ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

SCS
super computing systems

Department of Computer Science

Autumn Semester 2019

Learning tumors from OCT imagery with generative deep learning models

Bachelor Thesis

Fredin Thazhathukunnel
fredint@student.ethz.ch

February 2020

Supervisor: Pascal Kaiser, pascal.kaiser@scs.ch
Dr. Christof Bühler, christof.buehler@scs.ch
Professor: Prof. Dr. L. Thiele, thiele@tik.ee.ethz.ch

Acknowledgements

I am deeply grateful to my supervisors Pascal Kaiser and Dr. Christof Bühler (both part of Supercomputing Systems) for their supervision of the project, for their advice and their encouragement. Without Pascal Kaiser's guidance and persistent help, this thesis would not have been possible. A special word of thanks goes to Prof. Dr. Lothar Thiele for his advice, guidance and support throughout the thesis work. Furthermore, I would like to thank the company Supercomputing Systems (SCS) for offering me this opportunity and providing the necessary infrastructure. Additionally, this project was made possible with the help of Dr. Peter Maloca, Group Leader Ophthalmic Imaging at Institute of Molecular and Clinical Ophthalmology Basel (IOB), who provided the necessary Optical Computer Tomography (OCT) images.

Abstract

Medical doctors detect the presence of eye tumors with the help of Optical Coherence Tomography (OCT) scans. With the recent advance of deep learning in the field of computer vision it has become possible to automate the classification into healthy eyes and eyes with tumors. However, deep learning generally needs large sets of manually annotated ground truth data to learn from, which often forms a bottleneck.

On the one hand this thesis investigates whether a classifier's performance can be increased by augmenting the training data with data generated with generative models. On the other hand we examine if it is possible to generate realistic OCT imagery with generative models.

We trained classifiers on augmented training data sets and observed improvements in the predictive performance at times. Besides, we showed that training only on generated imagery leads to classifiers that show a comparable predictive performance as when only training on original data. We were also apt to generate OCT imagery displaying tumors. Part of the generated images have been classified by an expert as realistic OCT imagery.

Declaration of Originality

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor. For a detailed version of the declaration of originality, please refer to Appendix A.

Fredin Thazhathukunnel,
Zurich, February 2020

Contents

List of Figures

List of Tables

List of Acronyms

CNN . . . . . .Convolutional Neural Network

GANs . . . . . .Generative Adversarial Networks

OCT . . . . . .Optical Coherence Tomography

VAE . . . . . .Variational Autoencoder

Chapter 1

Introduction

1.1. Motivation

On an Optical Coherence Tomography (OCT) scan of the retina, medical doctors can detect the presence of tumors in the eyes. Still doctors need to carefully investigate the OCT imagery to determine whether a patient has an eye tumor or not. This process of analyzing and determining is very time-consuming. The manual inspection also causes variations in the results when analyzed by different doctors. Currently, there is no sophisticated algorithm for automated tumor detection in OCT imagery. One of the goals aimed by the company Supercomputing Systems (SCS) is to develop a deep learning algorithm for automated tumor detection based on OCT imagery.

SCS previously implemented a Convolutional Neural Network (CNN) approach to tackle this classification task. But the classification algorithm suffered from a lack of available ground truth data.

The first approach to overcome the shortage of limited ground truth data was to augment the data set using a classical data augmentation method, namely geometric data augmentation. Geometric data augmentation techniques apply various transformations to the original image [11]. The transformed images are later added to the training data set. These techniques include translation, rotation, mirroring of the original image and many more. Unfortunately, these methods did not succeed in increasing the performance of the classifier.

An alternative approach to increase the amount of ground truth data is to use generative models for data augmentation. The motivation behind this is that augmenting the training data with realistic but synthetic data in this manner can significantly reduce

overfitting and thus improve not only the accuracy but also the generalization ability of the underlying classification algorithm [12] [13].

1.2. Goals

To gain insight into whether synthetic ground truth data obtained from generative models can improve machine learning-based tumor detection in OCT imagery we investigate two separate problems:

1. Does classification performance improve in a benchmark setting (MNIST) when augmenting training data with data from generative models?
2. Is it possible to generate realistic OCT imagery with generative models that can deceive a medical doctor?

1.3. Approach

On one hand, we are examining whether augmenting a training data set with synthesized images improve a classifier's performance. To generate synthesized images for this proof of concept Generative Adversarial Networks are used. Generative Adversarial Networks are one of the most promising generative models for realistic image generation today [14]. Because of the complexity of the OCT imagery, this thesis demonstrates the proof of concept on the MNIST data set.

On the other hand, we are figuring out whether realistic OCT imagery can be generated by means of generative models. This paper chooses two models, namely Variational Autoencoder (VAE) and Generative Adversarial Networks (GANs) to investigate this question.

1.4. Outline

The second chapter of this thesis provides information about OCT imagery, the anatomy of the eye and then presents some related previous results. Furthermore, the chapter gives background knowledge regarding the chosen generative models. Chapter three gives an overview of the experimental data that is used. In chapter four, we will present the machine learning methods used for the results of this thesis. Moreover, chapter five presents the results which are discussed in chapter six. The seventh chapter concludes this report and mentions potential future work.

Chapter 2

Theory

The following chapter presents the theoretical background which is needed to understand this report. A brief introduction to OCT imagery and the anatomy of the eye is provided, followed by an explanation of the two generative models used in this study, namely Variational Autoencoder and Generative Adversarial Networks.

2.1. Eyes and OCT Imagery

Optical Coherence Tomography (OCT) is a two- or three-dimensional imaging technique, which is used in several medical fields. OCT performs high-resolution imaging of the internal microstructure in biological tissues [1]. Thus OCT is used in Ophthalmology in order to obtain high-resolution images of the retina and to detect and diagnose eye diseases at an early stage. OCT imagery is also applied in dermatology for the diagnosis of skin cancer and other dermatological diseases [15].

2.1.1. Technology of OCT

In low-coherence interferometry, the light emerging from a light source is directed onto a beam splitter. There it is divided into a reference beam and a measurement beam. The reflected light from the sample is interfered with reflected light from the reference arm (which has travelled a known distance) and detected with a photodetector at the output of the interferometer [1] [16]. Three dimensional images are then generated from the input of the photodetector. The underlying architecture of an interferometer is shown in figure 2.1.

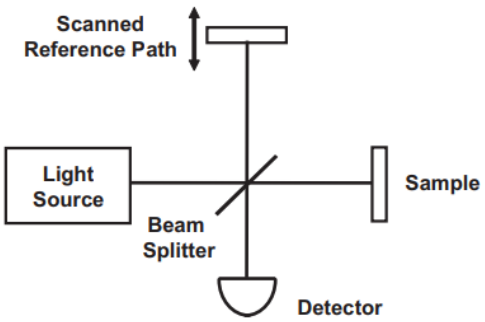

Figure 2.1.: Architecture of an interferometer. [1]

2.1.2. Anatomy of the Eye

Figure 2.2 shows the anatomy of the eye. In this thesis we investigate the retina, the macula and the choroid, which are highlighted in figure 2.2 with a black box.

- The **retina** is responsible for converting the light which comes through the lens into neural signals. The neural signals from photoreceptor cells are sent via the optic nerve to the brain in order to create visual perception [17].
- The **macula** is part of the retina and makes high-acuity vision possible. Additionally, the macula contains a high density of cones, which are photoreceptors with high acuity [18].
- The **choroid** supplies the outer retina with oxygen and nutrients with the help of its blood vessels. It is also responsible for 85% of the total blood flow in the eye [19].

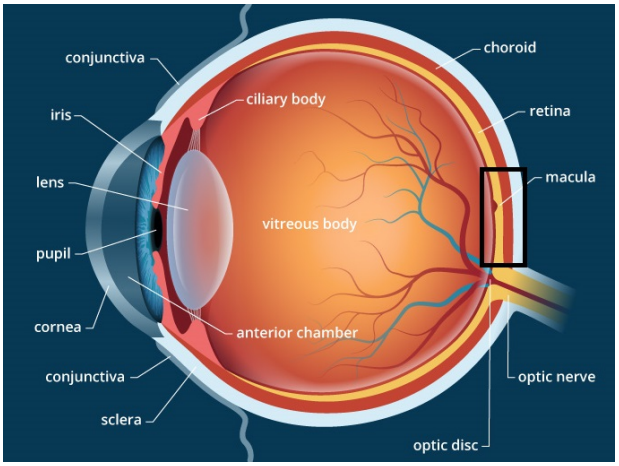

Figure 2.2.: The anatomy of the eye. The black box shows the area which is presented in an OCT scan [2].

2.1.3. OCT images of the Eye

Figure 2.3 shows an OCT scan of the retina. For a better understanding we have included the labeled image next to the OCT image. In this OCT scan the orange layer presents the vitreous and the retina is indicated with the blue label. The choroid is shown with the yellow label and the sclera is presented with the white label.

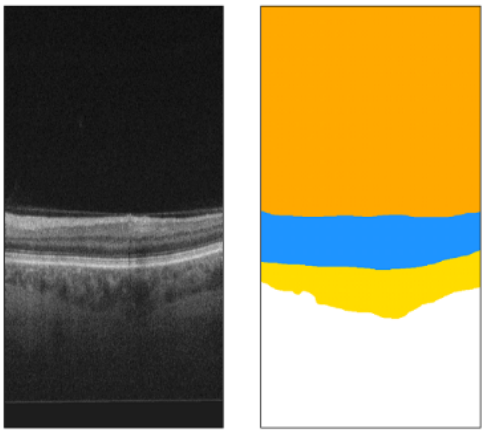

Figure 2.3.: OCT image of a healthy eye from the data set provided by Dr. Peter Maloca.

2.1.4. Eye Tumors

Eye tumors damage vision and may spread to the optic nerve, the brain and the rest of the body. There are two types of primary tumors which may arise in the eye, namely retinoblastoma and melanoma. Retinoblastoma is a cancer of the retina and arises in children. Melanoma occurs mostly in adults and occurs from uncontrolled growth of cells called melanocytes [20].

