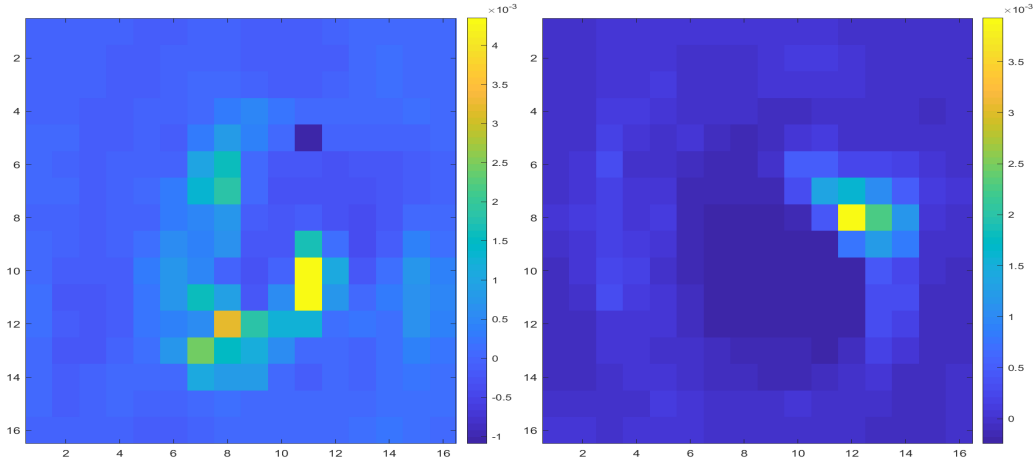
$$S_c(n,m) + \frac{\partial P_c(i)}{\partial i(n,m)} = w_c(n,m) \tag{A.10}$$

Pixel $i(n,m)$ has a large influence on the score if $S_c(n,m)$ is relatively large. However, in a deep convolutional network, the probability score $P_c(i)$ is a highly non-linear function. Still, the rate at which the score changes as the pixel value of one of the pixels changes still holds. Since the score is non-linear, the derivative depends on the choice of i and has therefore to be evaluated for a specific image. The magnitude of this derivative shows which pixels need to be changed in a minimal way to affect the prediction scores in a maximal way. If this is done for every pixel in an image, one can expect that the pixels in objects would correlate with the biggest changes.

For the exercise a numerical approach has been used to see how changes in pixel values affect the probability scores. The numerical expression for the sensitivity of the network will be:

$$S_{n,m} = \frac{P_c(i(R_{n,m}) + i_0) - P_c(i)}{i_0} \tag{A.11}$$

With $R_{n,m}$ a certain region inside image i. Changing the values of (n,m) will result in a finer or coarser sensitivity map of the image. A values of 30 has been added to the region $R_{n,m}$. In Figure A.5 the sensitivity maps of both the brain CT-scan with haemorrhage and without haemorrhage are visible. The sensitivity map is created with 16 regions $R_{n,m}$. Regions that tend more towards yellow have a higher influence on the prediction scores. That is, in this case, increase the prediction score for having no haemorrhage in Figure A.5a and increase the prediction score for having haemorrhage in Figure A.5b.



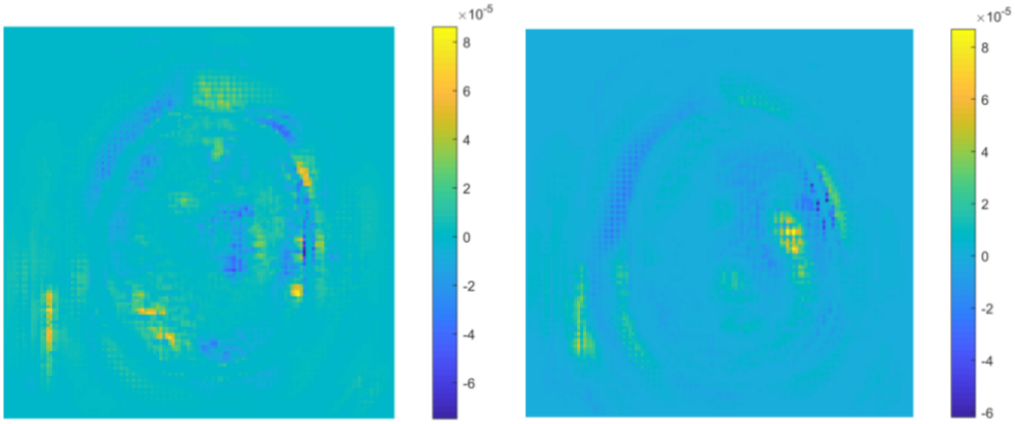
(a) Map without haemorrhage **(b)** Map with haemorrhage

Figure A.5: Numerical saliency maps from the CT-images of 16x16 regions. Regions that tend more towards yellow have a more increased prediction score in the corresponding class. Regions that tend towards the darker blue have a decreased predictions score in the corresponding class. In both cases this is compared to the prediction score with the original pixel value.

In Figure A.6 the sensitivity maps of both the brain CT-scan with haemorrhage and without haemorrhage are also visible. The sensitivity map, however, is created with 128 regions $R_{n,m}$. In both images more finer features are therefore visible. Also in this case regions that tend more towards yellow have a higher influence on the prediction scores. That is increase the prediction

score for having no haemorrhage in Figure A.6a and increase the prediction score for having haemorrhage in Figure A.6b.

In both cases it was to be expected that for the images that included haemorrhage, the area of the haemorrhage would have much influence on the prediction score. This is also what can be observed in Figure A.5b and Figure A.6b. Also, in Figure A.6a some outlines of the brain can be observed, which indicates that these area also influence the prediction score. For the images that do not include haemorrhage it is more difficult to predict which areas would have the most influence. One thing that could be expected is that in areas where haemorrhage is common, the prediction score for no haemorrhage might go down. However, when looking at Figure A.5a such areas are not clearly visible. In Figure A.6a it is also possible to see the outlines of the brain, which also indicates that these areas are important in the prediction score.



