TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technology
Department of Software Science

Erik Dzotsenidze 204251IAPM

GENERATIVE ADVERSARIAL NETWORKS AS A DATA AUGMENTATION TOOL FOR CNN-BASED PARKINSON'S DISEASE DIAGNOSTICS

Master's Thesis

Supervisor
Elli Valla
MSc
Co-supervisor
Sven Nõmm
PhD

Tallinn 2022

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Erik Dzotsenidze .....
(signature)

Date: May 10, 2022

Annotatsioon

Parkinsoni tõbi on neurodegeneratiivne haigus, mis põhjustab patsiendil värinaid, liigutuste aeglustumist ja muutusi käekirjas. Varajane Parkinsoni tõve tuvastamine on keeruline ülesanne, kuid tähtis, et patsiendi elukvaliteeti parandada. Kaasaegsed masin õppe mudelid on aidanud luua paremaid Parkinsoni tõve tuvastamise süsteeme, aga on tihti limiteeritud saada olevate treening andmete poolt.

Lõputöö põhieesmärgiks on kasudata generatiivsed võistlusvõrke (inglise keeles *Generative Adverserial Networks* - GANs) andmete augmenteerimis meetodina. Töö käigus treeniti nelja erinevat GAN arhitektuuri, mida kasutati Archimedeuse spiraalide genereerimiseks. Treenitud GAN mudelid ja genereeritud pilte headust hinnati kvantitatiivsete ja kvalitatiivsete meetoditega. Peale seda kasutati genereeritud andmeid konvolutsiooniliste närvinõrkude treenimisel (inglise keeles *Convolutional Neural Network* - CNN). CNNid treeniti ilma augmenteerimiseta, traditsioonilise augmenteerimisega ning GAN-põhise augmenteerimisega, et võrrelda kas genereeritud andmed suudavad parandada Parkinsoni tõve tuvastamise tulemusi. Teisejärguline eesmärk oli vaadata mõju on CNN arhitektuuril Parkinsoni tõve klassifitseerimisele. Selle täitmiseks valiti kuus CNN architektuuri ja nende tulemusi võrreldi.

Lõputöö tulemused tõid esile, et StyleGAN3 ja Projected GAN poolt genereeritud tulemused olid kõige paremad. GANide genereeritud andmete kasutamise tulemusena CNN mudelite tõusis tundlikus kõrgemale kui traditsiooniliste augmentatsioonide puhul aga samas mudelite spetsiifilisus oli madalam. Projected GAN genereeritud andmed saavutasid kõige kõrgema tundlikuse 96.6%. Baasjoone tundlikus oli 94.3% ja traditsioonilise augmenteerimise tundlikus oli 93.7%. Kõikides katsetatud CNN arhitektuuridest ResNet, VGG, Inception v3 ja Xception lõpptulemused olid väga sarnased. AlexNet ja DenseNet tulemused olid halvemad.

Tulemused näitavad, et GANide kasutamine lisa andmete genereerimisel on abiks paremate Parkinsoni tõve tuvastamise süsteemide ehitamise.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 61 leheküljel, 7 peatükki, 68

joonist, 10 tabelit.

Abstract

Parkinson's disease is an neurodegenerative disease that causes the patient to have tremor, bradykinesia, and changes in writing, among other symptoms. Early detection of Parkinson's disease is a challenging task, yet an important one to improve the patients quality of life. Modern machine learning models have helped to create better Parkinson's detection systems, but are often limited by the amount of available training data.

The primary goal of this thesis is to use GANs as data augmentation to generate more training data for the downstream classification task. Four different GAN architectures were trained and used to generate additional Archimedean spiral training data. The generated images and models were then evaluated with quantitative and qualitative methods. After that the generated images were used in CNN training. CNNs trained with GAN-based augmentations were compared with CNNs trained with no data augmentation and traditional augmentations to find out if the generated data could outperform them. A secondary objective was to find what kind of impact does the CNN architecture provide on the Parkinson's detection performance. To achieve this, six different CNN architectures were chosen and evaluated.

The main results of the thesis show that StyleGAN3 and Projected GAN resulted in the best generated data. GAN generated data resulted in CNN models that have higher sensitivity than traditional augmentations, while also having lower specificity. Projected GAN generated data lead to the highest sensitivity of 96.6%, when the highest sensitivity of the baseline was 94.3% and traditional methods was 93.7%. Out of all the tested CNN architectures - ResNet, VGG, Inception v3 and Xception performed with marginal differences, while AlexNet and DenseNet performed worse.

These results show that GAN-based augmentation can be a viable method in generating additional training data to help build better Parkinson's detection models.

The thesis is in English and contains 61 pages of text, 7 chapters, 68 figures, 10 tables.

List of abbreviations and terms

ACGAN	Auxiliary Classifier Generative Adversarial Network
CNN	Convolutional Neural Network
CSV	Comma-separated values
DCGAN	Deep Convolutional Generative Adversarial Network
DST	Dynamic Spiral Test
FFHQ	Flickr-Faces-HQ Dataset, image dataset of human faces
FID	Fréchet Inception Distance
GAN	Generative Adversarial Network
HC	Healthy control
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
JPEG	Joint Photographic Experts Group, image file format
JSON	JavaScript Object Notation
KID	Kernel Inception Distance
KNN	K-Nearest Neighbor
LSUN	Large-scale Scene UNderstanding, image dataset of scenes and objects
PD	Parkinson's patient
PNG	Portable Network Graphics, image file format
ROI	Region of Interest
RGB	Red Green Blue color model
SST	Static Spiral Test
t-SNE	t-distributed Stochastic Neighbor Embedding
VAE	Variational Autoencoders

Table of Contents

List of Figures

List of Tables

1. Introduction

Parkinson's disease is a neurodegenerative disease that causes the neurons from the central nervous system to die or become damaged, causing severe disability. Symptoms of Parkinson's disease include tremor (involuntary shaking of limbs), slowed movement (bradykinesia), and changes in the person's writing, among others. As there is currently no cure for Parkinson's disease, treatment consists of managing symptoms to improve the quality of life of patients [1]. Therefore, early detection and treatment of Parkinson's disease are important for a high quality of life for the patient [2].

Studies have found evidence that neurological disorders are one of the greatest burdens on the healthcare system in the world [3]. With 6.1 million people having Parkinson's disease worldwide in 2016, the number of patients has grown 2.4 times since 1990. Making Parkinson's disease the fastest growing neurological disease currently and the number of individuals with Parkinson's is expected to double again in the next generation[4].

Parkinson's disease is diagnosed using different kinds of writing and drawing tests. These include drawing patterns such as spirals, Luria's alternating series, and handwriting tests. Today, drawing tablets and other tablet computers such as the iPad are used for digital data collection [5]. This has allowed researchers to measure many more parameters that are difficult or impossible to measure with pen and paper. For example, kinematic parameters (speed and acceleration) and other parameters such as pressure. In the past, these parameters have been used to diagnose patients with Parkinson's disease [6].

With the increase in computing power over the last decade, CNNs (convolutional neural networks) have become very popular in the field of image classification. The popularity of CNNs comes from their ability to accurately classify difficult tasks and their ability to learn to extract features automatically. CNNs have previously been used to solve the problem of identifying Parkinson's patients by handwriting and drawing test images [7, 8]. Although the results have not been perfect, there has been shown that there is potential in using CNNs to diagnose patients with Parkinson's disease.

The main issue with using CNNs is that they are data hungry. Requiring large and diverse data sets to train them properly. In the medical field, data is hard to come by, as collecting

it takes a lot of time and resources. For Parkinson's writing tests there are a handful of data sets that at most contain around a few hundred images in total per test type. To solve the problem of data scarcity simple offline data augmentation has been used [5]. In the process of data augmentation, different transformations are applied to the images, for example, rotation, mirroring, and adding noise. Offline data augmentation refers to the fact that data augmentation takes place before the training phase. Although this has helped alleviate the need for more data and increased classification accuracy, the problem has not been solved.

Over the last few years new generative neural networks like GAN (Generative Adversarial Network) [9] and VAE (Variance Auto-encoder) [10] models have become popular, mostly because of their ability to generate never before seen images out of training data. Using these types of architecture, it could be possible to generate new Parkinson drawing test images using the data that is available right now, after which the generated images could be used to train the CNN classification model.

As GANs have only become popular in the last few years, they have not been applied to Parkinson's patient handwriting and drawing generation. Therefore, it would be important to study whether these methods could help alleviate the problem of data scarcity and improve Parkinson's disease classification performance. A more accurate classification would help reduce the resources and time needed to diagnose a patient and support physicians in the diagnosis process.

1.1 Related work

In [7], transfer learning with CNNs and data augmentation on the spiral test images was applied to diagnose Parkinson's disease. Combining multiple data sets (HandPD, NewHandPD, Parkinson's drawings), they managed to achieve a validation accuracy of 99% using AlexNet. Results that high are often met with skepticism in the medical field. They split their data set only into two (training, validation) and not three (training, validation, test), it is hard to say how unbiased their final results are. Otherwise, they give a nice overview of the impact that different data sets and augmentation methods have on classification performance.

Another study used six AlexNet models with a majority voting approach for Parkinson's detection, where each CNN classified one type of handwritten drawing test [11]. For their experiments, they used the HandPD data set. Their approach obtained an accuracy of 93.5%. They also considered standard classifiers, but found that CNN pre-trained on ImageNet data set produce the best single-model performance. Their single-model performance for Archimedean spirals was 78%. Furthermore, the training on Archimedean

spirals results in the CNN model having trouble classifying healthy controls.

Furthermore, [12] used data augmentation and enhanced their digital images of Parkinson's handwritten drawings with pressure information to train their CNN. For training, they used spiral pentagon and cube drawings for data collected by clinicians at Leeds Teaching Hospitals NHS Trust. Using ten-fold cross-validation, they achieved accuracy of 93.5%.

Image enhancement and data augmentation have also been used in [8] to train CNN (AlexNet) for Parkinson's disease detection. The work used DraWritePD data set and enhanced the images with pressure and velocity parameters. AlexNet was used as a classification model and achieved accuracy of 88.2% without any tuning.

In 2014, [9] designed a new framework for generative neural networks called the generative adversarial network. This new framework used an adversarial process to train the model, which meant training two models (generator and discriminator) simultaneously, similarly to a two-player minimax game. The adversarial training allows GANs to create outputs that have much better quality than those of other models, which often produce blurry results. Furthermore, the training process for GANs is unsupervised, which removed the need for data labeling in the training process, making it more straightforward from a data collecting view. However, training two models simultaneously made the training process much more unstable. Additionally, the first GAN models needed a lot of real input data for training and produced low-resolution images.

In the image classification domain, GANs have been shown to improve classification performance. A conditional GAN called ACGAN (Auxiliary Classifier Generative Adversarial Network) has been used to generate image of insects [13]. Using the generated images increased their CNN classifier F1-score from 92% to 95%.

Generators have also been applied in the low-shot learning domain to hallucinate new instances of novel concepts [14]. The work focusses more on metalearning, or in other words, the concept of learning to learn. Even so, they show that using a generator to hallucinate new images increases the accuracy of classification. This indicates that there is potential in using generated data as part of training data to improve classification performance.

Improvements have also been made in the medical image classification domain. In [15], unconditional DCGAN (Deep Convolutional Generative Adversarial Network) was used to synthesise images of liver lesions and was able to improve the accuracy of the CNN classification model by 7.1%. Their original data set contained 182 2-D CT scans from

which ROIs (region of interest) of the lesions were extracted. They used traditional data augmentation to find the optimal training data set for training the CNN model and used this training set to train the GAN models.

Furthermore, [16] generated chest X-ray images from GANs and used VGG for abnormal chest X-ray detection. They saw an increase in test accuracy of around 1% compared to traditional augmentation.

1.2 Problem statement

The goal of this thesis is to investigate GANs as a data augmentation tool in a limited data scenario. More specifically, to generate Parkinson's patients' drawing images. Furthermore, to evaluate if the generated data could be used in a downstream task. The secondary goal is to analyse the difference between CNN architectures, in terms of the performance of classifying Parkinson's patients' drawings. In the process of completing the thesis, the following questions should be answered:

- Can generative neural networks be used to generate meaningful Parkinson's handwriting and drawing images?
- How does the addition of GAN-generated data affect CNN model performance?
- Which CNN architectures perform the best with classifying Parkinson's patients' drawings?

In this thesis, GANs are used to generate images of Archimedean spirals drawn by PD (Parkinson's patients) and HC (healthy controls). The generated images and generative models are then evaluated with both quantitative and qualitative methods.

Compare the affect of generated images on the image classification task. Two sets of baseline classification models are trained. One is trained on only real images, and the other is enhanced by additional images that have been augmented by traditional methods. The trained GAN models are then used to generate additional images for the training set, and the classification models are trained on these training sets. The results are then compared to the baseline models to see how the generated images affected the results of the classification task.

1.3 Structure

The thesis is structured as follows. Chapter 2 describes in detail each data set used in the thesis. Chapter 3 explains the theory behind GANs and the specific GAN architectures used in the experiments for image generation. Chapter 4 provides an overview of the CNN architectures utilised for image classification. Chapter 5 describes the experimental workflow and hyperparameters used in each step. Chapter 6 analyses the result of image generation and classification and offers a discussion of the results.

2. Materials

Total of five different Parkinson's digital handwriting data sets containing Archimedean spirals are used in this thesis. The following sections will describe the data collection and content of these data sets.