Eye tumors can be detected on an OCT scan of the retina. The tumor is located in the choroid. As it can be seen in figure 2.4 there are small choroid lesions visible. In the OCT imagery, the surrounding of the tumor has a higher light intensity compared to the blood vessels of the choroid. Thus, irregularities of the blood vessels in the choroid often indicate the appearance of tumors.

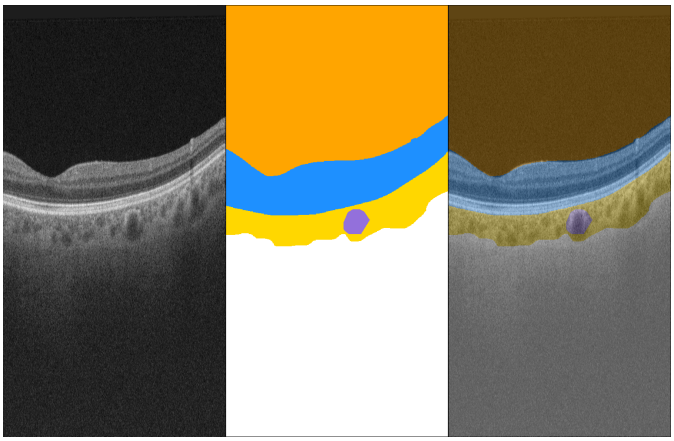

Figure 2.4.: OCT image displaying a tumor. For a better understanding we have shown the different labels of the compartments. The tumor is indicated with a violet label, the vitreous with a orange label, the retina with a blue label, the choroid with a yellow and the sclera with a white label. [3].

2.2. Related Work

2.2.1. Synthetic Data Augmentation using Generative Adversarial Networks (GANs)

In [4] Frid-Adar et al. propose a training scheme in which they first enlarge the training set using classical data augmentation and further augment the training set using synthetic images generated by Generative Adversarial Networks (GANs). They apply the proposed technique on 182 computed tomography (CT) images of liver lesions. For the GAN the authors have followed the architecture proposed by Radford et al. in [21].

The results of their experiment are given in figure 2.5. Using no augmentation, the classifier achieved an accuracy of 57%. One can recognize that the performance improved as the number of training examples increased. The classifier reaches a saturation at about 78.6%. From this point, increasing the data set using classical data augmentation techniques failed to improve classification accuracy. Since the saturation starts at 5000 samples per fold, they choose to augment the original data set with 5000 images per fold generated using classical augmentation methods and 3000 synthesized images per fold using GANs. The accuracy of the classifier improved from 78.6% with no synthesized images to 85.7% with synthesized images.

The data set in this thesis is between one and two orders of magnitudes larger in size than in [4].

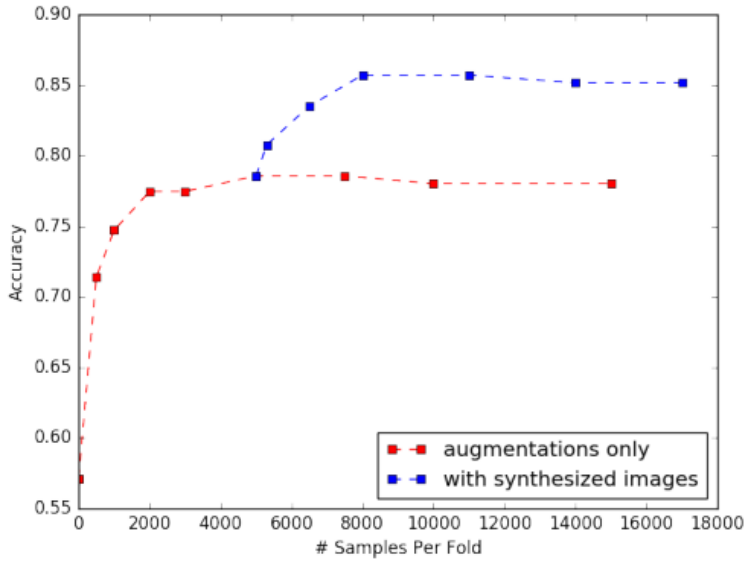

Figure 2.5.: The accuracy of a classifier trained on the augmented data set with increasing training set size. The red curve shows the classifiers accuracy using classical data augmentation and the blue curve shows the classifiers accuracy using synthesized images. They augmented the initial data set with 5000 images per fold generated using classical augmentation methods and 3000 synthesized images per fold using GANs [4].

2.2.2. Realistic OCT images with GANs

To our knowledge there is only one application of GANs for the synthesis of OCT imagery of the retina, namely [5]. The related work indicates the importance of understanding the native probability distribution of OCT representation of retinal diseases. This may

lead to a more in depth understanding of particular diseases and their pathology. In [5] the authors were able to generate realistic images depicting various retinal diseases such as macular holes or cystoid macular edema. The authors have used a database of 500'000 images after augmentation. Some of their generated images are shown in figure 2.6.

In contrast to the related work we try to capture tumors in the choroid only. Since the amount of OCT imagery displaying tumors is limited, we will develop a GAN which needs fewer images to converge.

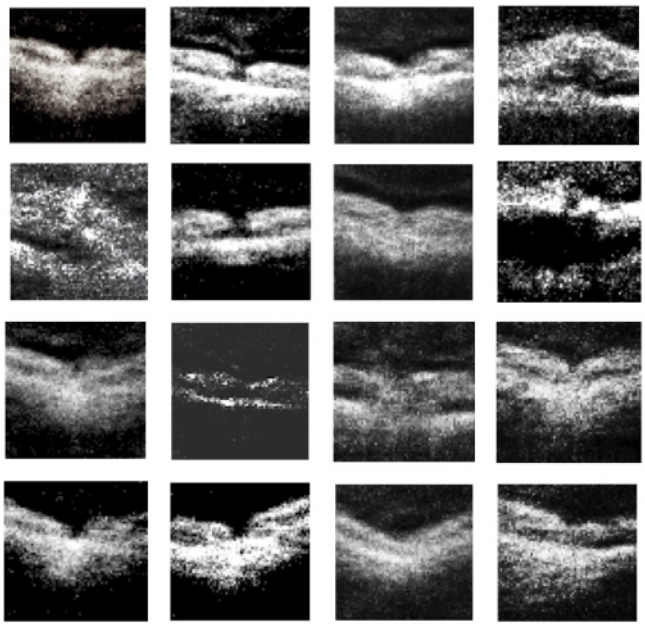

Figure 2.6.: AI generated images using a database of 500'000 images after augmentation [5].

2.3. Artificial Neural Networks

Artificial Neural Networks are computational processing systems which are inspired by the way a human brain operates. They are built by high numbers of computational nodes, which are called artificial neurons [6]. Each neuron calculates its output using a weight vector $w = (w_1, w_2, ..., w_n)$, input values $(x_1, x_2, ..., x_n)$ and an activation function φ as follows:

$$f = \varphi\Big(\sum_{i=1}^{n} w_i x_i\Big)$$

The basic structure of an artificial neural network is shown in figure 2.7. During training, a loss function quantifies the quality of an artificial neural network's predictions by measuring the difference between predicted and true values for an instance of the training data. The weights of the neurons are optimized by minimizing the loss. In gradient-based algorithms, backpropagation computes the gradient of the loss function and then updates the weights of the neurons such that the loss is reduced.

Convolutional Neural Networks are a form of artificial neural networks. The main characteristic of convolutional neural networks is that they are used in the field of digital image processing in order to solve tasks such as image classification or image recognition. In [6], the basic architecture and common terms of a convolutional neural network are described. Furthermore, [22] describes how convolutions are applied to digit classification. In [23], the authors discuss how deep learning methods, such as convolutional neural networks, have pushed the limits of what is possible in the domain of digital image processing.

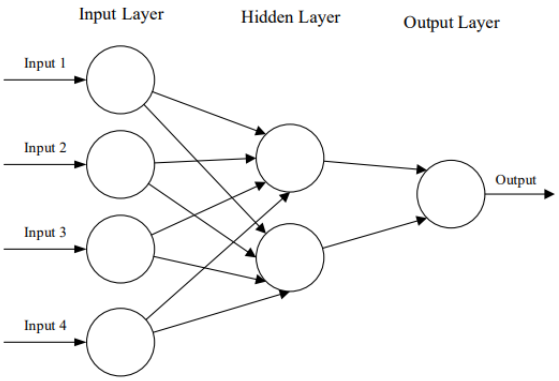

Figure 2.7.: Simple feedforward neural network consisting of an input layer, a hidden layer and an output layer. An artificial neural network is a set of artificial neurons and their connections to each other. [6].

2.4. Generative Modeling

In machine learning, one can make a distinction between discriminative and generative approaches. Generative models try to model the joint probability distribution of an observable random variable X and a target variable Y, namely $P(X = x, Y = y)$. Unlike generative modeling, discriminative modeling studies the conditional probability $P(Y = y|X = x)$. In other words a discriminative approach attempts to estimate the probability that an observation x of the random variable X belongs to a sample y of the random variable Y.

Discriminative models make fewer assumptions about the distribution compared to generative models. For example, given a set of labeled pictures of dogs and cats, a discriminative model tries to learn $P(Y|X)$ from the training data and calculates at prediction time for a new, unlabeled picture x the probability $P(Y|X = x)$ and usually determines the most likely class y to be the prediction. However, a generative model outputs a generated picture along with a class label based on the joint probability distribution $P(X, Y)$ which it learned during the training process. This joint probability distribution generally contains implicit knowledge about the underlying data like that all cats have whiskers [24].

„A generative model describes how a data set is generated in terms of a probabilistic model. By sampling from this model, we are able to generate new data "[25]. That means generative models are able to generate new instances which are similar to the data samples the model has seen during training.