(a) Map without haemorrhage **(b)** Map with haemorrhage

Figure A.6: Sensitivity maps from the CT-images of 128x128 regions. Regions that tend more towards yellow have a more increased prediction score in the corresponding class. Regions that tend towards the darker blue have a decreased predictions score in the corresponding class. In both cases this is compared to the prediction score with the original pixel value.

To conclude this, it is important to note what clinical value these sensitivity maps might have. Especially the sensitivity map for the prediction score of haemorrhage might be interesting since it would highlight areas the network thinks are important for this decision. In this way the map might be used to give an indication as to what areas might need further inspection when a diagnosis is made.

B Appendix 2 - Derivation of Optimal D

For any given G the training criterion for D is to maximise Equation 2.11, resulting in $D_G^* = \underset{D}{argmax} V(D,G)$. Using that the expectation $\mathbb{E}_{p(x)}[x] = \int_x x p_x(x) dx$ this will result in:

$$V(G,D) = \int_x p_{data}(x) log(D(x)) dx + \int_z p_Z(z) log(1 - D(G(z))) dz \tag{B.1}$$

Next, it can be shown that, using a concept in probability theory that is called "change of variable", that:

$$\int_z p_Z(z) log(1 - D(G(z))) dz = \int_x p_g(x) log(1 - D(x)) dx \tag{B.2}$$

Suppose a random variable x is given and its probability density function is $p_x(x)$. It is possible to calculate the probability density function of some other variable $K = G(x)$, in which G can be any function. This will result in the following probability density function:

$$p_K(k) = p_x(G^{-1}(k)) | \frac{d}{dk}(G^{-1}(k)) | \tag{B.3}$$

If we apply this to the original case, we have random variable z with probability density function $p_Z(z)$ and we want to calculate the probability density function of some variable $x = G(z)$ this will result in:

$$p_g(x) = p_z(G^{-1}(x)) \frac{d}{dx}(G^{-1}(x)) \tag{B.4}$$

It is written as p_g since it is the distribution coming out of G, but it could also be written as p_x. To prove Equation B.2, we need to assume that G is invertible, i.e. $z = G^{-1}(x)$. In that case we get:

$$\begin{aligned} \int_z p_Z(z) log(1 - D(G(z))) dz &= \int_x p_Z(G^{-1}(x) log(1 - D(x)) dG^{-1}(x) \\ &= \int_x p_Z(G^{-1}(x)) log(1 - D(x)) \frac{dG^{-1}(x)}{dx} dx \end{aligned} \tag{B.5}$$

If we rewrite Equation B.5 and use Equation B.4 this proves Equation B.2. Now, it is possible to write Equation B.1 as:

$$\begin{aligned} V(G,D) &= \int_x p_{data}(x) log(D(x)) dx + \int_x p_g(x) log(1 - D(x)) dx \\ &= \int_x [p_{data}(x) log(D(x)) + p_g(x) log(1 - D(x))] dx \end{aligned} \tag{B.6}$$

The optimal D, given G, is obtained by maximising Equation B.6. This can be done if we take the derivative with respect to D:

$$\begin{aligned} \frac{d}{dD(x)} [p_{data}(x) log(D(x)) + p_g(x) log(1 - D(x))] &= 0 \\ \frac{p_{data}(x)}{D(x)} - \frac{p_g(x)}{1 - D(x)} &= 0 \end{aligned} \tag{B.7}$$

Rewriting Equation B.7 result in the optimal D:

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \tag{B.8}$$

The only step left to do would be to prove if this is indeed the maximum value. This can be done by taking the double derivative of Equation B.7 with respect to dx and this will result in:

$$\frac{-p_{data}(x)}{(D(x))^2} - \frac{p_g(x)}{(1 - D(x))^2} < 0 \tag{B.9}$$

This is not practically calculable, since $p_{data}(x)$ is not known a priori. However, it is sufficient to show that it is a maximum.

C Appendix 3 - Derivation of Optimal G

The optimal G can be obtained by minimising Equation 2.11 with respect to G given the optimal discriminator: $G^* = \underset{G}{argmin} V(D_G^*, G)$. Filling in D_G^* this will result in:

$$\begin{aligned} G^* &= \underset{G}{argmin} \{\int_x \{p_{data}(x) log(D_G^*(x)) + p_g(x) log(1 - D_G^*(x))\} dx\} \\ &= \underset{G}{argmin} \{\int_x \{p_{data}(x) log(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)}) + p_g(x) log(\frac{p_g(x)}{p_{data}(x) + p_g(x)})\} dx\} \end{aligned} \tag{C.1}$$

If $(log2) p_{data}(x)$ is added and subtracted as well as $(log2) p_g(x)$ this results in:

$$\begin{aligned} G^* &= \underset{G}{argmin} \{\int_x \{(log2 - log2) p_{data}(x) + p_{data}(x) log(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)}) \\ &\quad + (log2 - log2) p_g(x) + p_g(x) log(\frac{p_g(x)}{p_{data}(x) + p_g(x)})\} dx\} \\ &= \underset{G}{argmin} \{\int_x \{-log2(p_{data}(x) + p_g(x)) dx + p_{data}(x)[log2 + \\ &\quad log(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)})] + p_g(x)[log2 + log(\frac{p_g(x)}{p_{data}(x) + p_g(x)})]\} dx\} \\ &= \underset{G}{argmin} \{-log2 \int_x (p_{data}(x) + p_g(x)) dx + \int_x p_{data}(x)[log2 + \\ &\quad log(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)})] dx + \int_x p_g(x)[log2 + log(\frac{p_g(x)}{p_{data}(x) + p_g(x)})] dx\} \end{aligned} \tag{C.2}$$

Resolving these three integrals and keeping in mind that the integral of the probability density function will be 1, gives:

$$G^* = \underset{G}{argmin} \{-log2(1+1) + \int_x p_{data}(x) log(\frac{p_{data}(x)}{\frac{p_{data}(x) + p_g(x)}{2}} dx + \int_x p_g(x) log(\frac{p_g(x)}{\frac{p_{data}(x) + p_g(x)}{2}} dx\} \tag{C.3}$$

The two integrals left in Equation C.3 are equal to the KL divergence. Using this information it is possible to write:

$$G^* = \underset{G}{argmin} \{-log4 + KL[p_{data}(x) || \frac{p_{data} + p_g}{2}] + KL[p_g(x) || \frac{p_{data} + p_g}{2}] \tag{C.4}$$

Equation C.4 can be rewritten again using the Jenson Shanon Divergence (JDS). This result in:

$$G^* = \underset{G}{argmin} \{-log4 + 2JSD(p_{data}(x) || p_g(x))\} \tag{C.5}$$

Now if $p_g(x) = p_{data}(x)$ the JSD term will become 0.