2.1 DraWritePD data set

DraWritePD contains digitized data of different drawing tests collected with a Apple iPad Pro (2016) and an Apple Pencil. A specially developed application was used to collect coordinates of the pencil tip and pressure applied to the screen up to 240 times per second. This data was stored in a JSON file as a time series, where each coordinate and pressure reading is tied to a timestamp. The assembled testing group contained 58 volunteers of which 24 were Parkinson's patients (mean age 74.1 ± 6.7) and 34 healthy controls with the same age (mean 74.1 ± 9.1) gender distribution. Each subject completed a series of 12 exercises and answered questions so that the practitioner could evaluate the subject's condition.

In this work, 48 Archimedean spirals collected during the data acquisition are used. Of these 48 spirals, 19 were healthy controls and 29 Parkinson's patients. To turn the time series data into images, each data point was plotted as a point on an image. Examples from the data set can be seen in Figure 1.

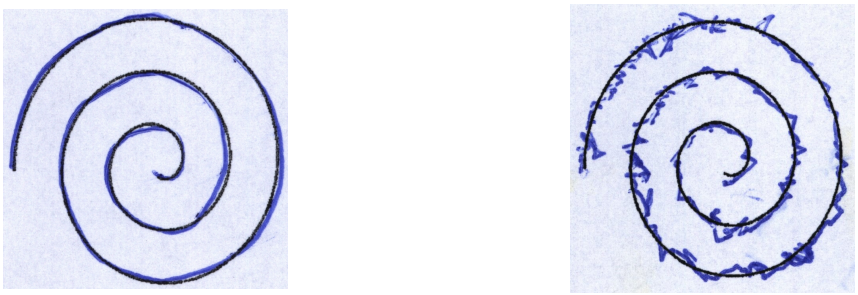
(a) Healthy control (b) Parkinson's patient

Figure 1. DraWritePD data set examples.

2.2 HandPD & NewHandPD data set

HandPD is a set of handwritten drawing tests collected at the Botucatu Medical School, São Paulo State University - Brazil [17]. The data set contains data from 92 individuals: 18 healthy controls and 74 Parkinson's patients. The healthy controls consisted people in the ages ranging from 19 to 79 years old (mean 44.22 ± 16.53) and Parkinson's patients in the age range of 38 to 78 (mean 58.75 ± 7.51). Each person filled out a paper form with templates of four Archimedean spirals and meanders. After that each test was cropped from the form and stored as an image in JPEG format.

In total, HandPD contains 368 images of Archimedean spirals with 72 images being from the healthy control group and the other 296 from the Parkinson's patients. Examples from the data set can be seen in Figure 2.



(a) Healthy control

(b) Parkinson's patient

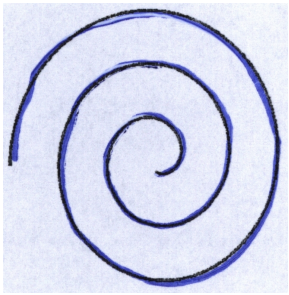
Figure 2. HandPD data set examples.

NewHandPD is an improved version of HandPD [18]. The data set is composed of 66 individuals: 35 healthy controls in the age range of 14 to 79 (mean 44.05 ± 14.88) and 31 Parkinson's patients with ages ranging from 38 to 78 years old (mean 57.83 ± 7.85). Data collection was carried out on a paper form like HandPD. In addition to the tests carried out in HandPD, NewHandPD also contains data on circle movements and left- and right-handed diadochokinesis. Furthermore, the dynamics of the handwriting was recorded using BiSP smart pen.

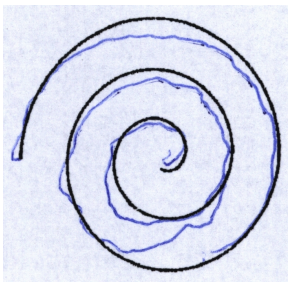
NewHandPD contains 264 images of Archimedean spirals. Of those images 140 are healthy controls and 124 Parkinson's patients. Examples from the data set can be seen in Figure 3.

Both HandPD and NewHandPD are publicly available from the HandPD data set home page[1].

[1] `https://www2.fc.unesp.br/~papa/pub/datasets/Handpd/`



(a) Healthy control



(b) Parkinson's patient

Figure 3. NeHandPD data set examples.

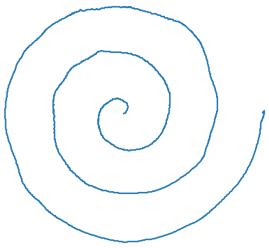
2.3 ParkinsonHW data set

ParkinsonHW data set is a digitized Archimedean spiral data set collected using the Wacom Cintiq 12WX graphics tablet. The collection was carried out at the Department of Neurology of the Cerrahpasa Faculty of Medicine, Istanbul University [19, 20]. The data set consists of data from 62 Parkinson's patients and 15 healthy controls, in total, 77 individuals. In the data set, there are three types of drawing tests. The first being the SST (Static Spiral Test); in this test, individuals are asked to retrace the Archimedean spiral visible on their tablet as closely as possible with a digital pen. The second test is DST (Dynamic Spiral Test); in this test, the spiral template appears and disappears in a certain time interval. The third test is STCP (Stability on Certain Point), where the subject must hold their digital pen above a point in the middle of the screen without touching the screen for a certain amount of time. The data for every individual was stores in a separate file, where the time series data was delimited as CSV values. In the data set they collected the coordinate values, pressure applied to the screen and the angle of the digital pen.

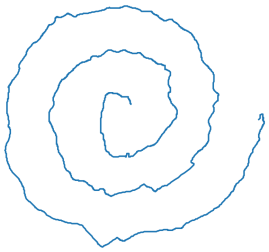
This thesis uses both the SST and DST data sets. As with the DraWritePD data set, each data point in the time series is plotted to an image. The SST data set consists of 76 images of which 15 are healthy controls and 61 are Parkinson's patients. The DST data set consists of 72 images, of which 15 are healthy controls and 57 are Parkinson's patients. Examples from the SST data set and DST data set can be seen in Table 4 and Table 5.

The data set is publicly available at UCI machine learning repository [2].

[2] `https://archive.ics.uci.edu/ml/datasets/Parkinson+Disease+Spiral+Drawings+Using+Digitized+Graphics+Tablet#`

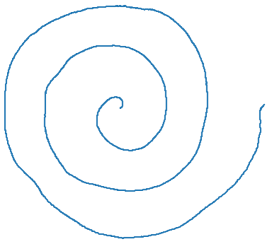


(a) Healthy control

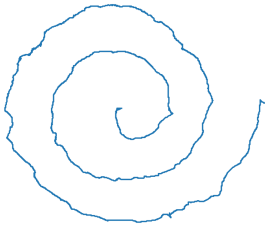


(b) Parkinson's patient

Figure 4. Isenkul SST data set examples.



(a) Healthy control



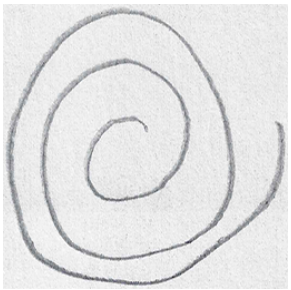
(b) Parkinson's patient

Figure 5. Isenkul DST data set examples.

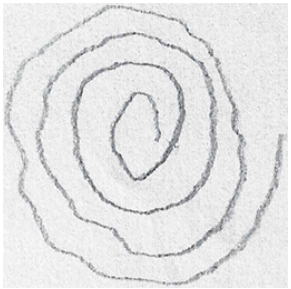
2.4 Parkinson's Drawings data set

Parkinson's Drawings data set is a set of Archimedean spirals and waves [21]. The data set contains data from 55 subjects: 28 healthy controls (mean age 71.32 ± 7.21) and 27 Parkinson's patients (mean age 71.41 ± 9.37). The data set consists of spirals and wave patterns drawn on paper and cropped into individual files.

In total there are 102 images of Archimedean spirals: 51 healthy controls and 51 Parkinson's patients. Examples from the data set can be seen in Table 6.



(a) Healthy control



(b) Parkinson's patient

Figure 6. Parkinson's Drawings data set examples.

Parkinson's Drawings data set is publicly available from Gaggle [3].

2.5 Summary

A summary of the data sets can be found in Table 1. In total, 930 images of Archimedean spirals were collected. The data collected includes both digitized spirals that are plotted using software and photographs of spirals drawn on paper. The collected data is unbalanced and has a ratio of $1:2$, healthy control images to Parkinson's patient images.

Table 1. Archimedean spiral data set summary.

Name	**HC images**	**PD images**	**Total size**
DraWritePD	19	29	48
HandPD	72	296	368
NewHandPD	140	124	264
ParkinsonHW SST	15	61	76
ParkinsonHW DST	15	57	72
Parkinson's Drawings	51	51	102
Total	**312**	**618**	**930**

[3] `https://www.kaggle.com/kmader/parkinsons-drawings`

3. Generative Adversarial Networks

GANs (Generative Adversarial Networks) were designed in 2014 [9]. The network consists of two parts, generator G and discriminator D. G uses a noise variable z as input from the distribution p_z (latent space) and produces an output in the form of an image that is in the generated distribution p_g. D takes in a image x and outputs the probability that x came from the real data distribution p_{real} rather than p_g. Figure 7 shows the general architecture of GANs.

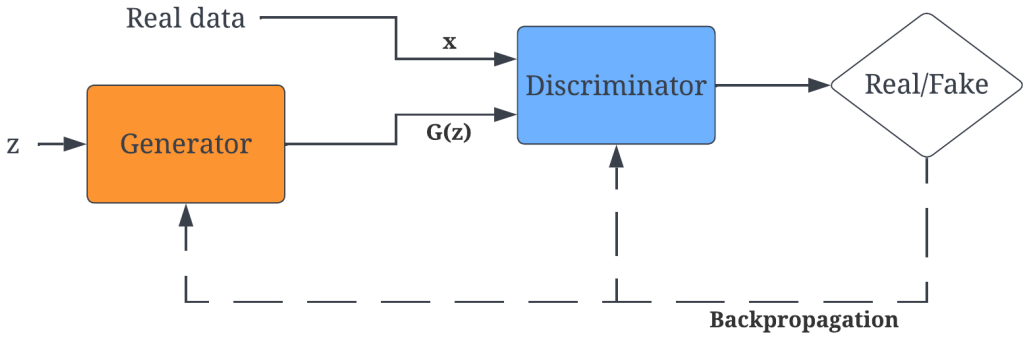


Figure 7. Generative adversarial network (GAN). [9]

To train G and D an adversarial process is used, where both networks compete with each other. D is trained to maximise the probability that the correct label is assigned to an image. Simultaneously G is trained to minimise the probability that the discriminator labels the fake images correctly. In other words, D and G play the following two-player minimax game with value function $V(D, G)$ [9]:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{real}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{3.1}$$

In theory, during training, the GAN model converges when both D and G reach a Nash equilibrium, i.e., a saddle point, which is the optimal point for Equation 3.1. The Nash equilibrium is achieved when players in a non-cooperative game lack any incentive to change their strategy. With GANs, this means that $p_g = p_{real}$ and D cannot tell the difference between real and generated samples. Finding this saddle point is a difficult task

and one of the main reasons why training GANs is difficult. For an example of how a standard and adversarial network are different in their training process, see Figure 8.

(a) Standard neural network

(b) Adversarial network

Figure 8. Standard and adversarial neural network training: (a) Standard neural network training always converges towards the minimum of the loss function. (b) Adversarial networks switch between minimisation and maximisation steps to converge to a saddle point of the loss function. [22]

From Figure 8b, it can be seen why the GAN training is highly sensitive to hyperparameters, such as the learning rate to find the saddle point. If the hyperparameters are not carefully chosen, a problem arises where one of the networks learns a lot faster and the other cannot keep up, making the model collapse and never recover.

GAN models have multiple different failure modes. In the case of *non-convergence*, the generator starts to output images that the discriminator can easily identify as fake. Usually, it is hard to recover from this state, which will lead to unstable training later on. The best way to identify *non-convergence* is to look at the loss of the model and the output scores of the discriminator.