2.5. Variational Autoencoder

2.5.1. The Standard Autoencoder

An autoencoder network is a pair of two neural networks, namely an encoder and a decoder. These models try to learn a compressed representation $c \in \mathbb{R}^m$ of the input data $x \in \mathbb{R}^n$. The encoder network takes in the input x from an $n-$dimensional space and maps it into a $m-$dimensional subspace. The subspace is often called latent space. This means the latent space has a lower dimensionality than the input space. The output of the encoder is called encoding or latent representation.
The goal of the decoder network is to reconstruct a representation from the latent representation that is as close as possible to the original input of the encoder [7].

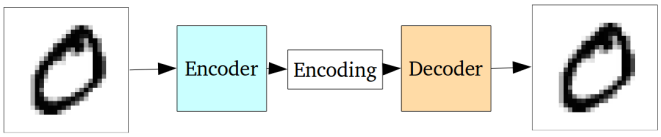

Figure 2.8.: Overview of an autoencoder [7]

The basic structure of an autoencoder is shown in figure 2.8. The objective of the autoencoder is to minimize the difference between every input and every output, i.e. the reconstruction error. Assuming the encoder function is denoted as $c = g_\Theta(x)$ and the decoder function is denoted as $x' = f_\theta(c)$ the reconstruction error of an autoenocder can be formulated as stated in equation 2.1. The parameters of the encoder, namely Θ and

the parameters of the decoder, namely θ are usually learned concurrently during training in order to minimize the reconstruction error [26]:

$$\mathcal{L}(\Theta, \theta, x) = \frac{1}{N} \sum_{i=1}^{N} (x^{(i)} - f_\theta(g_\Theta(x^{(i)})))^2 \tag{2.1}$$

Autoencoders are used in areas, where the interest lies in a dimensionality reduction while still preserving the most important features.

The problem with standard autoencoders

In figure 2.9, one can see the clusters representing the different digits in the two-dimensionial latent space for the MNIST dataset. This is sufficient in order to replicate images. However, with generative models, one wants to generate new samples by randomly sampling from the latent space. If the latent space has discontinuities, meaning gaps between the clusters and one samples from there, as indicated with „?“ in figure 2.9, the decoder may generate an unrealistic output. The decoder does not know how to decode that region of the latent space. During training, the decoder never receives encoded vectors coming from that region of the latent space [7].

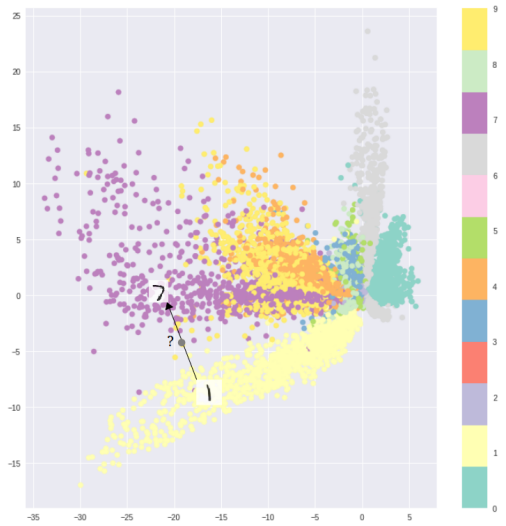

Figure 2.9.: Autoencoder on the MNIST dataset, visualizing the latent representation of each digit from a 2D latent space. [7]

2.5.2. The Variational Autoencoder

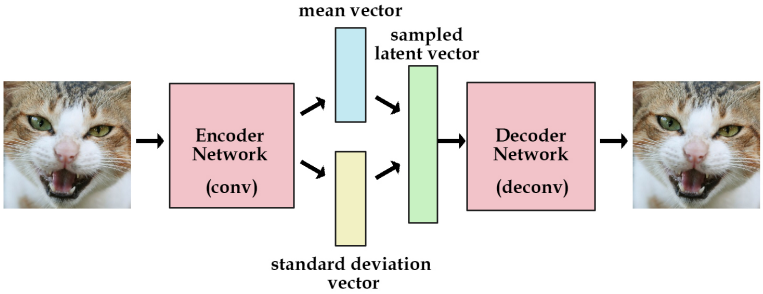

Figure 2.10.: Overview of a variational autoencoder [8]

Unlike vanilla autoencoders, Variational Autoencoders (VAE) are generative models. The VAE tries to overcome the shortcoming of a continuous latent space. The encoder of the VAE does not directly generate a latent vector of a data sample x. It rather provides a mean vector $\mu(x)$ and deviation vector $\sigma(x)$ of size m, whereby m is the dimension of the latent space. Moreover, the latent representation z is sampled from the Gaussian distribution parameterized by $\mu(x)$ and $\sigma(x)$. Then the sampled encoding is passed onward to the decoder. Otherwise, the decoder of the VAE works similar as the decoder of the autoencoder.

For the same input, while the mean and standard deviation remain the same, the actual encoding will vary due to this sampling procedure of the latent representation. As encodings are sampled randomly from the distribution, the decoder learns that not only a single point in latent space refers to a sample of that class, but all nearby points refer to the same class as well. The decoder is exposed to various encodings of the same input during training. This property allows the decoder not just to know the decoding of single specific points in latent space but also the decoding of latent points that slightly vary [7].

Objective of VAE

What we ideally want is to maximize the probability of each x in the training set under the entire generative process, according to equation 2.2:

$$p(x) = \int P(x|z)\, P(z)\, dz \tag{2.2}$$

„Provided powerful function approximators, we can simply learn a function which maps our independent, normally-distributed z values to whatever latent variables might be needed for the model, and then map those latent variables to x.“ [9] Thus we choose for the prior $P(z) = \mathcal{N}(0, I)$, where I is the identity matrix.

To approximate $p(x)$ we can sample a large number of z values $(z^{(1)}, z^{(2)}, ..., z^{(n)})$ and then compute

$$p(x) \approx \frac{1}{n} \sum_{i} P(x|z^{(i)}) \tag{2.3}$$

Unfortunately, in high dimensional spaces the number of samples n has to be extremely large in order to approximate $p(x)$ accurately [9].

The idea behind the variational autoencoder is to sample values of z that are likely to be produced by x. Thus, we need a conditional probability distribution $P(z|x)$, which takes a value of x and returns a distribution over z that are likely to have produced x. Since the calculation of $P(z|x)$ using Bayes rule involves $P(x)$ we can not compute $P(z|x)$ directly. Thus, we try to approximate $P(z|x)$ by another distribution $Q(z|x)$ which is defined in such a way that it has a tractable solution: $Q(z|x)$ is considered to be Gaussian distributed.

To achieve this approximation we can use the Kullback-Leibler (KL) divergence between the $Q(z|x)$ and $P(z|x)$. The KL divergence between two probability distributions measures how much they diverge from each other. Minimizing the KL divergence means optimizing the probability distribution parameters $\mu(x)$ and $\sigma(x)$ to resemble that of the target distribution [7] [27]:

$$D_{KL}(Q(z|x)||P(z|x)) = E_{z\sim Q(z|x)}\Big[log\, Q(z|x) - log\, P(z|x)\Big] \tag{2.4}$$

After simplifying equation 2.4 using Bayes rule and moving $P(x)$ out of the expectation, since the expectation is over z, one receives equation 2.5.

$$log\, P(x) - D_{KL}[Q(z|x)||P(z|x)] = E_{z\sim Q(z|x)}[log\, P(x|z)] - D_{KL}[Q(z|x)||P(z)] \tag{2.5}$$

The left-hand side of the equation describes what we want to optimize, namely $P(x)$ and an error term which causes Q to produce z values that can reproduce x. The right-hand side is the objective function of the variational autoencoder. The first term represents the reconstruction's likelihood and the second term ensures that the distribution Q is similar to our prior $P(z)$ [9].

Image Generation

In order to generate new instances we can simply input values $z \sim \mathcal{N}(0, I)$ into the decoder. This works since the first few layers of the decoder will learn a function which maps the independent normally-distributed z values to the latent variables needed to construct x'. This x' was likely to have created that particular z.

Reparametrizaton trick

The problem which occurs in the above architecture is that backpropagation is not possible. Note that we are sampling a latent vector from the mean and standard deviation vector, as shown on the left-hand side of figure 2.11. Such a sampling procedure is not differentiable and thus one cannot backpropagate through the nodes. The reparametrization trick tries to move the non-differentiable operation out of the network, as shown on the right-hand side of figure 2.11. Now the gradients do not need to flow through the random node since it has no learnable parameters. The feedforward behaviour of both networks is the same but backpropagation can only be applied if the reparametrization trick has been used.

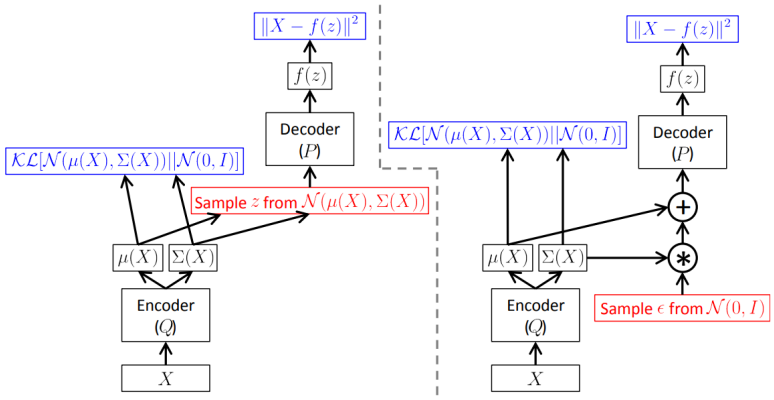

Figure 2.11.: Reparametrization trick which is needed for the back propagation. Left is without reparametrization trick, and right is with it. The distribution $N(0, I)$ is the prior $P(z)$. [9]

Multivariate Gaussian distribution

There are several reasons why we choose a multivariate Gaussian distribution for the latent space. First of all, we can apply the reparametrization trick here. Secondly, to generate new images we can simply input values $z \sim \mathcal{N}(0, I)$ into the decoder. Another reason for choosing Gaussian distribution is to evaluate the KL-divergence analytically.