D Appendix 4 - Other small Metric Experiments

Extra test with fingers data set

Next to applying the metrics on the six image data sets, two other small tests have been run to see the effect on the metric outcomes. They also have been applied to the real image test set. Furthermore, image set 3 (half data) has been obtained by using the same training parameters used to train the GAN for image set 3, only half the size of the images is used to train the GAN with. These results can be seen in Table D.1.

Table D.1: Metric results using the test set (containing real image data) and using image set 3, trained on half the size of images. Also, the results of Image set 3 (whole image data set) has been added for reference.

Image set	FID	GQI	NDB
Test Set	0.445	100.0	0
Image set 3 (half data)	205.30	83.61	16
Image set 3	163.82	96.06	12

It can be seen that using only half the data to train with has a big influence on the outcome of the metrics. The FID and the NDB have increased and the GQI has decreased. In all cases this means that the resulting generated images score worse on these metrics compared to when the bigger image data set is used to train with. This indicates that the amount of images used to train with has an influence on the performance of a GAN. One explanation could be that, because the training set is smaller, the discriminator and generator have had less updates. It would be interesting to see if the scores would be similar if double the amounts of epochs are used, which results in the same amount of updates.

Extra tests with nodule data set

To evaluate the metrics a more, it has been investigated if the ratio, in which the different classes of the data set are present, has any influence on the metrics. As described in Chapter 6 the used number of benign and malignant nodules was 1,710 and 400 respectively. To see the effect of this ratio, the number of benign and malignant nodules has been flipped. So, in this case 400 benign nodules and 1,710 malignant nodules are present. The results when the metrics are applied can also be seen in Table D.2.

Table D.2: Metric results using different ratios of benign and malignant nodules. In the original scenario 1710 benign and 400 malignant nodules are used, whereas in the flipped scenario 400 benign and 1710 malignant nodules are used.

Scenario	FID	GQI	NDB
Original	0.016	105.18	4
Flipped	0.072	103.54	8

It can be seen that all the metrics are influenced by using a different ratio of classes in the generated image data set. Both the FID and NDB become higher and the GQI becomes lower. All three indicate a performance that is worse compared to the original scenario. For the FID this could be because, having a larger amount of one class and a smaller amount of the other,

changes the activation distribution. In case of the GQI, the classifier trained on the generated image data sets will have learned more features about malignant nodules and less about benign nodules when compared to the original scenario. This has an influence when tested on the real image test set. When looking at the NDB, the presence of a higher amount of images of one class could results in more samples being assigned to one bin. This results in this bin being statistically different and therefore affecting the final score. This indicates that one should be very careful with the ratio of classes in a generated image data set when used with these metrics.

Another important aspect that was noticed with the NDB metric, is that its results vary every time it is applied on the same image data set. Therefore, the NDB metric has been applied on the original and flipped scenario 10 times to see its variation. The results can be seen in Figure D.1.

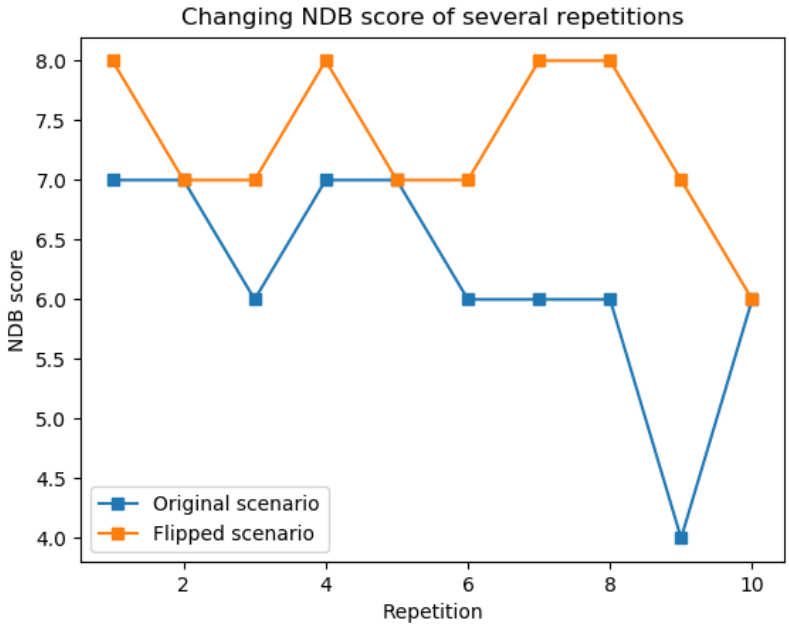


Figure D.1: Changing NDB metric on the original scenario (blue) and the flipped scenario (orange) when tested 10 times. The x-axis shows the test number and the y-axis shows the NDB metric score.

From this Figure it can clearly be seen that the results vary with each repetition. This also results in that at some repetition the score of the original scenario does not vary from the score of the flipped scenario, but in another repetition it varies a lot (see for instance repetition 9). This may be because, every time the metric is applied again, the real image data set is again divided into Voronoi cells according to the K-means clustering. This division may vary a little every time it is created. Apparently this has a big influence on the score and does therefore not make the metric very robust.