The most common type of failure mode is *mode collapse*, where the generator outputs one or a small subset of images for any input noise value z. A *mode* can be thought of as an output type, two images with the same mode are visually really similar but may not be exact copies. There are two types of *mode collapse*: partial and full. *Full mode collapse* means the model only outputs a single mode or a very small subset of modes. *Partial mode collapse* is more subtle; the generated distribution seems quite diverse at first glance, but a closer look will reveal that the images contain features similar to each other. *Partial mode collapse* is usually identified by visual inspection of the generated images, as there is no reliable way to measure it. *Full mode collapse* usually is identifiable by quality metrics. An example of *mode collapse* can be seen in Figure 9

The third common failure mode is *vanishing gradients* [23]. This is closely related to *non-convergence*. When the discriminator becomes too good at identifying real images from fakes, the discriminator starts to provide the generator with less information on how to make progress. This leads to the generator training failing. Keeping an eye on the loss

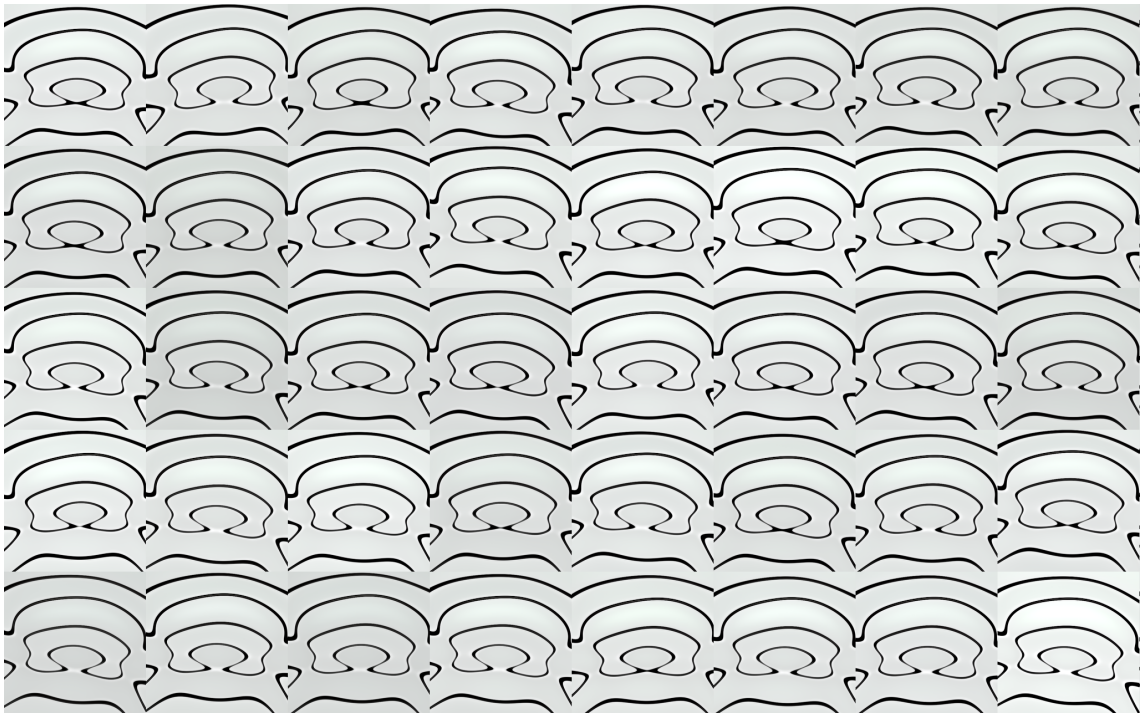


Figure 9. Outputs from mode collapsed GAN.

of the discriminator throughout the training process will help identify this failure mode.

Another important factor in training a quality GAN model is the training data provided to the model. GAN models usually need huge and diverse data sets to create quality images. Furthermore, data sets must have high image quality. GANs capture the data distribution of the training set. If the training set is corrupted in some way, these corruptions are also present in the generated distribution.

In summary, training GANs is a volatile process. Since 2014, a lot of research has been done on stabilising GANs and they have improved drastically. In this thesis, 4 different GAN architectures are used: StyleGAN2-ADA, StyleGAN2-ADA + LeCam, StyleGAN3, Projected GAN. These models were chosen based on the following criteria:

- Publicly available code,
- Generate images with resolution 256×256,
- Show good performance on small data sets,
- Publicly available pre-trained models for transfer learning

The following sections will give an overview of the chosen GAN architectures.

3.1 StyleGAN2-ADA

StyleGAN2 [24] is the updated version of StyleGAN [25], a progressive growing GAN, which significantly improves the quality of the generated images. This was achieved by changing the generator architecture and adding regularisation terms that help stabilise the training process. One significant fault that still remained was the need for large and good quality data sets; the models were trained on the FFHQ data set (approximately 70k images) and the LSUN data sets (>100k images). To reduce the number of imaged needed to train the models, stochastic discriminator augmentation was introduced [26].

Their solution uses a augmentation probability to apply a wide range of augmentation to the generated and real images that the discriminator sees. For this to work, the images augmentations cannot be leaky. Meaning that the generator should not be able to learn to generate images with augmentations. Non-leaking augmentation operations are defined by their ability to be invertible in the probability distribution. This allows the training to process to revert these augmentations and find the correct distribution. If only traditional data augmentation were used to increase our training set size, the GAN model would learn the distribution for augmented data and not the original distribution. The flow chart of StyleGAN2-ADA can be seen in Figure 10.

The strength of the augmentations is controlled by a heuristic that measures overfitting. If the model starts to overfit the augmentation strength, i.e. the probability that augmentations are applied, is increased and vice versa. One side effect of this adaptive technique is that if the augmentation strength increases too much, the generator no longer knows what a non-augmented image should look like and might still start to leak the augmentations. The effect of the augmentation probability on the images can be seen in 10.

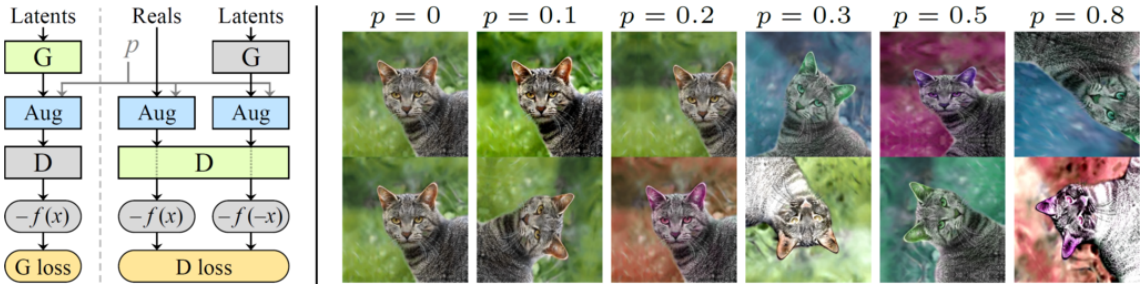


Figure 10. StyleGAN2-ADA. (left) StyleGAN2-ADA flow diagram, (right) effect of augmentation probability [26]

With this new method, the required data set size is reduced significantly. With only 1000 images, StyleGAN2-ADA managed to achieve only slightly worse FID (Fréchet Inception Distance) score than StyleGAN2 with 20k images. Furthermore, they showed that transfer learning from a GAN trained on a large data set significantly improved the quality of the

model when used with small data sets.

This thesis uses the PyTorch implementation of StyleGAN2-ADA publicly available on GitHub [1].

3.2 LeCam regularisation

This method regularises a f-divergence called the LeCam (LC)-divergence [27]. To achieve this, exponential moving averages are calculated for discriminator's predictions for both the real and generated images. The moving averages α_F and α_R, called *anchors* by the authors, are subtracted from the discriminator predictions. Followed by normalisation, squaring, and summing. Their proposed regularisation term:

$$R_{LC} = \mathbb{E}_{x \sim p_{real}}[||D(x) - \alpha_F||^2] + \mathbb{E}_{z \sim p_z}[||D(G(z)) - \alpha_R||^2] \tag{3.2}$$

Finally, R_{LC} is added to the discriminator training objective V_D and the strength of the regularisation term is controlled by a weight λ. The final objective of regularised discriminator training L_D:

$$\min_D L_D, L_D = -V_D + \lambda * R_{LC}(D) \tag{3.3}$$

The intuition behind this regularisation term is that the exponential moving averages are stable while the discriminator's predictions might not be and, therefore, the method penalises the difference between real and generated image predictions, holding the predictions in a particular range, helping the discriminator's predictions to converge to a stationary point [27].

Using their regularisation method, they showed that under limited training data conditions, i.e., 1000 images, adding R_{LC} to StyleGAN2-ADA improved the FID score from 23.27 to 21.70 on the FFHQ data set. LeCam regularizer is agnostic to the GAN architecture and can be added to any architecture. In this thesis, LeCam regularizer is added to the StyleGAN2-ADA architecture to see if it improves performance and is named StyleGAN2-ADA + LeCam.

[1] `https://github.com/NVlabs/stylegan2-ada-pytorch`

The PyTorch implementation for LeCam regularisation is available on GitHub [2].

3.3 StyleGAN3

StyleGAN3 [28] is an updated model of StyleGAN2-ADA, which tries to solve the problem of "texture sticking", whereby certain textures like hair appear to not move naturally with the object and seem to be stuck to the screen when in motion, and make the GAN equivariant to translation and rotation. They found that the aliasing problem was related to the poor handling of signal processing in the generator that leaked the positional information synthesis process. To fix this problem, they eliminated all positional references for details, so that a detail could be generated equally well at any pixel coordinate. Furthermore, they removed path regularisation, which encouraged sticking. The changes in the model make StyleGAN3 a bit more expensive to train.

The quantitative results of StyleGAN3 are similar to the previous version of StyleGAN2-ADA, the FID score decreased to 15.11 from 15.22, but the qualitative analysis shows that the aliasing problem is no longer present.

StyleGAN3 implementation is publicly available on GitHub [3].

3.4 Projected GAN

Projected GAN [29], promises to make the GAN training even more accessible. Their novel training method uses a pre-trained network to obtain embedding for images that the discriminator processes. The generator and discriminator do not optimise the image distribution in the image space, but in the pre-trained feature space. Projected GAN introduced a set of feature projectors P_l, which turn real and generated images into features mapped to the discriminator input. The features are obtained at different resolutions and, for every resolution, they use a separate discriminator D_l to classify between real and fake features. Projected GAN training can be formulated as follows:

$$\min_{G} \max_{D_l} \sum_{l \in \mathbb{L}} (\mathbb{E}_{x \sim p_{real}(x)}[\log D_l(P_l((x))] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D_l(P_l(G(z))))]) \quad (3.4)$$

This new training method results in the model converging much faster while still providing

[2] https://github.com/google/lecam-gan
[3] https://github.com/NVlabs/stylegan3

state-of-the-art FID scores. Projected GAN matched the previously lowest FID score 40 times faster. Using small data sets (1000 images) Projected GAN improved the FID score over StyleGAN2-ADA from 43.07 to 27.96.

The code for Projected GAN is publicly available on GitHub [4].

[4] https://github.com/autonomousvision/projected_gan

4. Convolutional Neural Networks

CNNs (Convolutional Neural Networks) are a popular tool for image classification, as they provide automatic feature extraction [30]. A CNN consists of two parts: a feature extractor and a classifier (Figure 11). Feature extractor consists of subsequent convolutional and pooling layers, which take images as input and create feature maps as output. This is then used as input to the classifier, outputing the class to which the image belongs.

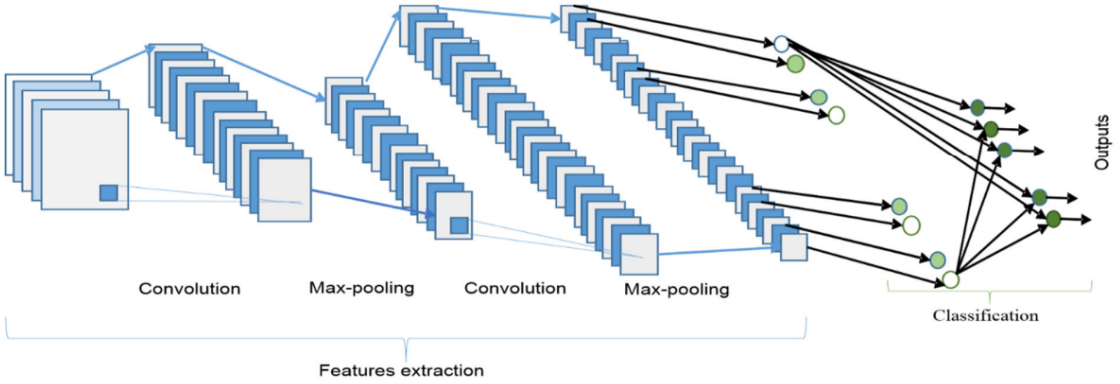


Figure 11. Basic architecture of CNN [31].

For the image classification task, 6 different convolutional neural networks: AlexNet, ResNet, VGG, Inception, Xception, DenseNet are used. The selection contains both recent and older model architectures. This was done to gain a better understanding of how architectures with different depths and levels of complexity affect the performance of Parkinson's classification. In the following sections, the model architectures will be described.

4.1 AlexNet

AlexNet [32] is one of the most important CNN architectures of the past decade. It started the use of ReLU activation function, pooling layers with overlapping and could be trained with multiple GPUs. Furthermore, it was one of the first models to use dropout. This allowed AlexNet to be the first CNN architecture to win the ILSVRC (Imagenet Large Scale Visual Recognition Challenge) with an error rate of 15.3%, over 10% better than the second-best model.

AlexNet consists of five convolutional layers and three fully connected ones. In total, it consists of 61 million learnable parameters. An illustration of AlexNet, from the original article, can be seen in Figure 12.

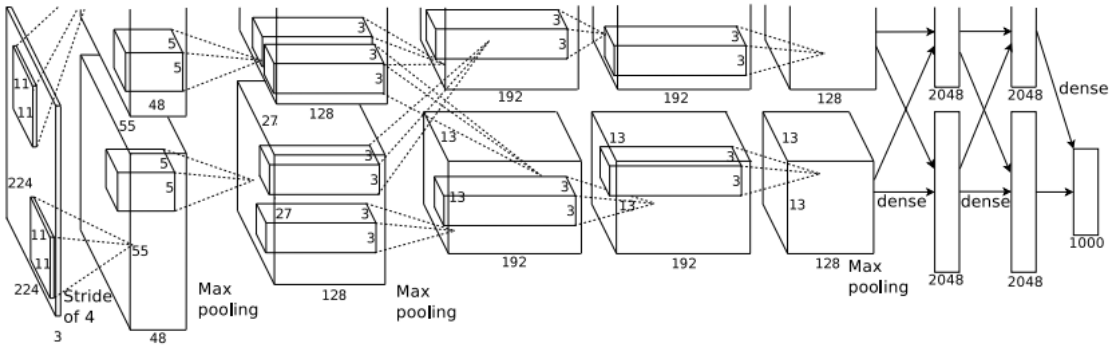


Figure 12. AlexNet architecture [32].