2.6. Generative Adversarial Networks

A Generative Adversarial Network (GAN) is a generative framework recently proposed by Ian Goodfellow and his co-authors in [28]. According to Yann Lecun, „Generative Adversarial Networks is the most interesting idea in the last ten years in machine learning.“

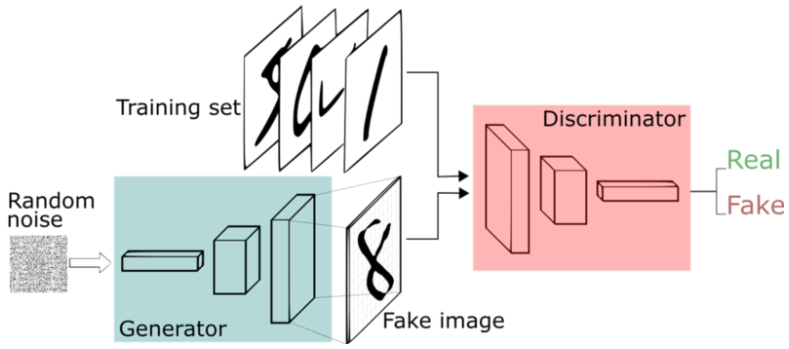

Figure 2.12.: Overview of a Generative Adversarial Network [10]

In a GAN architecture we have a **discriminator** and a **generator**, as it is shown in figure 2.12. They both are built as neural networks. The task of the generator is to generate new images as close as possible to the training data. Whereas, the task of the discriminator is to classify whether the received images come from the training data or were generated. Furthermore, the goal of the generator is to fool the discriminator by providing generated images and causing the discriminator to erroneously classify them as genuine. The goal of the discriminator is to maximize the probability of assigning the correct label to both training examples and samples generated by the generator [28].

As shown in figure 2.12, the input for the generator is a random noise vector. This random noise vector is either uniform or Gaussian distributed. The generator takes this random noise and transforms it into an image, which is then given to the discriminator, whereas the discriminator is trained using the training set. Moreover, the training set contains real images. Additionaly, the discriminator tries to classify whether the image seen is real or fake.

The discriminator is trained the same way as a binary image classifier. Goodfellow mentioned that GANs use the power of discriminative models and their benefits to get a good generative model. Furthermore, the discriminator is trained with the backpropagation algorithm. In the case of images generated by the generator, the gradients are further backpropagated through the generator. In this manner, the generator may learn how to generate new images which look more realistic to the discriminator.

Objective of GANs

The discriminator D and generator G are trained simultaneously. $D(x)$ represents the probability that sample x comes from the distribution of the training data rather than the distribution of the generated images. The input of the generator is the random noise vector $z \sim p_z$, whereby p_z is a prior on input noise variables. Furthermore, the discriminator wants to maximize $D(x)$ over the training data and the generator wants to maximize $D(G(z))$ over the random noise variables. This is equivalent to maximizing the logarithm of $D(x)$ over the training data and minimizing the logarithm of $log\,(1 - D(G(z)))$ over the random noise variables. The discriminator and the generator play the following minimax game with value function $V(D, G)$ [28]:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))] \tag{2.6}$$

Nevertheless, often one does not use equation 2.6 to train a GAN. Early in learning, the generated images will be roughly random and thus the discriminator can reject these samples with high confidence. In this case, $log(1 - D(G(z)))$ saturates. But we can train G to maximize $log\,D(G(z))$. Then the objective function provides much stronger gradients early in training [28].

The generator implicitly defines a probability distribution p_g as the distribution of the samples generated. It has been proven in [28] that for a fixed generator the optimal discriminator is:

$$D^*_{G(x)} = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \tag{2.7}$$

and for a fixed discriminator the optimal generator for the GAN objective becomes

$$p_g = p_{data} \tag{2.8}$$

Chapter 3

Data

In this chapter we present the data used for the experiments done in this thesis. We give a short overview of the training and test sets used for the neural networks.

3.1. Proof of Concept

The proof of concept was done using the MNIST data set. The MNIST data set is a collection of handwritten digits with a training set of 60'000 images and a test set of 10'000 images. The digits have been size-normalized and centered in a fixed-size image of 28×28 [29].

3.1.1. Experiment data

For the experiments a training subset and a test subset of the original MNIST data set were created. The subsets were obtained by randomly choosing the desired amount of distinct images from the original training and test set, respectively. We have done the experiment using different amounts of images. Hereby, we have chosen intentionally the same amount of distinct images for each digit. The different training subsets had 15, 30, 50, 100, 250, 500 and 1000 images per digit. The test subset used in all experiments had a size of 500 images per digit.

3.2. Generative Models

For training the generative models we had used the OCT imagery provided by Dr. Peter Maloca, Group Leader Ophthalmic Imaging at Institute of Molecular and Clinical Ophthalmology Basel (IOB). The OCT imagery had already been classified by SCS into *Emmetropia, Hyperopia, Myopia* and *Tumor* images. Emmetropia is the normal refractive condition of the eye, in which vision is sharp and thus no corrective lenses are needed. Hyperopia, also known as farsightedness is the state of vision in which close objects appear blurry. Myopia, also known as nearsightedness is the state of vision where distant objects appear blurry.

3.2.1. Variational Autoencoder

Experiment data

In order to train the variational autoencoder OCT scans of emmetropic eyes are used. We take OCT scans of the left eye of 20 random patients. Each OCT scan provides 256 images. Randomly 15 OCT scans are taken as the training set and 5 OCT scans for testing. In other words the training set consists of 3840 images and the test set of 1280 images.

OCT images of emmetropic eyes are chosen because compared to the other classes (*Hyperopia*, *Myopia* and *Tumor images*) the images of emmetropic eyes have a more similar and simpler structure. Thus, it is probably easier for the model to capture the desired distribution.

3.2.2. Generative Adversarial Networks

Experiment data

For training the generative adversarial network we first use a training set containing the training and test set used for the variational autoencoder. In other words, the training set for the GAN contains 5120 OCT images of emmetropic eyes. For training a GAN no test set is required. After optimizing the model to work on OCT imagery of emmetropic eyes we train a second model on OCT imagery displaying tumors. Hereby, the training set contains totally 1730 OCT images, which were obtained from left eyes as well as from right eyes of the patients.

Chapter 4

Methods

This chapter aims to provide a clear and complete explanation of the experimental steps undertaken in this thesis. First, we explain the experimental environment followed by the methods which were used for the proof of concept. Secondly we provide an explanation of the design techniques applied to the generative models.

4.1. Experiment environment

All models developed in this thesis were trained and evaluated on a NVIDIA Titan X graphics card, which was provided by Supercomputing Systems. The implementation was done using Python. Keras was used as the framework for the neural networks implemented in this thesis. TensorFlow was chosen to serve as the backend engine of Keras.

4.2. Proof of concept

As mentioned in section 1.1, the motivation for this thesis is to improve the performance of an automated tumor detection algorithm by augmenting the training data. In order to achieve this, we have to examine whether the augmentation of a training set with synthesized images generated by means of generative models leads to an improvement of the classifier. Investigating this hypothesis on the OCT imagery would exceed the scope of this thesis. Thus, we decided to provide a proof of concept on the MNIST data set.

4.2.1. Procedure

First, a subset of the MNIST data set is chosen in order to train the GAN. Subsequently, the subset is augmented with the generated images using the GAN. At this point, we have two data sets. One data set containing only the original images and a second data set containing the original images and the generated images. Afterwards, we train a CNN for MNIST digit classification on both of these data sets.

4.2.2. Implementation

GAN for MNIST

The architecture of the GAN trained on the subset of the training set is shown in figure 4.1 and in figure 4.2. Each convolution layer in the discriminator uses a stride of two, 4×4 kernel size and padding is set to *same*, which causes the image size to be halved after every convolution. After every convolution layer, we apply Batch Normalization and *LeakyReLU* activation. The single node in the output layer uses *sigmoid* activation since the discriminator needs to output probabilities.

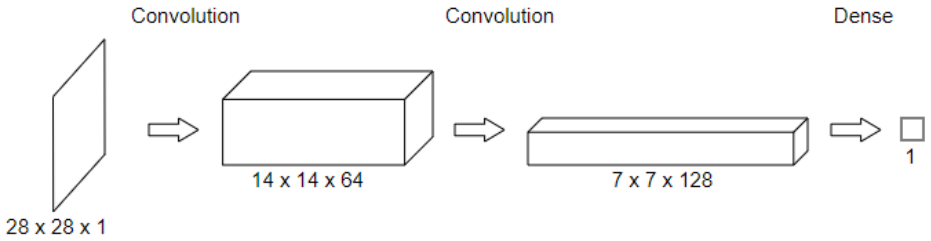

Figure 4.1.: Architecture of the discriminator used for training the GAN on the MNIST data set. The input is an image with one channel and 28×28 pixels in size. The output is a binary classification.

The first hidden layer of the generator needs enough neurons for multiple activation maps of the output image. This thesis has chosen to have 128 activation maps with a size of 7×7. This leads to a total number of $7 \cdot 7 \cdot 128 = 6272$ neurons. In order to upsample the low-resolution image we use transposed convolution layers with a stride of two, 4×4 kernel size and padding is set to *same*. Again, Batch Normalization and *LeakyReLu* are used after the transposed convolution layers, except for the last one. In the last convolution layer, we do not use Batch Normalization and the used activation function is *tanh* which squashes the output values to the open interval $[-1, 1]$.