E Appendix 5 - Architectures & Images

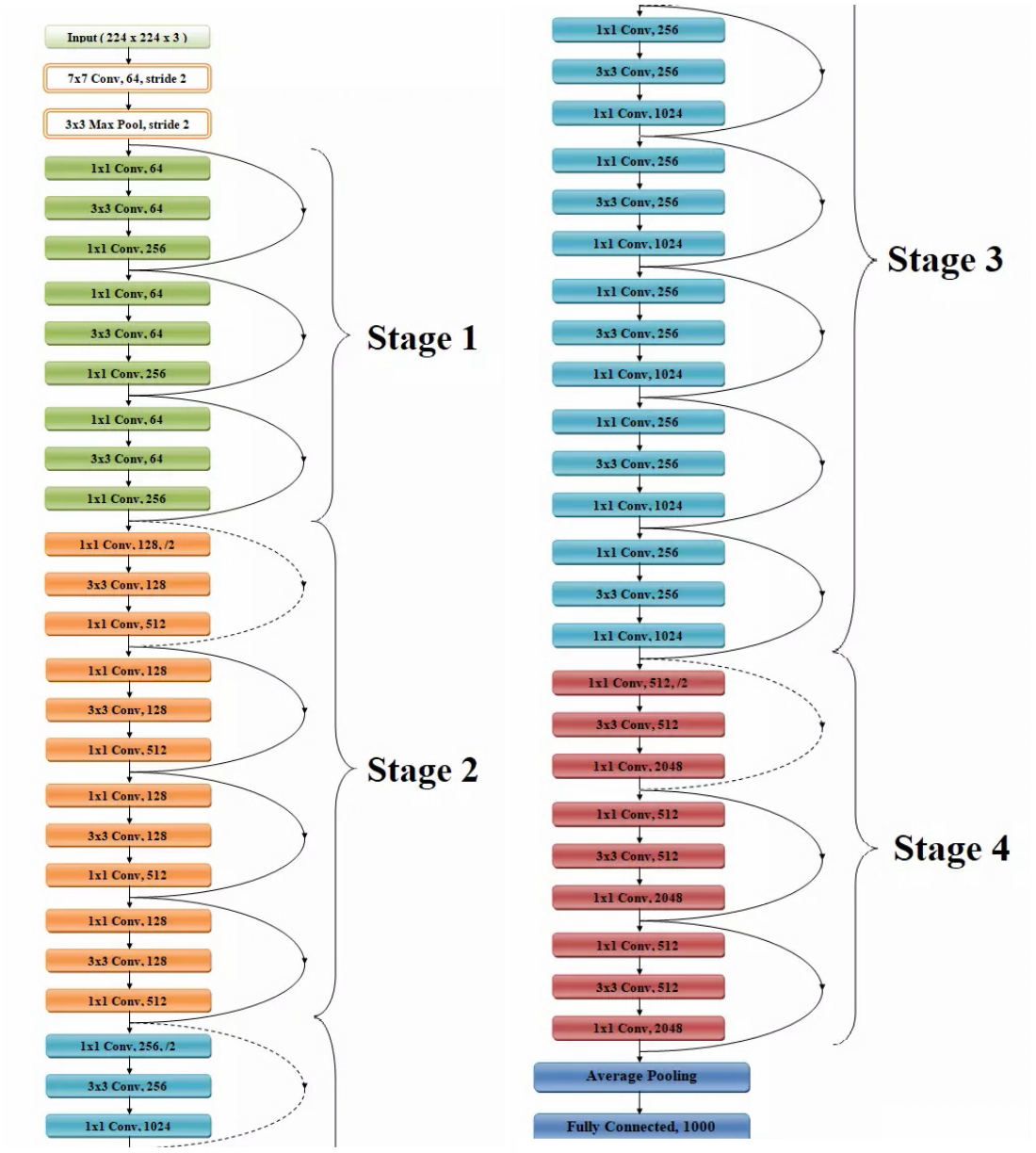
CNN architecture fingers data

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 128, 128, 64)	1088
activation_1 (Activation)	(None, 128, 128, 64)	0
batch_normalization_1 (Batch	(None, 128, 128, 64)	256
conv2d_2 (Conv2D)	(None, 128, 128, 64)	65600
activation_2 (Activation)	(None, 128, 128, 64)	0
batch_normalization_2 (Batch	(None, 128, 128, 64)	256
max_pooling2d_1 (MaxPooling2	(None, 64, 64, 64)	0
dropout_1 (Dropout)	(None, 64, 64, 64)	0
conv2d_3 (Conv2D)	(None, 64, 64, 128)	131200
activation_3 (Activation)	(None, 64, 64, 128)	0
batch_normalization_3 (Batch	(None, 64, 64, 128)	512
conv2d_4 (Conv2D)	(None, 64, 64, 128)	262272
activation_4 (Activation)	(None, 64, 64, 128)	0
batch_normalization_4 (Batch	(None, 64, 64, 128)	512
max_pooling2d_2 (MaxPooling2	(None, 32, 32, 128)	0
dropout_2 (Dropout)	(None, 32, 32, 128)	0
conv2d_5 (Conv2D)	(None, 32, 32, 128)	262272
activation_5 (Activation)	(None, 32, 32, 128)	0
batch_normalization_5 (Batch	(None, 32, 32, 128)	512
conv2d_6 (Conv2D)	(None, 32, 32, 128)	262272
activation_6 (Activation)	(None, 32, 32, 128)	0
batch_normalization_6 (Batch	(None, 32, 32, 128)	512
max_pooling2d_3 (MaxPooling2	(None, 16, 16, 128)	0
dropout_3 (Dropout)	(None, 16, 16, 128)	0
flatten_1 (Flatten)	(None, 32768)	0
dense_1 (Dense)	(None, 128)	4194432
activation_7 (Activation)	(None, 128)	0
dense_2 (Dense)	(None, 12)	1548

Total params: 5,183,244
Trainable params: 5,181,964
Non-trainable params: 1,280

Figure E.1: CNN architecture used when trained with the fingers image data set. In the last layer, $dense_2$, the outputlayer is 6 instead of 12.