4.2 VGG

To further improve classification accuracy, [33] increased the depth of the model by adding convolutional layers and used 3×3 kernels throughout the network. In 2014 their model got a top-5 error rate of 6.8% at ILSVRC (10-crop), only beaten by GoogLeNet. A visualisation of the model can be seen in Figure 13.

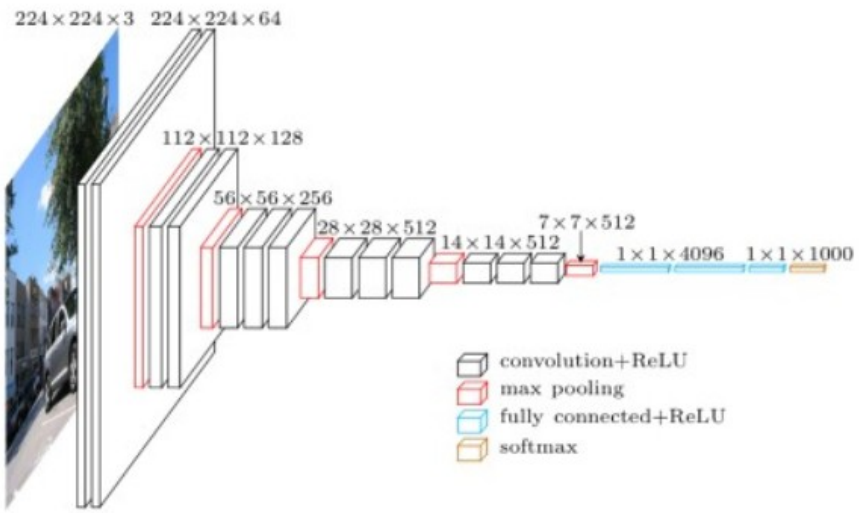


Figure 13. VGG architecture [33].

The smallest configuration of the model consists of 133 million learnable parameters, and the largest 144 million, more than double the size by number of parameters. Configuration A described in [33] is used in this thesis.

4.3 Inception v3

The Inception architecture [34] performs convolution on the input with different filter sizes, and additionally, the input is max pooled. This Inception module helps the network to collect global and local information in the image. The naive version of the Inception module can be seen in Figure 14. Furthermore, they use average pooling before the final fully connected layer. To deal with vanishing gradients, the model uses two auxiliary classifiers, which increase the gradient signal that is backpropagated.

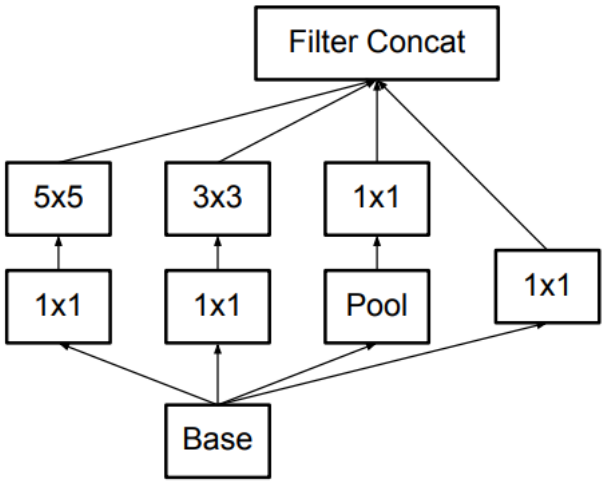


Figure 14. Inception module [35].

The first model to use the Inceptions architecture, GoogLeNet (Inception-v1), won first place on the ILSVRC 2014 classification task with a top-5 error rate of 6.7% (10-crop), slightly outperforming VGG. Another benefit of this model is that it uses 6.8 million learnable parameters, 20 times less than VGG.

The second and third versions of the Inception architecture were published a year later [35]. The second version optimises the convolutional operations by creating multiple versions of the inception model. One in which the 5×5 convolution is replaced with two 3×3 convolutions, another in which the n convolution is replaced with a combination of 1 and $n \times 1$ convolutions, and the final in which the filter bank output is expanded. The third version further improves the model by adding label smoothing, factorising 7×7 convolutions, and batch normalising the fully connected layers of the auxiliary classifier. With these improvements, Inception-v3 achieved a top-5 error rate of 4.2% (multi-crop) on the ILSVRC classification benchmark. Inception-v3 has 24 million model parameters.

4.4 ResNet

The authors of ResNet noted that increasing the depth of the network by simply adding layers does not improve performance [36]. As a network gets deeper, vanishing gradients become a larger issue. This means that as the gradient is backpropagated through the layers, it might become insignificant. Resulting in a network with degraded performance. The authors of [36] introduced a residual block. This works by skipping one of more layers and adding the input of the skipped layer to the outputs. Figure 15 shows a visualisation of the block.

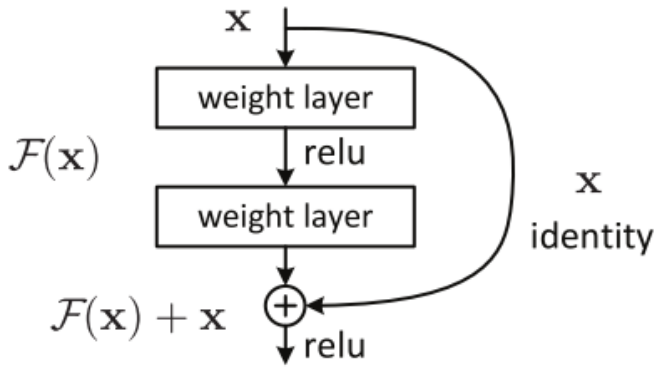


Figure 15. Residual block [36].

Their model architecture is inspired by VGG but they remove two of the three fully-connected layers from the end and replace it with average pooling. Furthermore, they use convolutional layers with smaller input and output. Both of these changes significantly simplify the models' complexity, bringing it down to 12 million learnable parameters with the 18-layer configuration. In comparison, the ResNet configuration with 152 layers is still less complex than VGG. ResNet won first place on the ILSVRC 2015 classification task with a top-5 error rate of 3.57%.

4.5 Xception

Xception [37] introduces an "extreme" version of the Inception module. This version of the Inception module uses a 1x1 convolution to cross-channel correlations, after which each output channel's spatial correlation is individually mapped. This "extreme" Inception module is very similar to depthwise separable convolution. An illustration of the module can be seen in Figure 16. Additionally, each of the "extreme" Inception modules uses residual connections, similar to ResNet.

This change results in a model that has a similar amount of model parameters to Inception-

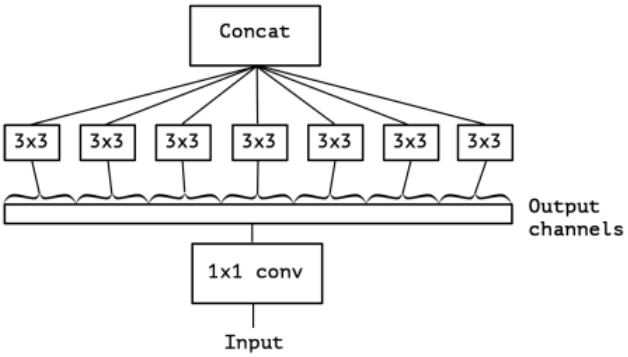


Figure 16. Extreme Inception module [37].

v3, while improving the top-5 classification error rate by 0.4% (single crop, single model) over Inception-v3.

4.6 DenseNet

DenseNet architecture [38] connects each layer of the network with every other layer. This helps further alleviate the problem of vanishing gradients, encourage feature reuse, strengthen feature propagation, and substantially reduce the number of model parameters. To achieve this, the architecture consists of multiple dense blocks where every layer is connected to every other layer, and transitional layers that change the feature map sizes. Simply connecting layers in a regular CNN would not work because CNNs use downsampling layers that change the feature map size between convolutional layers. An illustration of DenseNet can be seen in Figure 17.

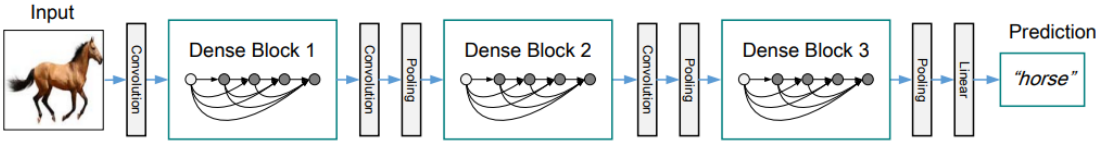


Figure 17. DenseNet architecture [38].

The authors report a 5.29% top-5 error rate (10-fold) on the ILSVRC 2012 classification validation set.

5. Experimental setting

This chapter describes the overall workflow, experimental settings, and hyperparameters used for image processing, GANs, and CNNs. This consists of several different steps, a visualisation of the workflow can be seen in Figure 18. The following sections will describe each of these processes in more detail.

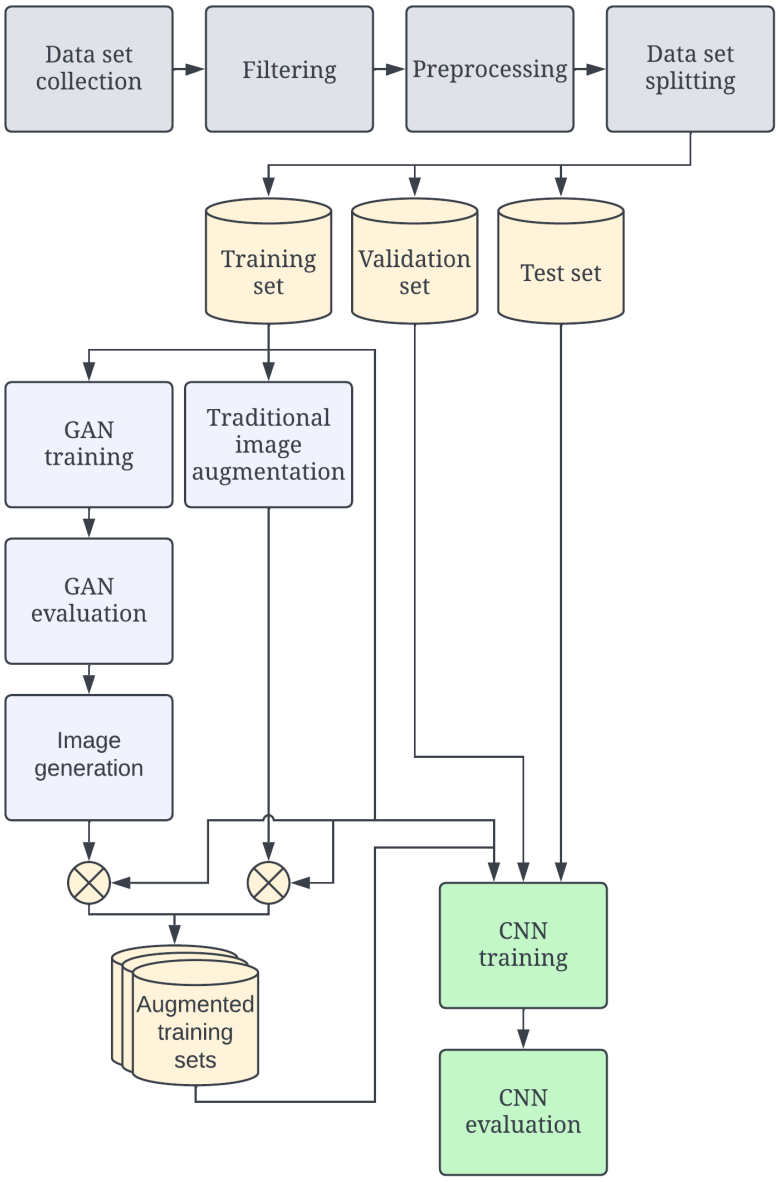


Figure 18. Experimental workflow.

5.1 Data filtering

To clean up the data set, exact duplicate and low-quality images were searched for. Duplicates were found using a perceptual hashing algorithm implemented in the imagededup library [1]. Perceptual hashing takes an image and creates a fingerprint from that. After all the images are hashed, Hamming distance is used to compare the similarity of images. Exact duplicates were found in two data sets HandPD and NewHandPD, other data sets did not contain any duplicates. As the NewHandPD data set is an extension of the HandPD data set, these data sets contain inter data set duplicates. Furthermore, the NewHandPD data set itself contains intra data set duplicates. In total, 144 duplicate images were found. An example of duplicate images can be found in Figure 19, the image *0005-1.png* is from HandPD and others from NewHandPD.