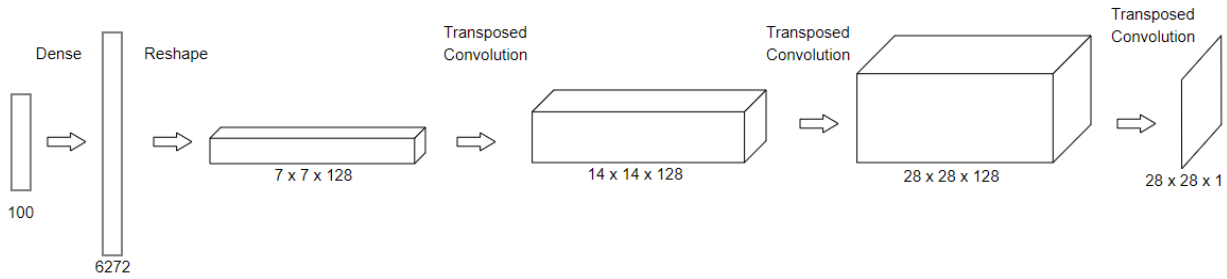

Figure 4.2.: Architecture of the generator used for training the GAN on the MNIST data set. The input of the generator is a 100 element vector drawn iid from a standard normal distribution. The output is a grayscale image of 28×28 pixel.

CNN for MNIST

After generating synthesized handwritten digits we augment the original training subset used for training the GAN with the generated images. Now one can compare a classifier's performance between the data set containing original images only and on the augmented data set. In order to have a more robust prediction performance comparison the classifier is trained and tested for each training set five times and performance measures are averaged.

For image classification, we use the CNN shown in figure 4.3. In order to downsample the image we use MaxPooling layers with a stride of two, kernel size of 4×4 and padding is set to *same*. The activation function used for each layer except for the last one is *relu*. For the output layer we use *softmax* activation.

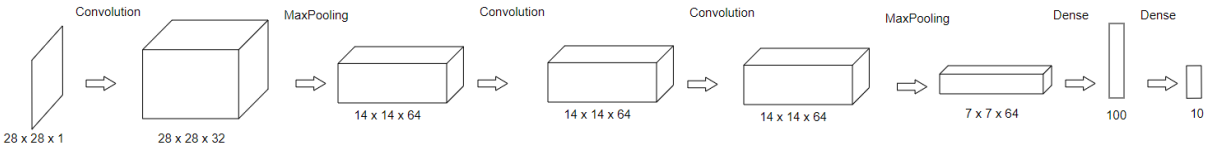

Figure 4.3.: Architecture of the CNN used for image classification. The input is an image with one channel and a size of 28×28 pixels. The output is the predicted digit.

4.3. Generative Models

In order two answer the question of whether one can generate realistic OCT imagery using generative models, we choose to investigate on two models, namely Variational Autoencoder (VAE) and Generative Adversarial Network (GAN).

4.3.1. Procedure

We train both generative models using the OCT imagery in order to generate new realistic images.

4.3.2. Implementation

Variational Autoencoder

The architecture of the encoder is shown in figure 4.4 and of the decoder in figure 4.5. Each convolution layer of the encoder uses *relu* activation, a kernel size of 3×3 and *padding* = *same*. For the purpose of downsampling the image, we use MaxPooling layers with a stride and a kernel size of two.

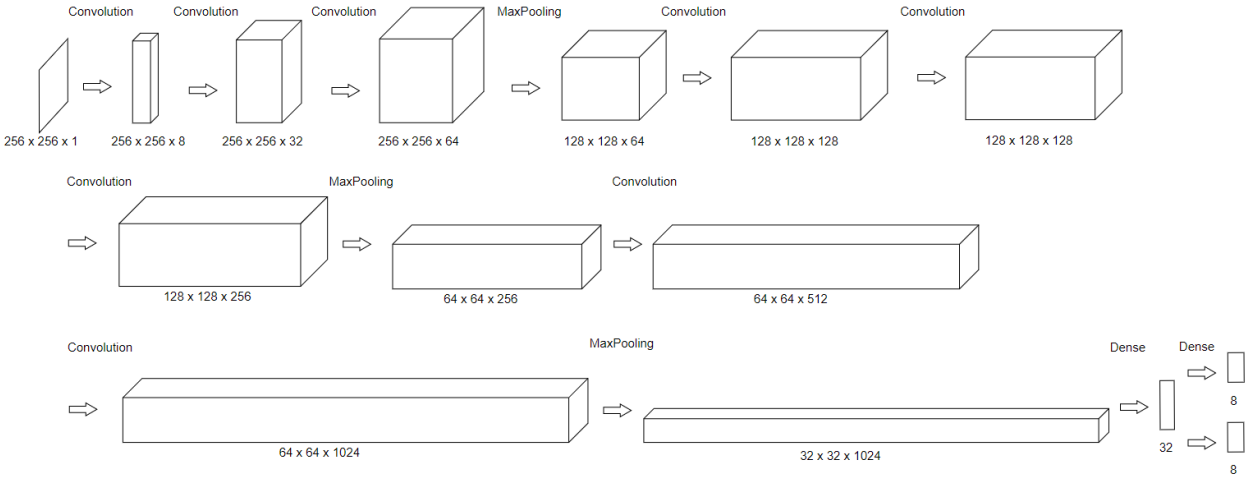

Figure 4.4.: Architecture of the VAE encoder. The input is a grayscale image x with one channel and a size of 256×256 pixels. The outputs are two vectors, namely $\mu(x)$ and $\sigma(x)$.

As explained in 2.5.2, the variational autoencoder uses the vectors $\mu(x)$, $\sigma(x)$ and $\epsilon \sim \mathcal{N}(0, I_8)$ in order to sample a latent representation z of the input image x. This latent representation is the input for the decoder of the variational autoencoder. We have chosen

for the first convolution layer of the decoder to have 32 channels of the low-resolution image with a spatial dimension of 32×32. This causes the first hidden layer of the network to have $32 \cdot 32 \cdot 32 = 32768$ neurons. For upsampling, we are using transposed convolution layers with a stride of two, a kernel size of 3×3 and zero padding. Again, the activation function used in the convolution layers is *relu*.

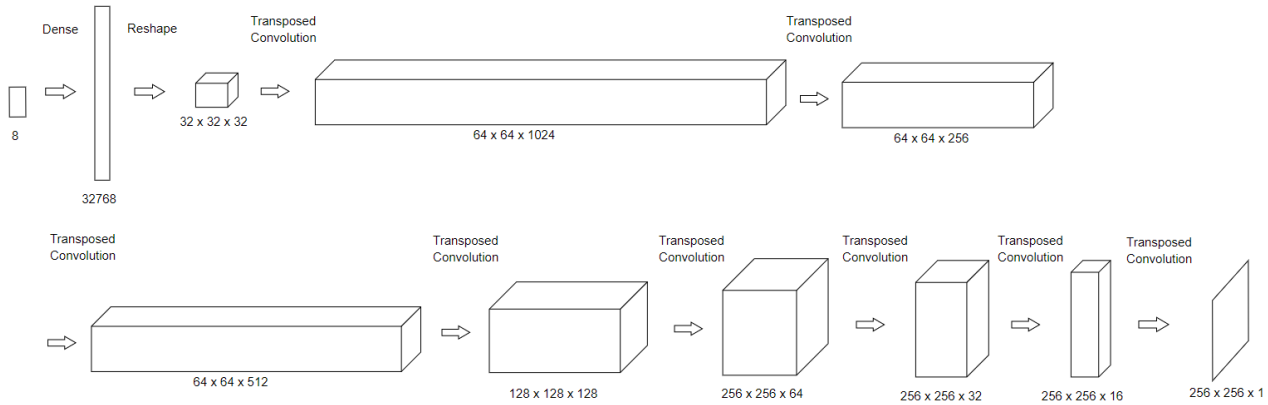

Figure 4.5.: Architecture of the VAE decoder. The input is the latent representation of an input image x sampled from the vectors $\mu(x)$ and $\sigma(x)$. The output of the decoder is the reconstructed representation of the original image.

In [30] the authors propose a new framework named ***β* - VAE** for learning a disentangled latent representation of an image. The framework β - VAE qualitatively outperforms VAE for disentangled factor learning. This can be achieved by forcing the components of the latent representation z to be independent. To encourage independence we weight the KL-divergence term in the objective function of the VAE with a factor β, as shown in equation 4.1:

$$\mathcal{L} = E_{z \sim Q(z|x)}[log\, P(x|z)] - \beta D_{KL}[Q(z|x)||P(z)] \tag{4.1}$$

In this thesis we have used a value of $5 \cdot 10^{-4}$ for β.

Generative Adversarial Networks

For the sake of training a GAN on the OCT imagery the architectures shown in figures 4.6 and 4.7 are used. Once more each convolution layer in the discriminator uses a stride of two, which causes the image size to be halved. Every convolution layer is followed by a Batch Normalization and *LeakyRelu* activation. The main differences of the discriminator used for the OCT imagery (figure 4.6) and the discriminator used for the MNIST data set (figure 4.1) are on the one hand the number of convolution layers and their number of filters and on the other hand the additional techniques used in order

to improve the stability of the network.

MinibatchDiscrimination is a recently suggested technique to improve the training for GANs, which was proposed in [31]. One of the common difficulty when training GANs is mode collapse. As a result of mode collapse the generator creates only a limited diversity of samples. MinibatchDiscrimination tries to overcome mode collapse by computing the similarity of an image with images in the same batch. If the generator is outputting similar images the discriminator can detect a mode collapse using the similarity score and prevent a mode collapse by penalizing the generator.

Another technique proposed in [31] to improve the GAN is **one-sided label smoothing**. One-sided label smoothing replaces the 0 and 1 labels in the GAN architecture with smoothed values, like 0.9 or 0.1. It has been shown that this technique reduces the vulnerability of neural networks to adversarial examples [32]. This technique is called one-sided because only the positive labels are smoothed to 0.9, leaving negative labels set to 0, since smoothing negative labels barely cause an improvement.

In [33] Chintala proposes to use **noisy labels** to improve generalization and stability of the trained neural network. To achieve noisy labels we flip the labels of the real and fake images with a certain probability.