ResNet50 architecture



(a) ResNet50 architecture - part 1

(b) ResNet50 architecture - part 2

Figure E.2: ResNet50 architecture schematically represented. Picture taken from https://cv-tricks.com/keras/understand-implement-resnets/

Variance images generated nodules

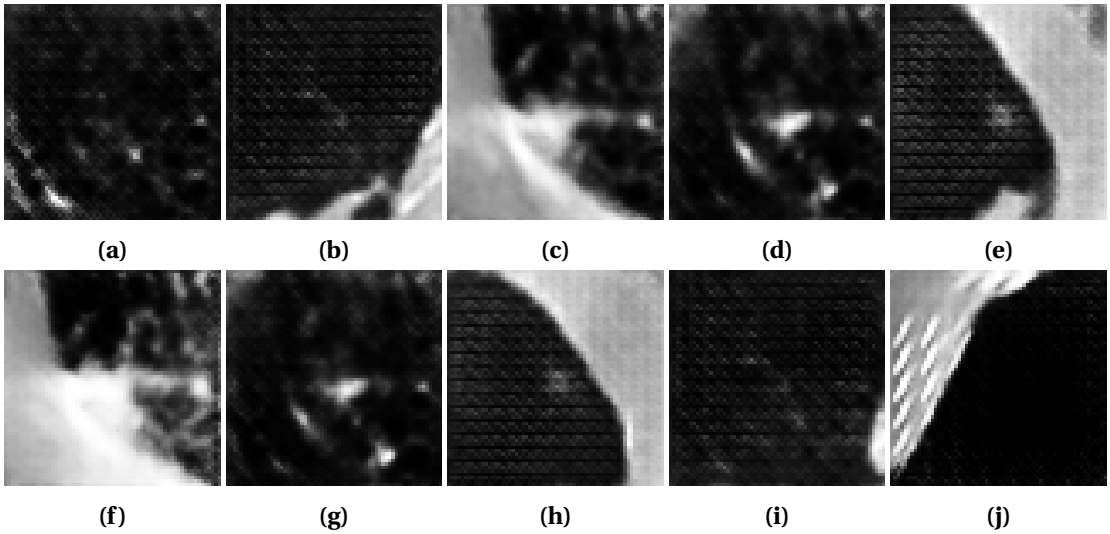


Figure E.3: Ten generated images of benign nodules produced by the same generator. It can be seen that (c) and (f) look much alike, as well as (e) and (h) and (d) and (g).

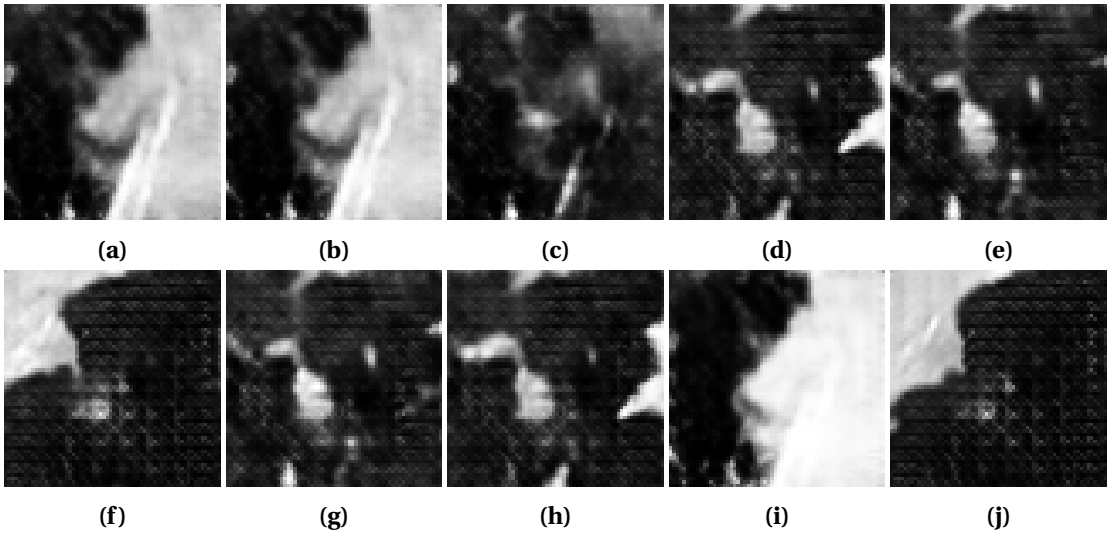
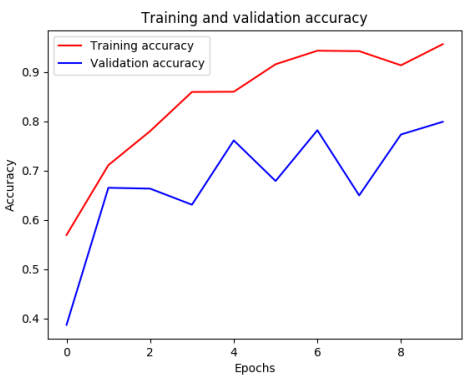
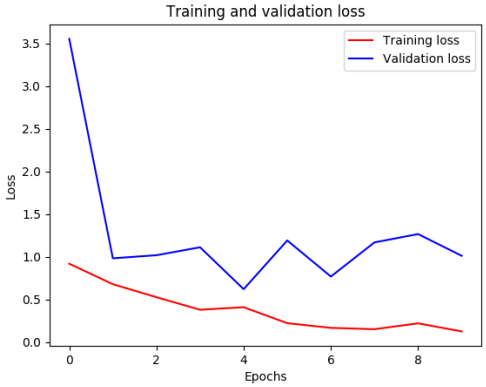


Figure E.4: Ten generated images of malignant nodules produced by the same generator. It can be seen that (a), (b) and (i) look much alike, as well as (c), (d), (e) and (h).