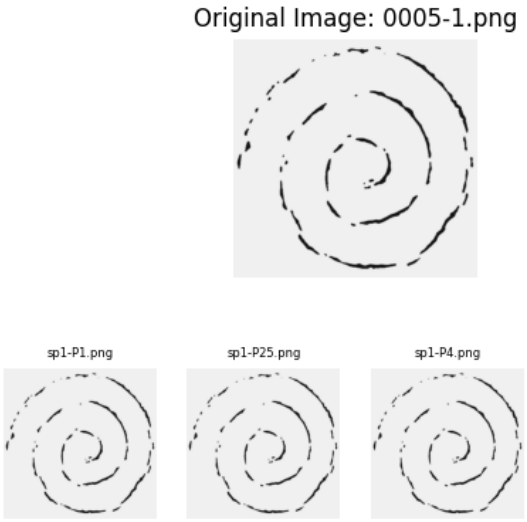


Figure 19. HandPD and NewHandPD duplicates example.

Additionally, the data sets were combed over manually to find images of poorer quality and remove them. NewHandPD images in the range xxx-H16 to xxx-H37 were particularly poor quality and appear to exhibit image compression artefacts (Figure 20). NewHandPD contained 84 of these types of images.

After the removal of duplicates and low quality images, 702 images were left (210 healthy controls and 492 Parkinson's patients).

[1] `https://idealo.github.io/imagededup/`

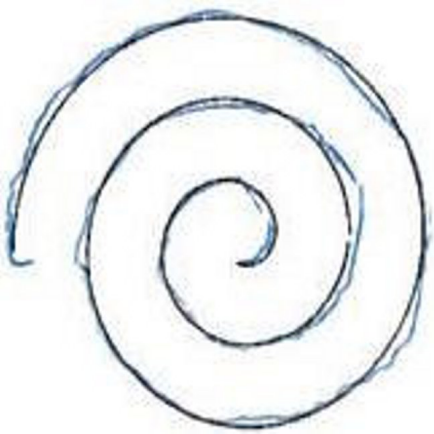


Figure 20. Poor quality NewHandPD image (sp2-H24).

5.2 Preprocessing

As the images used by the classification and GAN models come from several different data sets, the style and image quality varies significantly between the data sets. Before training, each image was passed through the preprocessing step, where the background noise and template, if present, were removed and the images were converted to greyscale. This step was carried out so that the images would be as similar in style as possible. Each data set was handled slightly differently, taking into account the characteristics of each data set. Examples of the final pre-processed images can be seen in Figure 22.

For HandPD and NewHandPD, the image preprocessing steps of [39] were followed, which consisted of blurring the image and thresholding. An illustration of the process from [39] can be seen in Figure 21. Additionally, images were turned into greyscale, as previously mentioned.

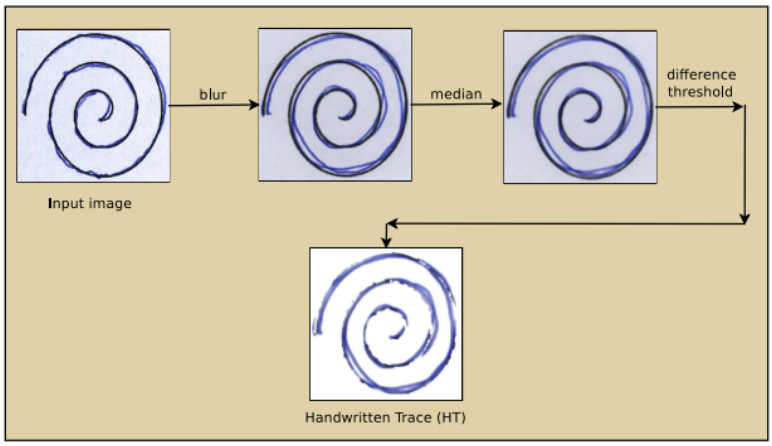


Figure 21. HandPD and NewHandPD preprocess steps [39]

A similar preprocessing pipeline was used with Parkinson's drawing data set. These images

are greyscale from the start, so instead of thresholding based on the difference in each colour channel, thresholding was based on the intensity of the black value.

Digitised data sets, like DraWritePD and Parkinson's Drawings, do not have any background noise or spiral templates in their images. This made pre-processing the most simple; the images needed to be only grayscaled and blurred.

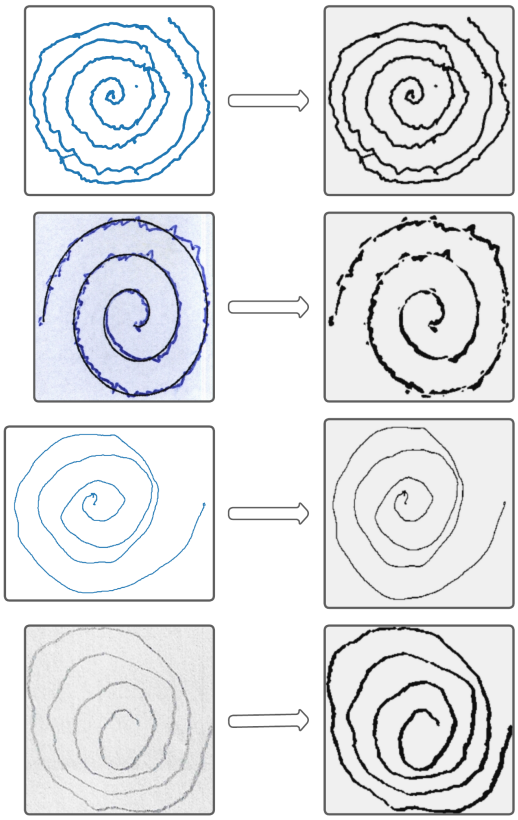


Figure 22. Preprocessing results for DraWritePD, HandPD, ParkinsonHW, Parkinson's Drawings (from top to bottom). (left) Original images, (right) preprocessed images.

In addition to all the previously mentioned steps, the images are also resized to a uniform size of 256×256 pixels and a small Gaussian blur filter is applied. Aliased images, where the edges of the stepped steps are prominent, have been shown to make the calculation of the KID more inconsistent and the training of GANs more difficult [28, 40]. Applying an anti-aliasing filter removes the jagged edges. Additionally, the maximum and minimum colour value differences were decreased in the image by not setting the background to pure white and the spiral to pure black. Instead, RGB values of $(240, 240, 240)$, for the background and $(15, 15, 15)$, for the spirals, were used, so that the GAN models should not have to generate extreme colour values.

All preprocessed images were saved in PNG format.

5.3 Data set splitting

Images were divided into a 80%/10%/10% train/validation/test split, where the validation and test set were balanced, as seen in Table 2. The validation and test splits were balanced to eliminate bias in the testing phase. Validation and test sets contained images from each data set proportionally to the size of each data set.

Table 2. Data set split. HC - Healthy control, PD - Parkinson's disease

Label	Train images	Validation images	Test images
HC	140	35	35
PD	422	35	35

5.4 Traditional image augmentation

To augment the images, the Albumentations library [2] was used. The augmentation pipeline is described in Table 3. Each of the augmentation functions is applied with a probability of 50%. Visualisation of the entire pipeline can be seen in Figure 23, where all augmentations are applied to a spiral.

Table 3. Traditional image augmentation pipeline.

Augmentation pipeline
HorizontalFlip()
VerticalFlip()
RandomRotate90()
GridDistortion(border_mode=cv2.BORDER_CONSTANT, value=240)
RandomScale()
RandomBrightnessContrast()

5.5 GAN training

With each GAN architecture, two unconditional GAN models were trained, one that generates spiral images of healthy controls and the other that generates spiral images of Parkinson's patients.

Each of the GAN models was trained using transfer learning from a model trained on the FFHQ (Flickr-Faces-HQ) data set [25]. Only the training split, which was amplified with

[2] https://albumentations.ai/

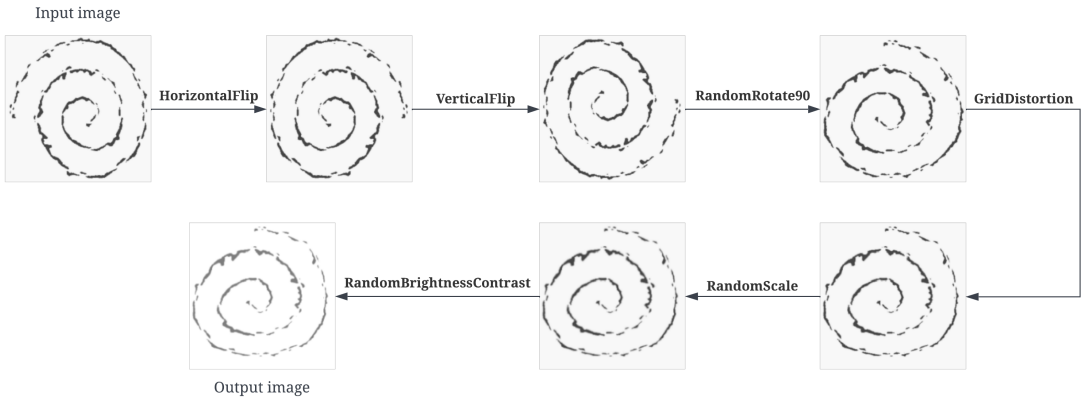


Figure 23. Visualisation of image enhancement pipeline.

horizontal, vertical flips, and horizontal + vertical flips, was used during GAN training, quadrupling the size of the training set. The images generated are of size 256×256 pixels. Each model was trained on a single NVIDIA A100 GPU for three days. The configuration for each of the GAN architectures can be seen in Table 4. The base configuration was chosen based on the recommendations of each architectures authors for training with limited data. For the batch size, the largest power of two that the GPU would manage was selected. R1 regularisation value was selected based on the formula found in Section D in [24]. LeCam regularizer used the recommended $lambda$ value, when used with StyleGAN [27].

Table 4. GAN configurations.

GAN	**Base config**	**Batch size**	**R1 regularization**	**LeCam λ**
StyleGAN2-ADA	auto	64	0.2048	-
StyleGAN2-ADA + LeCam	auto	64	0.2048	$3 * 10^{-7}$
StyleGAN3	StyleGAN3-T	64	0.2048	-
Projected GAN	FastGAN-lite	64	-	-

5.6 GAN evaluation

GAN evaluation is usually done with both quantitative and qualitative analysis.

For quantitative evaluation of the GAN model, KID (Kernel Inception Distance) was used, which measures the dissimilarity between probability distributions and is unbiased when used with small data sets, unlike FID, which is more commonly used as a GAN quality metric [41]. KID was calculated every 50 steps and the model checkpoint with the lowest KID was selected as the best. Calculating multiple metrics could have also been an option, but these metrics are often expensive to calculate, taking around 10 minutes to calculate

per metric.

Furthermore, the loss of both models (generator and discriminator) and the discriminator prediction scores were analysed to better understand the training process. Providing a way to see if any of the failure modes were produced during training.

A qualitative evaluation was performed using several methods. The most simple is the K-Nearest Neighbour (KNN) analysis. This method entails passing a generated image and all training images through a pre-trained CNN feature extractor (in this case ResNet50) and collecting the embeddings for each image. Euclidean distance is then used to find the training images closest to the generated image. An example of the output can be seen in Figure 24. This method can be used to see if the generated images are unique and to check that the model has not memorised the training images.

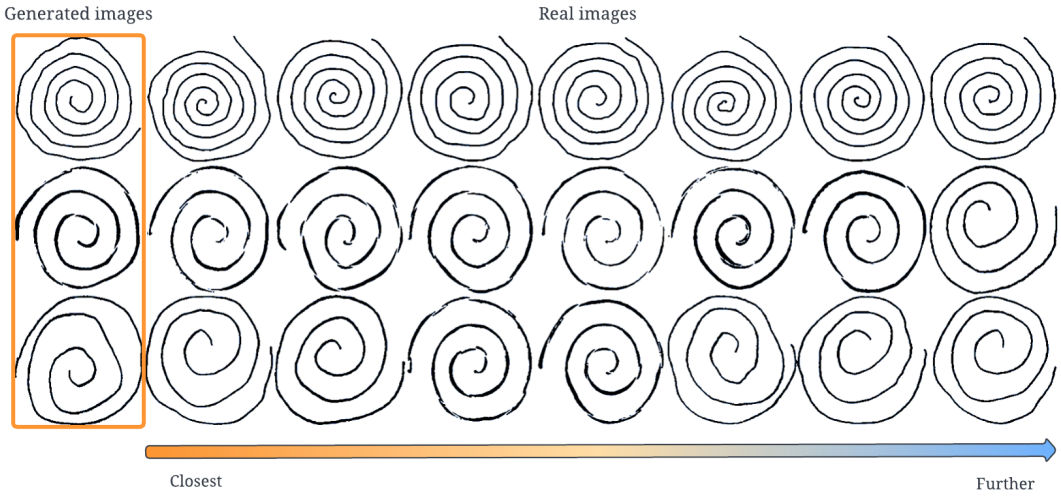


Figure 24. K-Nearest Neighbor example. The images in the first column are generated and the following images are real images ordered by distance to the generated image.

Overfitting can also be detected by inspecting the latent space interpolation, this can be more helpful when dealing with small data sets [29] as metrics like KID do not detect overfitting well on small data sets. In the perfect scenario, GAN models should generate smooth interpolations between samples. This would indicate that the model does not memorise the training data and generalises the training distribution.

Another method used is t-SNE (t-distributed stochastic neighbour embedding), which visualises high-dimensional data in a low-dimensional map, similarly to PCA. The main difference being that t-SNE is non-deterministic. This comes from the fact that the objective function of t-SNE is minimised using gradient decent and randomly initialised. This allows t-SNE to provide a better visualisation of the data, in most cases. The results of the method

are often dependent on the selected hyperparameters. The most important being perplexity, setting the perplexity to 40 gave quite good results with this data set. The algorithm was iterated for 300 iterations. The example output of t-SNE can be seen in Figure 25. Comparing the real and generated data distributions gives a better understanding of how the GAN model performs and if the generated distribution matches the real one.