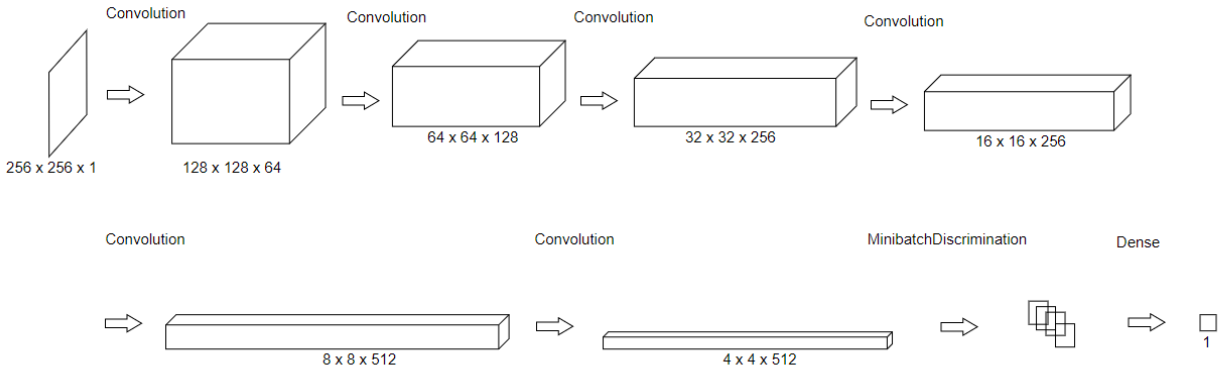

Figure 4.6.: Architecture of the discriminator used for training the GAN. The input is an image with one channel and 256×256 pixels in size. The output is a binary classification.

For the first convolution layer of the generator we have chosen to have 512 low-resolution channels with a spatial dimension of 4×4. Thus, the first hidden layer of the network needs to have $4 \cdot 4 \cdot 512 = 8192$ neurons. We use eight transposed convolution layers with a stride of two in order to upsample the image to a size of 256×256. The last two transposed convolution layers do not use any strides, leaving the image size unchanged.

Batch Normalization and *LeakyReLu* is used after each transposed convolution layer, except for the last one. In the last convolution layer no Batch Normalization is used and the used activation function is *tanh*.

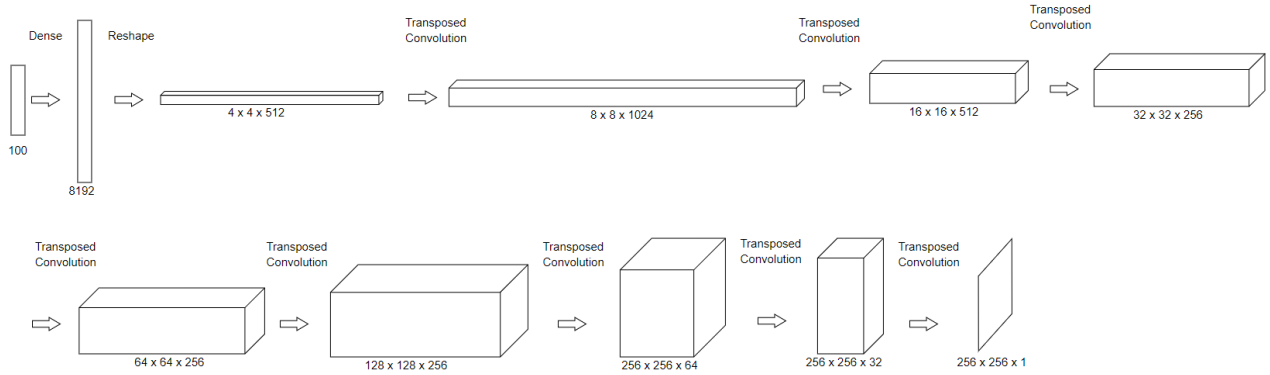

Figure 4.7.: Architecture of the generator used for training the GAN.The input of the generator is a 100 element vector of elements draw iid from a standard normal distribution. The output is a grayscale image of 256×256 pixel.

Chapter 5

Results

In this chapter we present the results obtained by using the data mentioned in chapter 3 and applying the methods explained in chapter 4.

5.1. Proof of concept

As described in section 4.2, first a subset of the MNIST data set is chosen in order to train the GAN. Afterwards we augment the training subset with the images generated using the GAN. As a result we are able to train the classifier for the handwritten digits on two data sets, namely the original training subset and the augmented data set.

The first column of table 5.1 shows how many images per digit the original training subset contains. This original training subset is used for training the GAN. The second column provides the number of images per digit generated using the GAN. The classifier is then trained on the original training subset and on the augmented training set.

For the columns three to five the accuracy is the mean accuracy after training and testing the model for five times.

The third column presents the classifier's performance on the original training subset and the fourth column states the classifier's performance on the augmented data set. The fifth column reveals the classifier's performance when trained only on generated images and tested on original images. The last column is the performance gain respectively the performance loss through augmenting the training subset with generated imagery.

Training size per digit	**generated images per digit**	**accuracy w/o augmentation**	**accuracy with augmentation**	**accuracy on only generated images**	**gain**
15	250	77.44	75.332	66.5	-2.108
30	500	83.74	82.4	69.108	-1.34
50	500	88.536	89.156	83.276	0.62
100	2000	92.164	92.78	89.368	0.616
250	2000	95.16	95.396	83.98	0.236
250	4000	95.336	95.396	92.92	0.06
500	2000	96.876	96.76	93.564	-0.116
500	5000	96.728	96.736	94.624	0.008
1000	4000	97.776	97.7	94.76	-0.076

Table 5.1.: Results obtained from training the classifier on different training sets.

In figure 5.1 the classifier's predictive performance by means of cross-entropy loss and accuracy is shown. The classifier has been trained and tested five times. In blue, the classifier's learning process on the augmented training set is presented. In orange, we display the classifier's performance on a test set, which is a subset of the original test set of the MNIST data set.

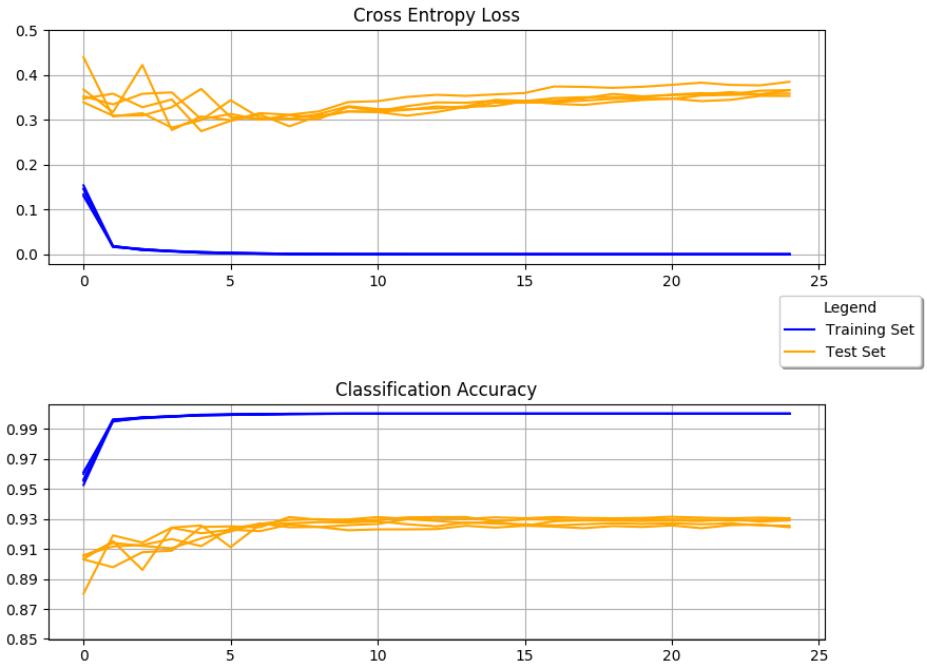

Figure 5.1.: The classifiers performance on the test subset (orange) and the classifiers learning process on the augmented training set (blue) are shown. The original training subset contains 100 images per digit and 2000 images per digit are generated with the GAN. The x-axis describes in both plots the number of epochs the classifier was trained. The y-axis represents in the first plot the loss and in the second plot the accuracy.

In figure 5.2 the boxplots of the classifier's performance on three different training sets is provided. Each boxplot presents the accuracy resulting from the 5 trainings done on that particular training set. The leftmost boxplot shows the result of training the classifier on 100 original images per digit. In the middle boxplot the result of training the classifier on 100 original and 2000 generated images per digit is provided. The rightmost boxplot presents the result of the classifier when trained on 2000 generated images per digit.

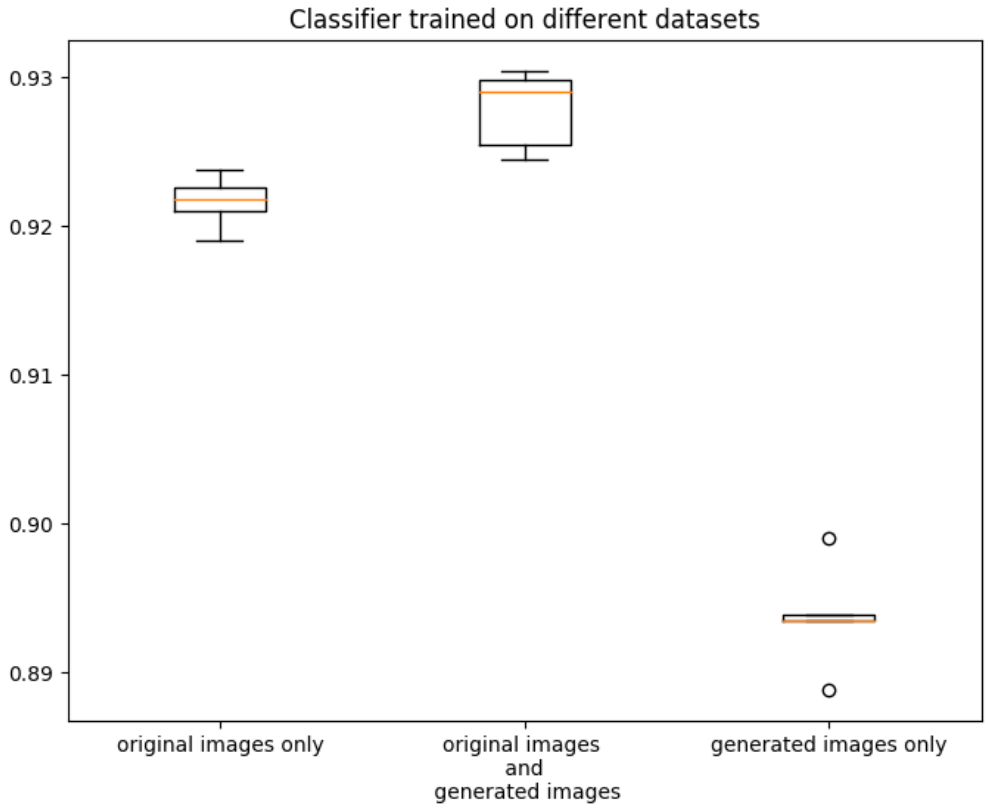

Figure 5.2.: Boxplots showing the accuracy of three different training sets. On each training set the classifier has been trained 5 times, which is represented by each boxplot.