Accuracy and Loss graphs of trained ResNet50

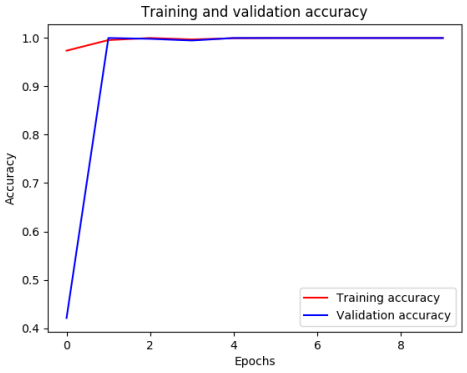


(a) Accuracy of classifier over 10 epochs

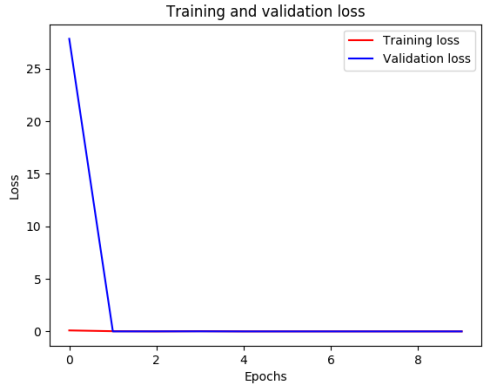


(b) Loss of classifier over 10 epochs

Figure E.5: Graph of the accuracy and loss during training of the classifier on the real nodule training set. The left graph shows the accuracy with the epoch number on the x-axis and the accuracy on the y-axis. The right graph shows the loss with the epoch number on the x-axis and the loss on the y-axis.



(a) Accuracy of classifier over 10 epochs



(b) Loss of classifier over 10 epochs

Figure E.6: Graph of the accuracy and loss during training of the classifier on the real nodule training set. The left graph shows the accuracy with the epoch number on the x-axis and the accuracy on the y-axis. The right graph shows the loss with the epoch number on the x-axis and the loss on the y-axis.

Bibliography

Allanson, J. (1956), Some properties of randomly connected neural nets, *Information Theory*.

Arjovsky, M., S. Chintala and L. Bottou (2017), Wasserstein Generative Adversarial Networks, in *Proceedings of the 34th International Conference on Machine Learning*, PMLR, Sydney, Australia, pp. 214–223.

Armato III, S. G., G. McLennan, L. Bidaut, M. F. McNitt-Gray, C. R. Meyer, A. P. Reeves and L. P. Clarke (2015), LIDC-IDRI - The Cancer Imaging Archive (TCIA) Public Access - Cancer Imaging Archive Wiki, doi:10.7937.

Berthelot, D., T. Schumm and L. Metz (2017), BEGAN: Boundary Equilibrium Generative Adversarial Networks, *CoRR*, arXiv ID 1703.10717.

Bí, M., D. J. Sutherland, M. Arbel and A. Gretton (2018), Demystifying MMD GANs, in *ICLR*, arXiv ID 1801.01401v4.

Borji, A. (2018), Pros and Cons of GAN Evaluation Measures, *Computer Vision and Image Understanding*, volume 179, arXiv ID 1802.03446.

Brock, A., J. Donahue and K. Simonyan (2018), Large Scale GAN Training for High Fidelity Natural Image Synthesis, *CoRR*, arXiv ID 1809.11096v2.

Chuquicusma, M. J. M., S. Hussein, J. Burt and U. Bagci (2018), How to fool radiologists with generative adversarial networks? A visual turing test for lung cancer diagnosis, in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, IEEE, pp. 240–244, ISBN 978-1-5386-3636-7, doi:10.1109/ISBI.2018.8363564.

Clark, K., B. Vendt, K. Smith, J. Freymann, J. Kirby, P. Koppel, S. Moore, S. Phillips, D. Maffitt, M. Pringle, L. Tarbox and F. Prior (2013), The Cancer Imaging Archive (TCIA): Maintaining and Operating a Public Information Repository, *Journal of Digital Imaging*, volume 26, doi:10.1007/s10278-013-9622-7.

Courville, A., D. Erhan, Y. Bengio and P. Vincent (2009), Visualizing Higher-Layer Features of a Deep Network Visualizing Higher-Layer Features of a Deep Network, *University of Montreal*.

Daniel, W. W. and C. L. Cross (2018), *Biostatistics : a foundation for analysis in the health sciences*, Wiley, ISBN 9781119282372.

Dayan, P., B. J. Frey and G. E. Hinton (1996), Does the Wake-sleep Algorithm Produce Good Density Estimators?, in *Advances in neural information processing systems 8 (NIPS '95)*, Eds. M. M. D. Touretzky and M. Hasselmo, MIT Press, Cambridge, MA, pp. 661–667.

Denton, E., S. Chintala, A. Szlam and R. Fergus (2015), Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks, *Advances in neural information processing systems*.

Detterbeck, F. C., P. E. Postmus and L. T. Tanoue (2013), The Stage Classification of Lung Cancer, *Chest*, volume 143, doi:10.1378/chest.1435S1.

Fong, R. C. and A. Vedaldi (2017), Interpretable Explanations of Black Boxes by Meaningful Perturbation, in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3429–3437, arXiv ID 1704.03296v3.

Goodfellow, I. (2016), NIPS 2016 Tutorial: Generative Adversarial Networks, in *NIPS 2016 tutorial: Generative adversarial networks.*, arXiv ID 1701.00160v4.

Goodfellow, I., Y. Bengio and A. Courville (2016), *Deep Learning*, MIT Press.

Goodfellow, I. J., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio (2014), Generative Adversarial Nets, *Advances in neural information processing systems*, arXiv ID 1406.2661v1.

He, K., X. Zhang, S. Ren and J. Sun (2016), Deep Residual Learning for Image Recognition, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, arXiv ID 1512.03385.

Heusel, M., H. Ramsauer, T. Unterthiner, B. Nessler and S. Hochreiter (2017), GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium, *Advances in Neural Information Processing Systems*.