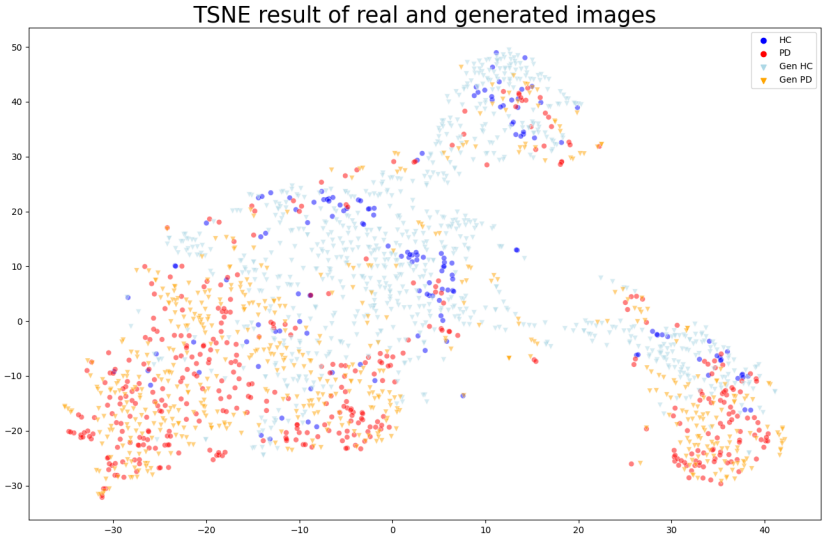


Figure 25. t-distributed stochastic neighbor embedding example.

5.7 Image generation

To generate images with GANs, randomly generated latent variables z are given as input to the generator. This results in the generated set of images covering the largest area of the generators distribution, as latent space variables that are close together are also similar in image space.

Often when generating images with GANs a truncation trick is used. This allows for the control of image fidelity and diversity after the model is trained. It works by truncating the normal from which the vector z is sampled and resampling any values that fall outside the truncated normal. The truncation is controlled by a hyperparameter ψ. Reducing ψ results in z elements being truncated toward zero and the images becoming less diverse but higher in quality. Increasing ψ, on the other hand, creates more diverse images but reduces the fidelity of the images [42]. Truncation ψ can be any positive real number, but is usually set in a range from 0–2.0 with default being 1.0. The effect of different ψ values can be seen in Figure 26. When generating images with GANs in this work, ψ is set to 1.0.

Figure 26. Truncation trick affect. From left to right ψ is set to 2, 1, 0.5, 0.04. [42]

5.8 Augmented training set generation

To measure the effectiveness of GAN-generated images in the classification task, multiple training sets were created. For each of the GAN architectures, the original training set was extended with images generated from the GAN models. Furthermore, one training set was extended with traditionally augmented images, to compare the training sets with GAN generated images. Because the original training set is highly imbalanced, the extended training sets were created to contain nearly the same amount of images of healthy controls and Parkinson's patients. The sizes of the training sets can be seen in Table 5.

Table 5. Training sets.

Training set	**Size**	**HC**	**PD**
Original	562	140	422
Traditionally augmented	1992 (562 + 1430)	990 (140 + 850)	1002 (422 + 580)
GAN-augmented	1992 (562 + 1430)	990 (140 + 850)	1002 (422 + 580)

5.9 CNN training

Each of the model architectures (Figure 6), was trained using transfer learning with the base models trained on the ImageNet data set [43]. Turing transfer learning, the model was fine-tuned, meaning that all the layers of the model were updated.

Table 6. CNN architecture summary.

Name	**Config**	**Parameters (million)**	**Input size**
AlexNet	-	61	224×224
VGG	A [33]	133	224×224
Inception-v3	-	24	299×299
Xception	-	23	299×299

Continues...

Table 6 – *Continues...*

Name	**Config**	**Parameters (million)**	**Input size**
ResNet	50-layer [36]	26	224×224
DenseNet	DenseNet-121 [38]	8	224×224

The training configuration used for the image classification models can be seen in Table 7. The settings are based on related work [7, 44].

Table 7. CNN training configuration.

Setting	**Value**
Training epochs	30
Batch size	64
Initial learning rate	0.0001
Learning rate decay	Exponential
Learning rate decay gamma	0.9
Optimiser	Adam
Optimiser weight decay	0.0005
Loss	Cross Entropy

5.10 CNN evaluation

The performance evaluation of the trained CNNs was carried out using two quality metrics: sensitivity (equation 5.1) and specificity (equation 5.2). Sensitivity, or true positive rate, can be thought of as the Parkinson's patient prediction accuracy. On the other hand, specificity or the true negative rate is the accuracy of healthy control prediction. These were calculated on the basis of the test set to remove any bias from the results. These metrics were selected because they have clear meaning in the medical domain, they are intrinsic to the test (CNN model) itself and are not dependant on the prevalence of a disease in the population.

$$Sensitivity = \frac{TP}{TP + FN} \tag{5.1}$$

$$Specificity = \frac{TN}{TN + FP} \tag{5.2}$$

Furthermore, the accuracy of the training and validation set was plotted throughout the training. This gives more insight into how each model performs and if there might be any concern of overfitting the models to the training set.

6. Results

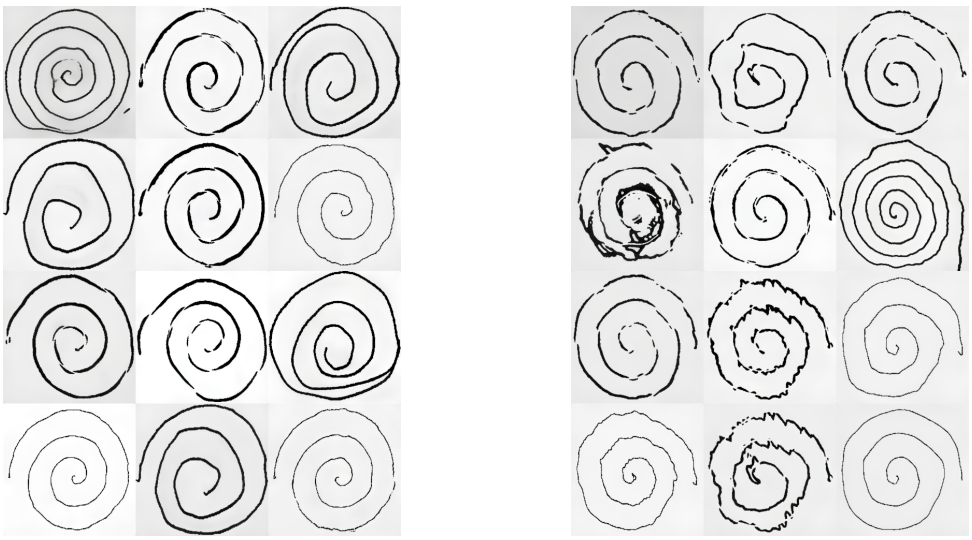
This chapter reports all the results obtained in this thesis. First, all the GAN image generation results are described and analysed. Afterwards, the impact of the generated data on the CNN classifiers is investigated. Finally, a discussion about the results is provided.

6.1 GAN results analysis

In this section, performance of each GAN architecture is analysed. From here onwards, the models trained with spirals of healthy controls will be suffixed with HC and the models trained with spirals of Parkinson's patients will be suffixed with PD.

6.1.1 StyleGAN2-ADA

During the three-day training period, the StyleGAN2-ADA models train for approximately 2900 steps and saw 11.5×10^6 images. In that time, StyleGAN2-ADA HC achieved a KID score of **0.01416** (step 1300) and StyleGAN2-ADA PD **0.01054** (step 1450). A handpicked set of generated images from the best checkpoint can be seen in Figure 27, and a larger sample grid can be viewed in Appendix 2 (Figures 61 and 62).



(a) (b)

Figure 27. Handpicked examples from StyleGAN2-ADA best checkpoint. (a) HC, (b) PD.

Visually, the images look quite good, and clearly the models managed to learn to generate spirals. Partial mode collapse is seen in both models, more so in the HC model (Figure 27a - last row first and third images and Figure 27b - last column third and fourth images). Additionally, it can be seen that the sample background is different shades of grey and that the backgrounds are not of uniform colour. There are also darker areas in the image that almost look like someone had erased the spiral with a eraser and drawn another one. This is probably due to leaking colour augmentations. Furthermore, it can be noted that the HC model produces spirals drawn clockwise starting from the left, whereas the spirals from the PD model are drawn counterclockwise and start from the right.

Looking at the results of the k-nearest neighbour in Figures 28 and 29. The generators manage to mimic the styles of the original data sets and generate samples that are not present in the training data set but can be quite close. Confirming that the generator manages to create novel images of Archimedean spirals.

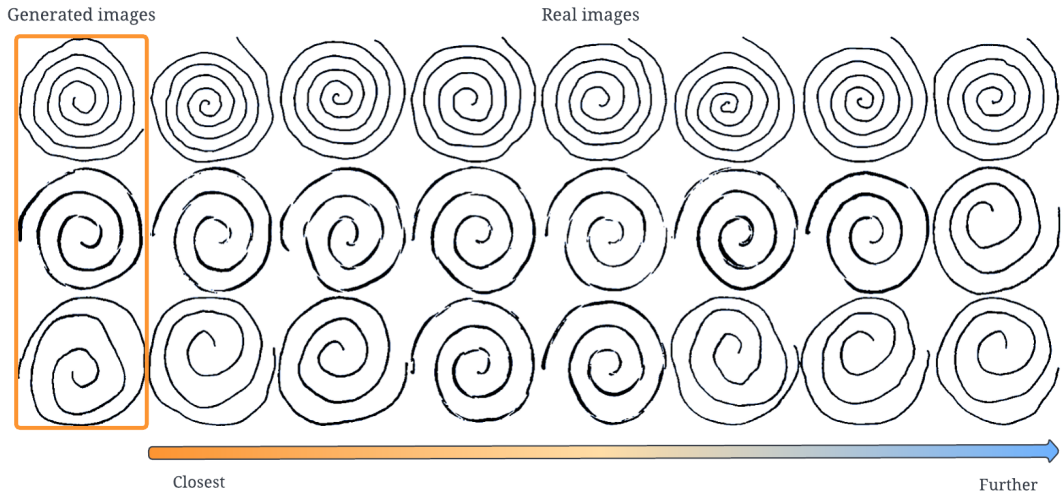


Figure 28. StyleGAN2-ADA HC k-nearest neighbours

The KID steadily decreases for around 1000 steps, after which both model KID become quite (Figure 30). The HC model continues to be stable until the end of training, while the KID scores for the PD model start to increase around step 2200. Indicating that the generator has started to produce samples that are worse than before and that training any further most probably will lead to a failure mode later on.

From the loss graphs in Figure 31, both generator losses increase in the training process, PD slightly slower than HC, at the beginning, but at the end the losses are similar. Showing that the generator for both models becomes worse as the model trains. This is confirmed by the loss of the discriminator, which is near zero, which means that the discriminator can distinguish between real and fake images.

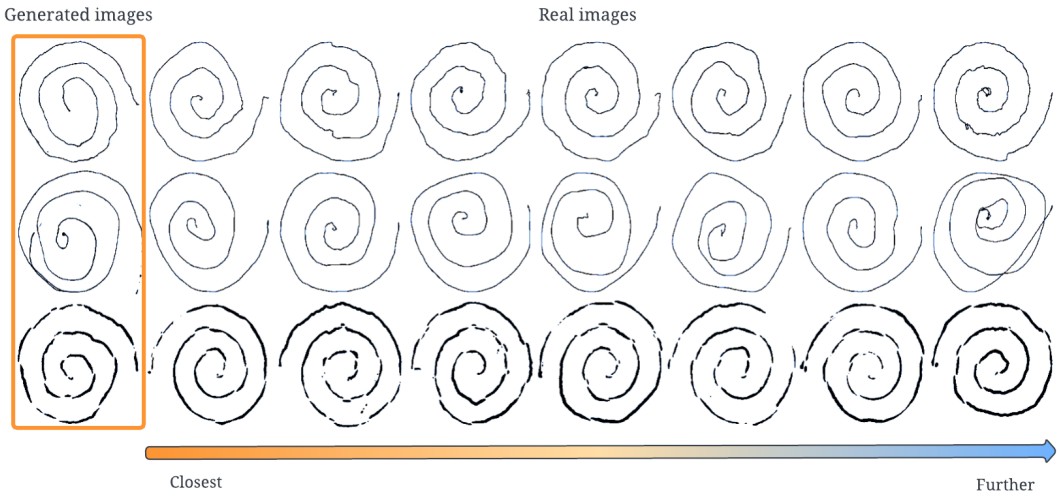


Figure 29. StyleGAN2-ADA PD k-nearest neighbours

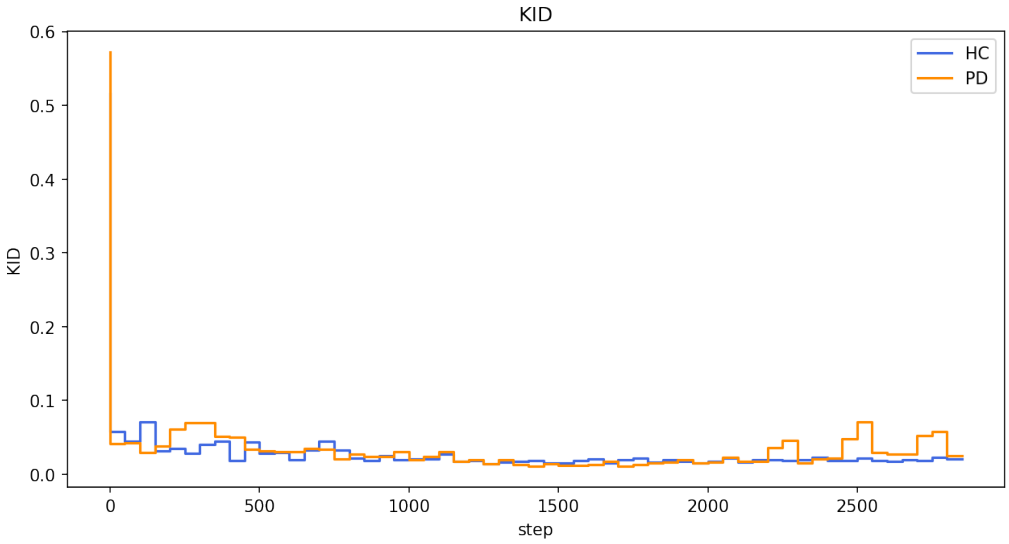


Figure 30. StyleGAN2-ADA KID

This can also be seen in the score plot in Figure 32. As training progresses, the discriminator becomes really good at separating real images from fakes. On the other hand, the generator cannot learn anything that would make discriminators work harder. Ideally, the discriminator scores for both real and fake images would converge toward zero, as this would mean that there is a 50% probability that the discriminator correctly identifies the image.