5.2. Generative Models

For generating realistic OCT imagery we trained the models using the architecture explained in chapter 4.

5.2.1. Variational Autoencoder

Figure 5.3 shows three OCT images of an emmetropic eye, generated using a variational autoencoder. We trained the model for 300 epochs and had a validation loss of 0.4547.

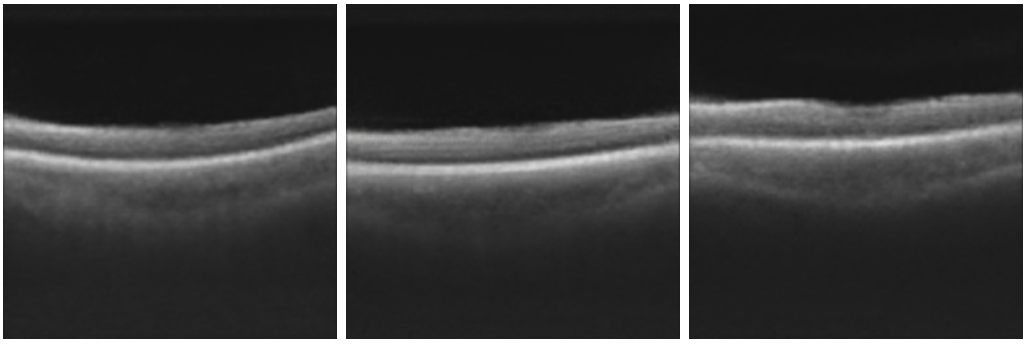

Figure 5.3.: OCT images of emmetropic eye generated using VAE.

5.2.2. Generative Adversarial Networks

Figure 5.4 shows OCT images of an emmetropic eye, generated using a generative adversarial network.

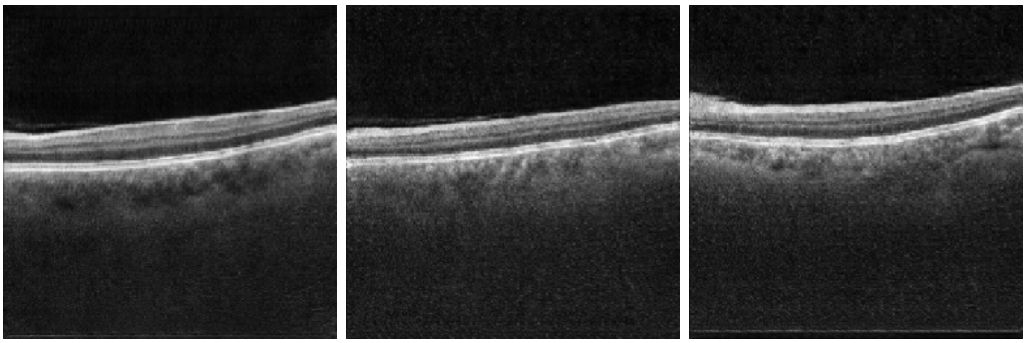

Figure 5.4.: OCT images of emmetropic eye generated using GAN

Figure 5.5 summarizes the predictive performance of the generative adversarial network by means of loss and accuracy. The model was trained on 5120 OCT images of emmetropic eyes. The GAN was trained for around 386 Epochs with a batch size of 16 which results in 117800 single training steps.

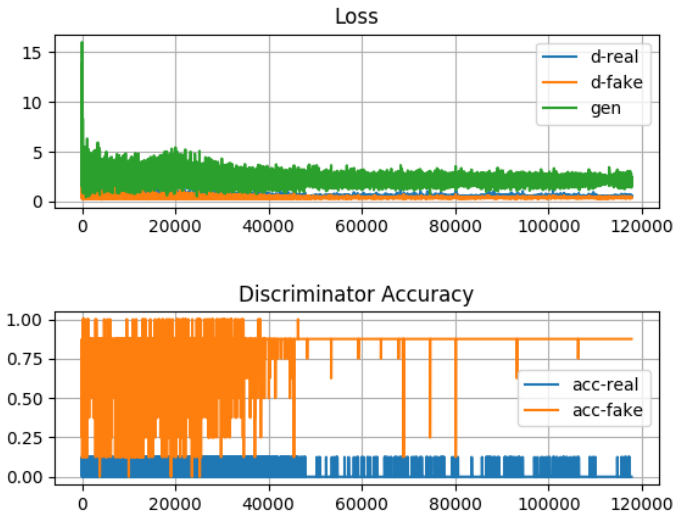

Figure 5.5.: GANs predictive performance performance by means of loss and accuracy on OCT imagery of emmetropic eyes. The x-axis shows in both plots the number of single training steps and the y-axis present in the first plot the loss and in the second plot the accuracy.

Figure 5.6 shows OCT images of an eye tumor generated using a generative adversarial network. The surrounding of the tumor shows a high light intensity. One can also detect a vanishing shadow beneath the tumor in the OCT images.

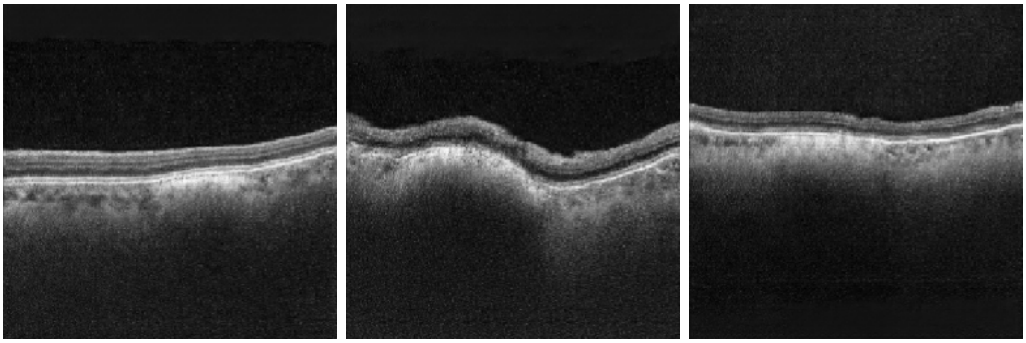

Figure 5.6.: OCT images displaying eye tumor generated using GAN

Figure 5.7 summarizes the performance of the generative adversarial network trained on 1730 OCT images displaying tumors. The GAN was trained for 300 epochs with a batch size of 16, which results in 54000 steps.

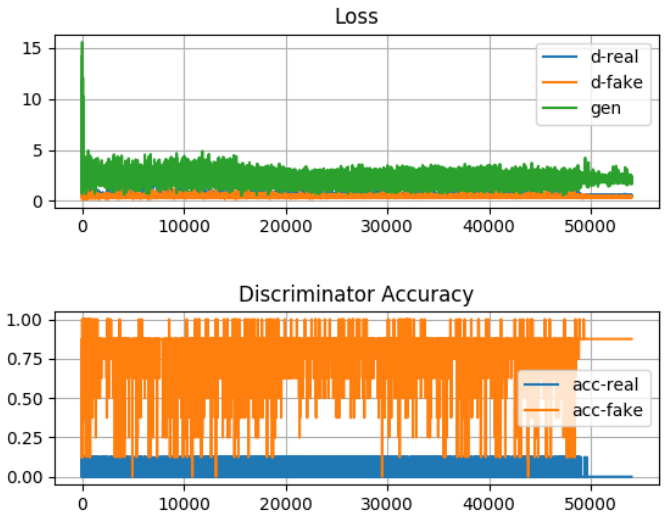

Figure 5.7.: GANs performance on OCT imagery displaying tumors. The GAN was trained for 54000 steps. The x-axis shows in both plots the number of single training steps and the y-axis present in the first plot the loss and in the second plot the accuracy.

Chapter 6

Discussion

In this chapter, we analyse the results presented in chapter 5. In particular, we discuss possible causes for the results received.

6.1. Proof of Concept

6.1.1. Augmenting training set with generated imagery

By augmenting the training set with synthesized images using generative models our goal was to increase the amount of representations and to ultimately improve a classifiers performance. The motivation was that generative models may learn the distribution in an other manner than a discriminative classifier and thus may provide more information about the data. As it can be observed in table 5.1 the approach led to a relatively modest improvement of the classifier in most of the cases. One possible explanation for having only such a marginal improvement may be that we did not try to overcome imbalanced data. In particular, when dealing with imbalanced data there are majority classes and minority classes. Then one tries to oversample the minority classes. In our setting we had the same amount of images for every class and tried to augment the training set with generated images for every class. In other words, every class was a minority class in our setting.
In [34] Tanaka and Aranha came to a similar conclusion to ours. They have reported a marginal increase in the performance of the classifier although dealing with imbalanced data sets. Since, the increase is known to be minimal augmenting the data set with synthesized images generated with generative models might not be sufficient in order to improve the classifier by significant amounts.