Heuvelmans, M. A., M. Vonder, M. Rook, H. J. M. Groen, G. H. De Bock, X. Xie, M. J. Ijzerman, R. Vliegenthart and M. Oudkerk (2019), Screening for Early Lung Cancer, Chronic Obstructive Pulmonary Disease, and Cardiovascular Disease (the Big-3) Using Low-dose Chest Computed Tomography: Current Evidence and Technical Considerations., *Journal of thoracic imaging*, volume 34, doi:10.1097/RTI.0000000000000379.

de Hoop, B., D. W. De Boo, H. A. Gietema, F. van Hoorn, B. Mearadji, L. Schijf, B. van Ginneken, M. Prokop and C. Schaefer-Prokop (2010), Computer-aided Detection of Lung Cancer on Chest Radiographs: Effect on Observer Performance, *Radiology*, volume 257, doi:10.1148/radiol.10092437.

International Agency for Research on Cancer (2018), Lung Fact-Sheet, doi:http://gco.iarc.fr/today/data/factsheets/cancers/15-Lung-fact-sheet.pdf.

Jin, C.-B., H. Kim, M. Liu, W. Jung, S. Joo, E. Park, Y. S. Ahn, I. H. Han, J. I. Lee, X. Cui, C.-B. Jin, H. Kim, M. Liu, W. Jung, S. Joo, E. Park, Y. S. Ahn, I. H. Han, J. I. Lee and X. Cui (2019), Deep CT to MR Synthesis Using Paired and Unpaired Data, *Sensors*, volume 19, ISSN 1424-8220, doi:10.3390/s19102361.

Jordan, M., Z. Ghahramani and T. Jaakkola (1999), An Introduction to Variational Methods for Graphical Models, *Machine Learning*, volume 37, doi:10.1023/A:1007665907178.

Jorritsma, W., F. Cnossen and P. van Ooijen (2015), Improving the radiologist–CAD interaction: designing for appropriate trust, *Clinical Radiology*, volume 70, doi:10.1016/j.crad.2014.09.017.

Karras, T., S. Laine and T. Aila (2019), A Style-Based Generator Architecture for Generative Adversarial Networks Timo Aila NVIDIA, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410, arXiv ID 1812.04948.

Kendall, M. G. (1938), A new measure of rank correlation, *Biometrika*, volume 30.

Kingma, D. P. (2013), Fast Gradient-Based Inference with Continuous Latent Variable Models in Auxiliary Form, Technical report, arXiv ID 1306.0733.

Kodali, N., J. Abernethy, J. Hays and Z. Kira (2017), On Convergence and Stability of GANs, *CoRR*, arXiv ID 1705.07215.

Kullback, S. and R. A. Leibler (1951), On information and sufficiency, *The annals of mathematical statistics*, volume 22.

Lucic, M., K. Kurach, M. M. Google, B. O. Bousquet and S. Gelly (2018), Are GANs Created Equal? A Large-Scale Study, in *Advances in neural information processing systems 31*, Curran Associates, Inc., pp. 700–709.

MacMahon, H., D. P. Naidich, J. M. Goo, K. S. Lee, A. N. Leung, J. R. Mayo, A. C. Mehta, Y. Ohno, C. A. Powell, M. Prokop, G. D. Rubin, C. M. Schaefer-Prokop, W. D. Travis, P. E. Van Schil and A. A. Bankier (2017), Guidelines for management of incidental pulmonary nodules detected on CT images: From the Fleischner Society 2017, *Radiology*, volume 284, doi:10.1148/radiol.2017161659.

Majtey, A., P. W. Lamberti, M. T. Martin and A. Plastino (2005), Wootters' distance revisited: A new distinguishability criterium, *European Physical Journal D*, volume 32, arXiv ID quant-ph/0408082, doi:10.1140/epjd/e2005-00005-1.

McCorduck, P. (2004), *Machines Who Think*, A K Peters, Ltd., ISBN 1-56881-205-1.

McLennan, G., S. G. Armato, L. Bidaut, M. F. McNitt-Gray, C. R. Meyer, A. P. Reeves, B. Zhao, D. R. Aberle, C. I. Henschke, E. A. Hoffman, E. A. Kazerooni, H. MacMahon, E. J. R. van Beek, D. Yankelevitz, A. M. Biancardi, P. H. Bland, M. S. Brown, R. M. Engelmann, G. E. Laderach, D. Max, R. C. Pais, D. P.-Y. Qing, R. Y. Roberts, A. R. Smith, A. Starkey, P. Batra, P. Caligiuri, A. Farooqi, G. W. Gladish, C. M. Jude, R. F. Munden, I. Petkovska, L. E. Quint, L. H. Schwartz, B. Sundaram, L. E. Dodd, C. Fenimore, D. Gur, N. Petrick, J. Freymann, J. Kirby, B. Hughes, A. Vande Casteele, S. Gupte, M. Sallam, M. D. Heath, M. H. Kuhn, E. Dharaiya, R. Burns, D. S. Fryd, M. Salganicoff, V. Anand, U. Shreter, S. Vastagh, B. Y. Croft and L. P. Clarke (2011), The Lung Image Database Consortium and Image Database Resource Initiative: A Completed Reference Database of Lung Nodules on CT Scans, *Medical Physics*, volume 38, doi:10.1118/1.3528204.

Mino, A. and G. Spanakis (2018), LoGAN: Generating Logos with a Generative Adversarial Neural Network Conditioned on color, in *17th IEEE International Conference on Machine Learning and Applications*, IEEE, arXiv ID 1810.10395.

Mirsky, Y., T. Mahler, I. Shelef and Y. Elovici (2019), CT-GAN: Malicious Tampering of 3D Medical Imagery using Deep Learning, *CoRR*, arXiv ID 1901.03597v3.

Mirza, M. and S. Osindero (2014), Conditional Generative Adversarial Nets, *CoRR*, arXiv ID 1411.1784v1.

Nguyen, A., J. Clune, Y. Bengio, A. Dosovitskiy and J. Yosinski (2017), Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4467–4477.

Nguyen, X. V., S. J. Adams, S. K. Hobbs, D. Ganeshan and A. P. Wasnik (2019), Radiologist as Lifelong Learner: Strategies for Ongoing Education, *Academic Radiology*, volume 26, doi:10.1016/j.acra.2019.03.019.