The plot of the real and generated images using t-SNE shows that the PD model manages to better capture the original data distribution of the PD images. The generated HC images seem to be farther from the original distribution and create their own clusters.

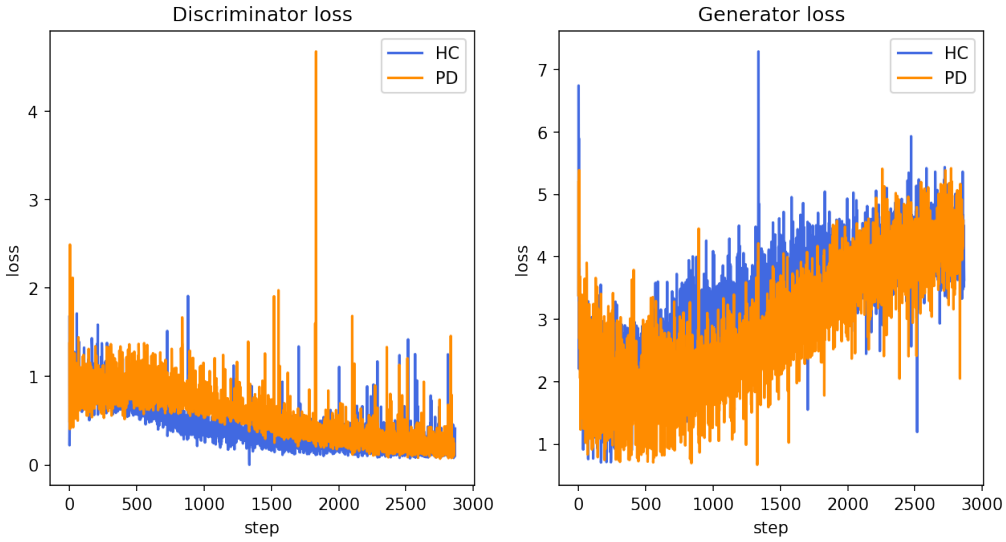


Figure 31. StyleGAN2-ADA (left) discriminator and (right) generator loss

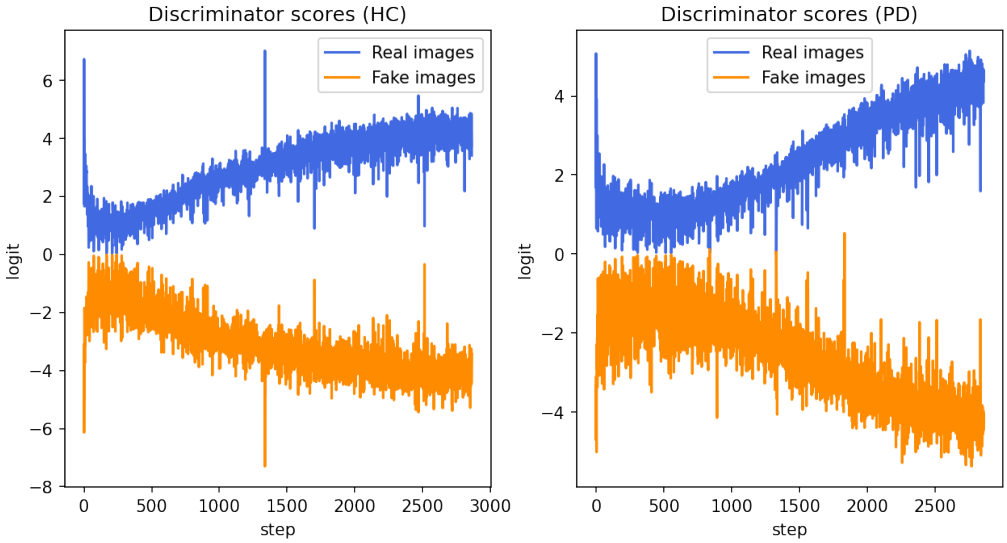


Figure 32. StyleGAN2-ADA HC (left) and PD (right) discriminator scores

The latent space interpolation videos for both StyleGAN2-ADA models can be accessed from Google Drive [1]. The videos show that the PD model interpolates more smoothly between the spirals than the HC model.

6.1.2 StyleGAN2-ADA + LeCam

In total, each StyleGAN2-ADA + LeCam trained for 2600 steps and saw 10.5×10^6 images. The best KID scores were **0.01826** (step 2400), for the HC model, and **0.02367** (step 1050), for the PD model. A small selection of handpicked samples from the best checkpoint can

[1] `https://drive.google.com/drive/folders/1DFT-9D50J37mLkHvKCBl_xvp_fLDnrog?usp=sharing`

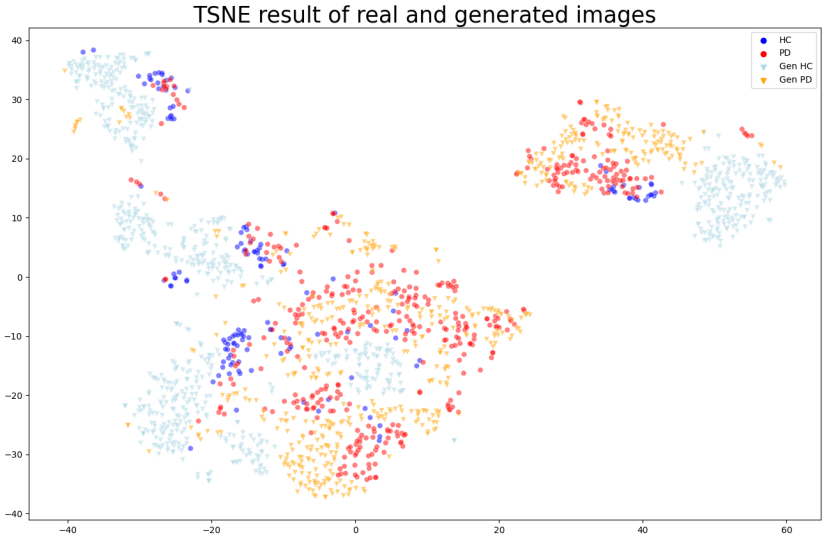
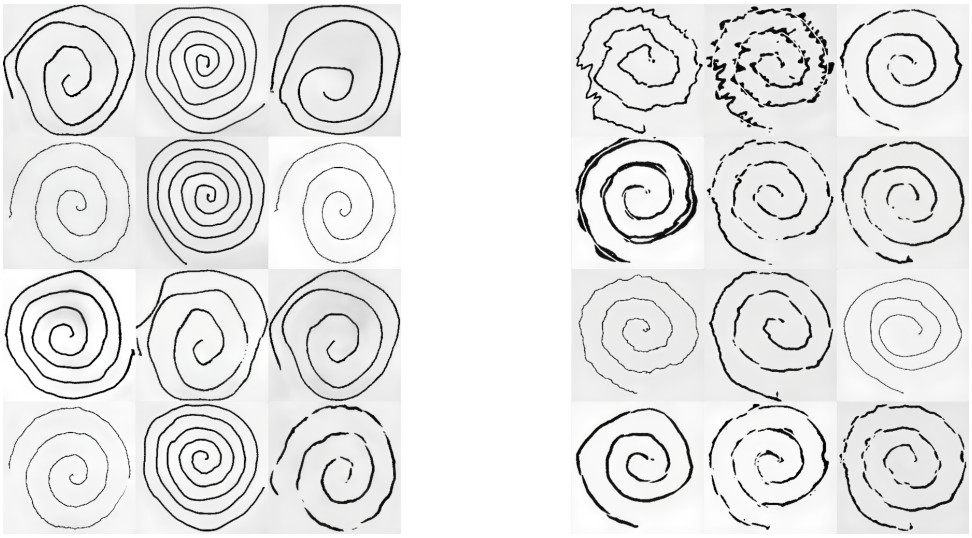


Figure 33. StyleGAN2-ADA t-SNE

be seen in Figure 27. The larger grid can be viewed in Appendix 2 (Figures 63 and 64).



(a) (b)

Figure 34. Handpicked examples from StyleGAN2-ADA + LeCam best checkpoint. (a) HC, (b) PD.

The image quality seems to be really similar to StyleGAN2-ADA. Again, the background is not the same between the samples, the darker smudges on the image seem to be more prevalent, and the model still exhibits partial mode collapse. Additionally, the PD model spirals start at the same position, while the HC model samples are more random with their starting position, which might be caused by the leakage of the augmentations.

K-nearest neighbour plots in Figures 35 and 36 give a similar conclusion as before. The

generated images can in some ways be quite similar to the real images, but other areas can be unique to the generated images. For example, the shape of the spiral can be very close to that of a real image, but the tremors in the spirals might be located in other parts of the spiral.

Figure 35. StyleGAN2-ADA + LeCam HC k-nearest neighbours

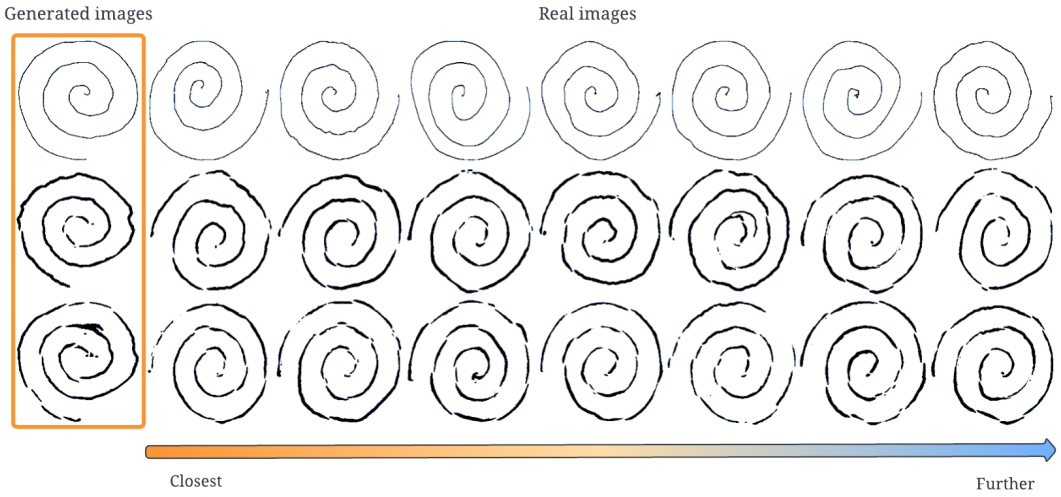


Figure 36. StyleGAN2-ADA + LeCam PD k-nearest neighbours

Figure 37 shows that the KID for the HC model decreases at a steady rate until the end of training. However, the PD model's KID is much more volatile near the end of training, exhibiting larger differences between neighbouring KID calculations.

Model losses (Figure 31) are very similar to the StyleGAN2-ADA losses (Figure 31). One small difference is that the StyleGAN2-ADA + LeCam discriminator has more spikes in the losses.