6.1.2. Training on generated images only

As presented in figure 5.2 training a classifier on generated images only, results in a similar predictive performance as when training the classifier on the original data set. Since, it is possible that training data contains sensitive information which could be misused it can be desirable to train an algorithm on realistic synthetic data. As shown in this thesis training on generated images only, is possible at the cost of a relative small loss of accuracy of the classifier. The ability of providing such comparable performance is also an indicator for the success of learning. In particular, this ensures that our generative model was apt to learn the underlying distribution and sample from the learnt distribution.
In [34] training on generated images only even outperformed the classifier's performance compared to training on the original data set.

6.2. Generative Models

6.2.1. Variational Autoencoder

As it can be seen in figure 5.3 the OCT image generated with the Variational Autoencoder is blurry. Bluriness is a common issue for Variational Autoencoders. Goodfellow et al. state in [35] that the causes of this phenomenon is not known yet. One possibility is that the bluriness is caused by minimizing the KL-divergence in equation 2.5. As described in [35] the KL-divergence is asymmetric. The version of the approximation used for the Variational Autoencoder will assign a high probability to points that occur in the training set but it may also assign a high probability to other points. These other points may include blurry images [35].

6.2.2. Generative Adversarial Networks

With the help of Generative Adversarial Networks it was possible to synthesize realistic OCT imagery, in particular OCT images displaying eye tumors as shown in figure 5.6. The generated OCT imagery has been presented to an expert, namely Dr. Peter Maloca Group Leader Ophthalmic Imaging IOB. His expertise was used to quantify the quality of the generated imagery. From the exposed 25 generated images the expert classified 16 images as realistic OCT images displaying tumors. Nevertheless, on all 25 generated images one can identify the four compartments (vitreous, retina, choroid and sclera). The expert mentioned that these could be OCT images which were generated by an older Imaging System due to the low local resolution and a size of 256×256 pixels of the images. An other imperfection is that an Optical Coherence Tomography Imaging System would generate an OCT scan which is axially stretched for better readability whereby the generated OCT imagery using Generative Adversarial Networks are square-shaped.

Nevertheless, regarding the goodness of the generated imagery the GAN outperformed the VAE by far.

Chapter 7

Conclusion and Future Work

In this chapter we first conclude our findings and then provide some prospective points for the future work of this research.

7.1. Conclusion

In this thesis we come to the conclusion that augmenting a training set with generated images by means of generative models leads to a marginal increase in the performance of a classifier in most of the cases. Thus, increasing the amount of OCT ground truth data by means of generative models alone might not be sufficient enough to increase a classifiers performance.

In this work we have identified that the images generated using Variational Autoencoder are blurry. This is even the case for emmetropic eyes where the model had enough training data. We have also proven that Generative Adversarial Networks are sufficiently advanced to generate OCT imagery displaying tumors which have been classified by an expert as realistic. Although the GAN had a training set containing only 1730 OCT images displaying tumors it is able to generate realistic OCT imagery with eye tumors. Thus, we suggest the use of Generative Adversarial Networks for synthetic image generation since it outperformed the Variational Autoencoder by far.

7.2. Future Work

Since, in this thesis the impact of augmenting a training set was analyzed on the MNIST data set a possible first future step might be to examine the effect of augmenting the training set consisting of OCT imagery. Furthermore, one may use the OCT images

generated in this work for this purpose.

As described in 2.2.1 one can increase a classifiers performance and overcome the imbalance of data by using a combination of classical augmentation techniques and generative models. This may lead to a new hypothesis which can be investigated in the field of OCT imagery.
An other possibility to overcome the lack of ground truth data would be to use an other augmentation technique such as SMOTE [36]. This kind of techniques have already been proven to lead to the desired result [34]. Moreover, Tanaka and Aranha state that the GAN did not perform better than augmentation techniques such as SMOTE or ADASYN for imbalanced data.

Appendix A

Declaration of Originality

Bibliography

[1] J. Fujimoto and W. Drexler, *Introduction to Optical Coherence Tomography*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1–45. [Online]. Available: https://doi.org/10.1007/978-3-540-77550-8_1

[2] L. Segre, "Eye anatomy: A closer look at the parts of the eye," 2019, [Online; accessed 11-December-2019]. [Online]. Available: https://www.allaboutvision.com/resources/anatomy.htm

[3] P. Friedli, "Extend machine- and deep learning into 3d volumetric analysis of medical image data," Semester Thesis, ETH Zürich, 6 2019.

[4] M. A. J. G. H. G. Maayan Frid-Adar, Eyal Klang, "Synthetic data augmentation using gan for improved liver lesion classification," 2018. [Online]. Available: https://arxiv.org/pdf/1801.02385.pdf

[5] M. M. S. Stephen G. Odaibo, M.D., "Generative adversarial networks synthesize realistic oct images of the retina," 2019. [Online]. Available: https://arxiv.org/pdf/1902.06676.pdf

[6] R. N. Keiron O'Shea, "An introduction to convolutional neural networks," 2015. [Online]. Available: https://arxiv.org/pdf/1511.08458.pdf

[7] I. Shafkat, "Intuitively understanding variational autoencoders," 2018, [Online; accessed 14-October-2019]. [Online]. Available: https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

[8] K. Frans, "Variational autoencoders explained," 2016, [Online; accessed 15-October-2019]. [Online]. Available: http://kvfrans.com/variational-autoencoders-explained/

[9] C. Doersch, "Tutorial on variational autoencoders," 2016. [Online]. Available: https://arxiv.org/pdf/1606.05908.pdf

[10] T. Silva, "An intuitive introduction to generative adversarial networks (gans)," 2018, [Online; accessed 23-October-2019]. [Online]. Available: https://www.freecodecamp.org/news/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394/

[11] H. Kumar, "Data augmentation techniques," [Online; accessed 8-December-2019]. [Online]. Available: https://iq.opengenus.org/data-augmentation/

[12] "Data augmentation using generative adversarial networks (cyclegan) to improve generalizability in ct segmentation tasks."

[13] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, 2018, pp. 117–122.

[14] "Systematic analysis of image generation using gans."

[15] T. M. P. E. A. Mette Mogensen, Lars Thrane and G. B. E. Hemec, "Oct imaging of skin cancerand other dermatological diseases," *Journal of BIOPHOTONICS*, 2009. [Online]. Available: https://doi.org/10.1002/jbio.200910020

[16] S. A. B. James G Fujimoto, Costas Pitris and M. E. Brezinski, "Optical coherence tomography: An emerging technology for biomedical imaging and optical biopsy," 2000, [Online; accessed 10-December-2019]. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1531864/#FN1

[17] Healthline, "Retina," 2015, [Online; accessed 11-December-2019]. [Online]. Available: https://www.healthline.com/human-body-maps/retina#1

[18] Wikipedia, "Macula of retina," 2019, [Online; accessed 11-December-2019]. [Online]. Available: https://en.wikipedia.org/wiki/Macula_of_retina

[19] "Anatomy and regulation of the optic nerve blood flow."

[20] W. E. Institute, "Eye tumors," [Online; accessed 11-December-2019]. [Online]. Available: https://www.hopkinsmedicine.org/wilmer/conditions/tumors.html

[21] S. C. Alec Radford, Luke Metz, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2016. [Online]. Available: https://arxiv.org/pdf/1902.06676.pdf

[22] P. Sermanet, S. Chintala, and Y. LeCun, "Convolutional neural networks applied to house numbers digit classification," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, 2012, pp. 3288–3291.

[23] A. C. S. H. G. V. H. L. K. D. R. J. W. Niall O' Mahony, Sean Campbell, "Deep learning vs. traditional computer vision," 2019. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1910/1910.13796.pdf

[24] "Discriminative model," 2019, [Online; accessed 17-December-2019]. [Online]. Available: https://en.wikipedia.org/wiki/Discriminative_model

[25] D. Foster, *Generative Deep Learning.* O'Reilly Media, Inc., 2019, chapter 1.

[26] A. M'Charrak, "Deep learning for natural language processing (nlp) using variational autoencoders (vae)," Master Thesis, ETH Zürich, 10 2018.

[27] A. Kumar. Deep learning 22: (4) variational autoencoder : Derivation of the loss function. [Online]. Available: https://www.youtube.com/watch?v=Hlr3CYfRMf0

[28] M. M. B. X. D. W.-F. S. O. A. C. Y. B. Ian J. Goodfellow, Jean Pouget-Abadie, "Generative adversarial nets," 2014. [Online]. Available: https://arxiv.org/pdf/1406.2661.pdf

[29] C. J. B. Yann LeCun, Corinna Cortes, "The mnist database." [Online]. Available: http://yann.lecun.com/exdb/mnist/

[30] A. P. C. B. X. G. M. B.-S. M. A. L. Irina Higgins, Loic Matthey, "$\beta - vae$: Learning basic visual concepts with a constrained variational framework," 2017.

[31] W. Z. V. C. A. R. X. C. Tim Salimans, Ian Goodfellow, "Improved techniques for training gans," 2016. [Online]. Available: https://arxiv.org/pdf/1606.03498.pdf

[32] D. T. Tamir Hazan, George Papandreou, "Adversarial perturbations of deep neural networks," in *Perturbations, Optimization, and Statistics.* MIT Press, 2017, pp. 311–342.

[33] S. Chintala, "How to train a gan?" 2016, [Online; accessed 30-December-2019]. [Online]. Available: https://www.youtube.com/watch?v=X1mUN6dD8uE

[34] C. A. Fabio Henrique Kiyoiti dos Santos Tanaka, "Data augmentation using gans," 2019. [Online]. Available: https://arxiv.org/pdf/1904.09135.pdf

[35] A. C. Ian Goodfellow, Yoshua Bengio, *Deep Learning.* MIT press, 2016, chapter 3 and Chapter 20.

[36] L. O. H. W. P. K. Nitesh V. Chawla, Kevin W. Bowyer, "Smote: Synthetic minority over-sampling technique," in *Journal of Artificial Intelligence Research.* Morgan Kaufmann, 2002, pp. 321–357.

[37] A. C. Y. Bengio and P. Vincent, "Representation learning: A review and new perspectives," 2014.