Nie, D., R. Trullo, J. Lian, C. Petitjean, S. Ruan, Q. Wang and D. Shen (2017), Medical Image Synthesis with Context-Aware Generative Adversarial Networks, in *Medical Image Computing and Computer Assisted Intervention 2017*, volume 10435, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 417–425, ISBN 978-3-642-40762-8, doi:10.1007/978-3-642-40763-5.

van den Oord, A., N. Kalchbrenner and K. Kavukcuoglu (2016), Pixel Recurrent Neural Networks, *CoRR*, arXiv ID 1601.06759v3.

Peter B. O'donovan, M. (1997), The Radiologic Appearance of Lung Cancer, *Lung Cancer, Oncology*, volume 11.

Pinto, A. and L. Brunese (2010), Spectrum of diagnostic errors in radiology, *World Journal of Radiology*, volume 2, doi:10.4329/wjr.v2.i10.377.

Purandare, N. C. and V. Rangarajan (2015), Imaging of lung cancer: Implications on staging and management., *The Indian journal of radiology & imaging*, volume 25, doi:10.4103/0971-3026.155831.

Radford, A., L. Metz and S. Chintala (2015), Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, in *ICLR 2016*, arXiv ID 1511.06434v2.

Ratliff, L. J., S. A. Burden and S. S. Sastry (2013), Characterization and computation of local Nash equilibria in continuous games, in *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, Monticello, IL, USA, pp. 917–924, ISBN 978-1-4799-3410-2, doi:10.1109/Allerton.2013.6736623.

Reiner, B. (2008), Automating Radiologist Workflow, Part 3: Education and Training, *Journal of the American College of Radiology*, volume 5, doi:10.1016/j.jacr.2008.05.013.

Rezende, D. J., S. Mohamed and D. Wierstra (2014), Stochastic Backpropagation and Approximate Inference in Deep Generative Models, in *International Conference on Machine Learning 31*, JMLR: W&CP, arXiv ID 1401.4082.

Ribeiro, M. T., S. Singh and C. Guestrin (2016), "Why Should I Trust You?" Explaining the Predictions of Any Classifier, in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining.*, ACM, pp. 1135–1144, ISBN 9781450342322, doi: 10.1145/2939672.2939778, arXiv ID 1602.04938v3.

Richardson, E. and Y. Weiss (2018), On GANs and GMMs, in *Advances in Neural Information Processing Systems*, pp. 5847–5858, arXiv ID 1805.12462.

Rosenblatt, F. (1961), Principles of neurodynamics. perceptrons and the theory of brain mechanisms, Technical report, Cornell Aeronautical Lab Inc Buffalo NY.

Salimans, T., I. Goodfellow, W. Zaremba, V. Cheung, A. Radford and X. Chen (2016), Improved Techniques for Training GANs, *Advances in neural information processing systems*.

Samek, W., A. Binder, G. Montavon, S. Bach and K.-R. Müller (2016), Evaluating the visualization of what a Deep Neural Network has learned, *IEEE transactions on neural networks and learning systems*, volume 28, arXiv ID 1509.06321v1.

Selvaraju, R. R., M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra (2017), Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization, in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626, arXiv ID 1610.02391v3.

Shaukat, F., G. Raja and A. F. Frangi (2019), Computer-aided detection of lung nodules: a review, *Journal of Medical Imaging*, volume 6, doi:10.1117/1.jmi.6.2.020901.

Simonyan, K., A. Vedaldi and A. Zisserman (2013), Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, *CoRR*, arXiv ID 1312.6034v1.

Smilkov, D., N. Thorat, B. Kim, F. Viégas and M. Wattenberg (2017), SmoothGrad: removing noise by adding noise, *CoRR*, arXiv ID 1706.03825v1.

Springenberg, J. T., A. Dosovitskiy, T. Brox and M. Riedmiller (2014), Striving for Simplicity: The All Convolutional Net, in *ICLR 2015*, arXiv ID 1412.6806v3.

Sundararajan, M., A. Taly and Q. Yan (2017), Axiomatic Attribution for Deep Networks, in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR, pp. 3319–3328, arXiv ID 1703.01365v2.

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich (2014), Going Deeper with Convolutions, *CoRR*, arXiv ID 1409.4842.

Szegedy, C., V. Vanhoucke, S. Ioffe and J. Shlens (2016), Rethinking the Inception Architecture for Computer Vision, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826.

The National Lung Screening Trial Research Team (2011), Reduced Lung-Cancer Mortality with Low-Dose Computed Tomographic Screening, *New England Journal of Medicine*, volume 365, doi:10.1056/NEJMoa1102873.

Theis, L., A. v. d. Oord and M. Bethge (2015), A note on the evaluation of generative models, in *ICLR 2016*, arXiv ID 1511.01844.

Turing, A. M. (1950), Computing Machinery and Intelligence, *the Journal of the Mind Association*, volume LIX.

Uhr, L. and C. Vossler (1961), A pattern recognition program that generates, evaluates, and adjusts its own operators, in *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, ACM, pp. 555–569.

Van Den Oord, A., N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves and K. Kavukcuoglu (2016), Conditional Image Generation with PixelCNN Decoders, in *Advances in Neural Information Processing Systems 29*, Curran Associates, Inc., pp. 4790–4798.

White, C. S., R. Pugatch, T. Koonce, S. W. Rust and E. Dharaiya (2008), Lung Nodule CAD Software as a Second Reader: A Multicenter Study, *Academic Radiology*, volume 15, ISSN 1076-

6332, doi:10.1016/J.ACRA.2007.09.027.

Ye, Y., Y. Tian, Y. Wu, L. Wang, Y. Chen, Z. Liu and Z. Zhang (2018), GAN Quality Index (GQI) By GAN-induced Classifier, *CoRR*.

Yi, X., E. Walia and P. Babyn (2019), Generative Adversarial Network in Medical Imaging: A Review, *Medical Image Analysis*, volume 58, arXiv ID 1809.07294v3.

Zeiler, M. D. and R. Fergus (2014), Visualizing and Understanding Convolutional Networks, in *European conference on computer vision*, Springer, Cham, pp. 818–833, arXiv ID 1311.2901v3.