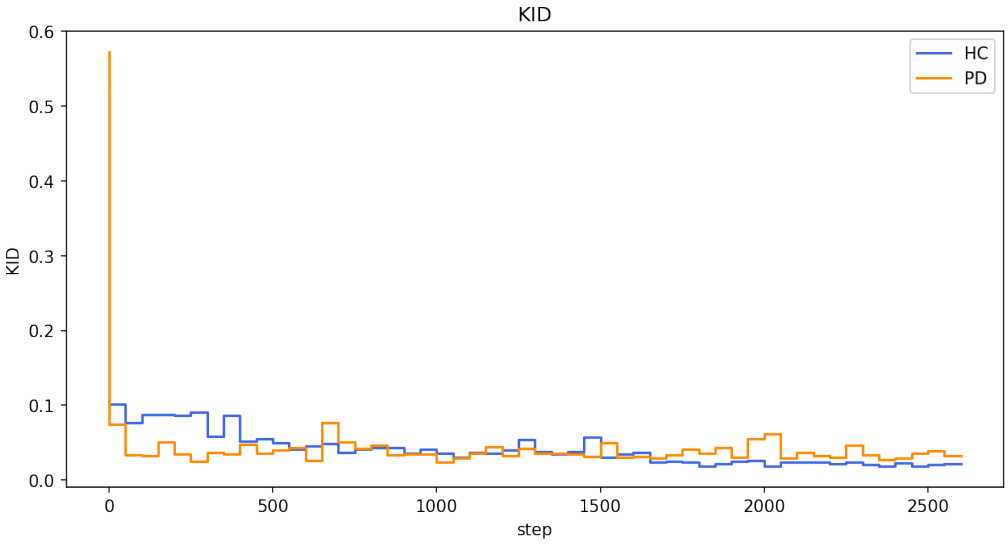


Figure 37. StyleGAN2-ADA + LeCam KID

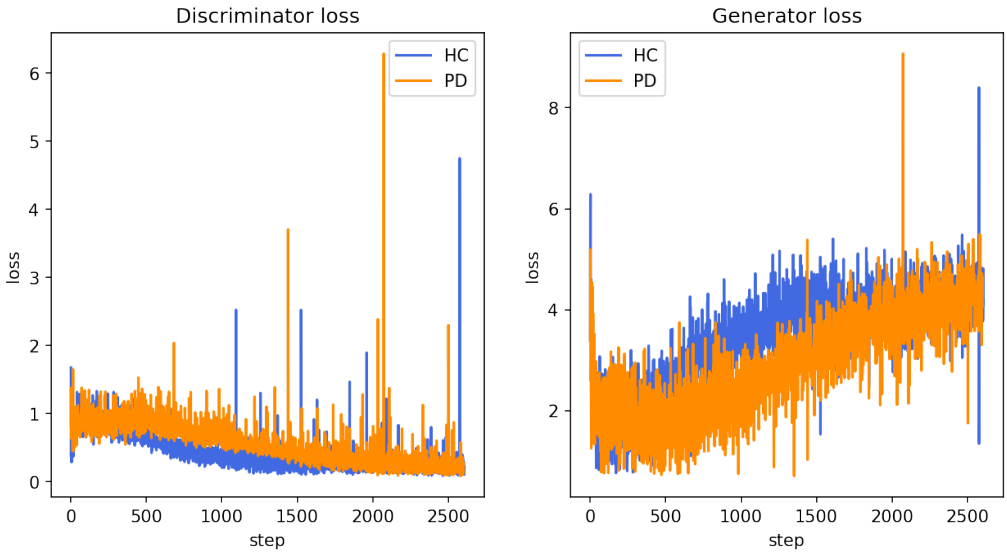


Figure 38. StyleGAN2-ADA + LeCam (left) discriminator and (right) generator loss

Furthermore, the discriminator scores are also very similar to StyleGAN2-ADA (Figure 39). All in all, it seems that adding the LeCam regularizer did not improve the performance of the StyleGAN2-ADA model.

The t-SNE plot shows that this architecture has even more trouble generating images in the real data distributions. There is hardly any overlap between the generated and real HC distributions. Furthermore, the generated PD distribution seems to also be more separate from the real distribution.

The latent space interpolation videos for both StyleGAN2-ADA + LeCam models can be

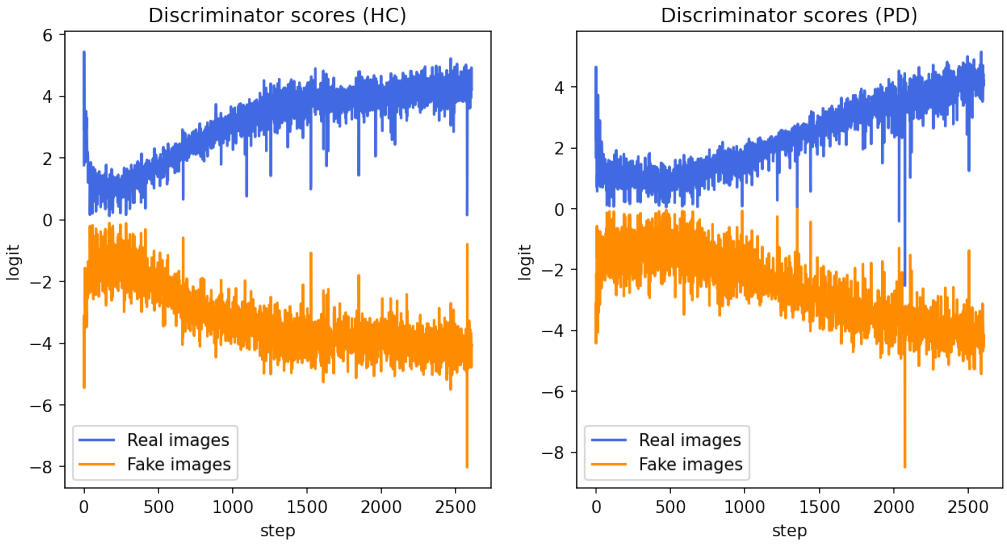


Figure 39. StyleGAN2-ADA + LeCam HC (left) and PD (right) discriminator scores

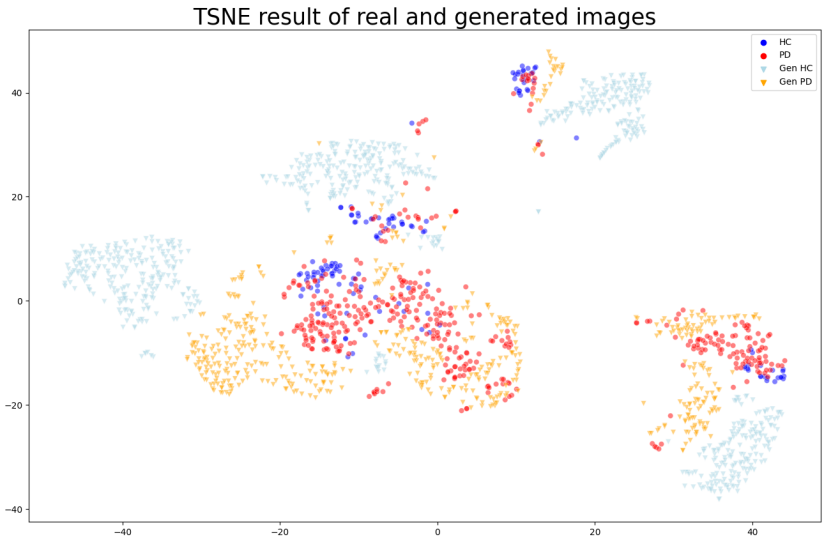


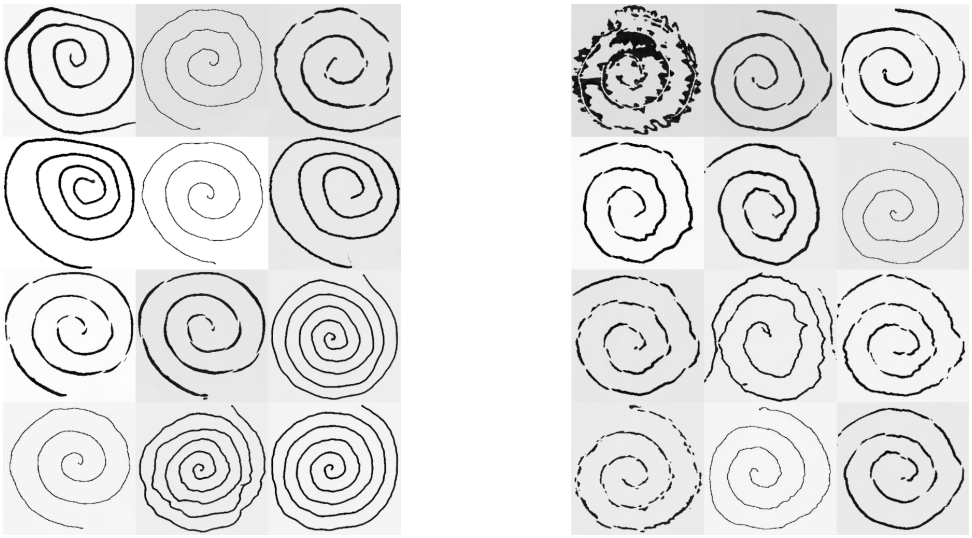
Figure 40. StyleGAN2-ADA + LeCam t-SNE

accessed from Google Drive [2]. The HC model shows sudden changes in the interpolation, while the PD model interpolation is smoother. From the HC interpolation, visual artefacts (darker blotches) can be clearly seen.

[2] https://drive.google.com/drive/folders/1DFT-9D50J37mLkHvKCBl_xvp_fLDnrog?usp=sharing

6.1.3 StyleGAN3

StyleGAN3 models trained for 1400 steps, in which time each model saw 5.6×10^6 images. The HC model achieved a KID of **0.02148** (step 900) and the PD model **0.2113** (step 650). Handpicked samples from the best performing models are shown in Figure 41, larger grids are available in Appendix 2 (Figures 65 and 66).



(a) (b)

Figure 41. Hand-picked examples from the StyleGAN3 best checkpoint. (a) HC, (b) PD.

The samples generated with StyleGAN3 seem to be clearer than the preceding architectures, with no visible darker areas. Otherwise, the same issues persist: inconsistent background colour from sample to sample and partial mode collapse.

The k-nearest neighbour results in Figures 42 and Figures 43 suggest a similar level of overfitting as before. The generated images can be quite close to the real ones, but are never quite the same.

Figure 44 illustrates that for the first day and a half of training, both models slowly decreased their KID scores. After that the KID for both models rises rapidly, indicating that the generators started to produce images of poor quality.

Looking at the loss values, there is a sudden increase in both generator losses and the losses have a larger amplitude (Figure 45). As a result, the discriminator also became unstable. Both models generator loss trended upward and the discriminator downward, indicating non-convergence which ended with the model becoming unstable.

Figure 46 illustrating the discriminator scores for the model shows that for the first few

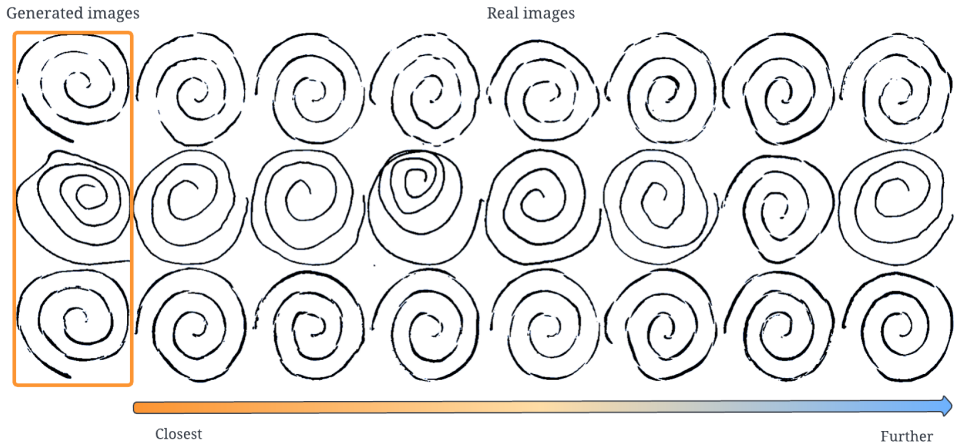


Figure 42. StyleGAN3 HC k-nearest neighbour

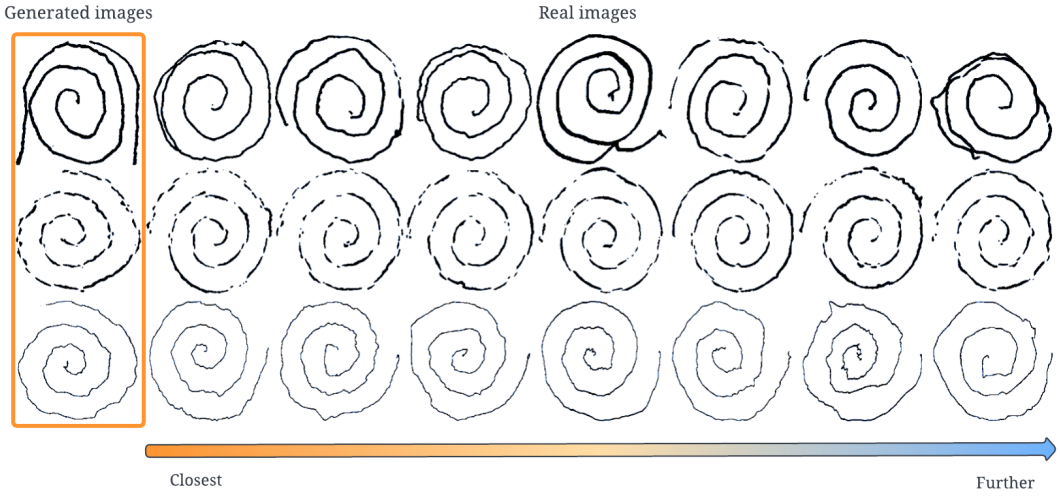


Figure 43. StyleGAN3 PD k-nearest neighbour

hundred steps the scores converged. After which the scores started to diverge from each other. Resulting in the training process becoming unstable and producing images of very poor quality.

The real and StyleGAN3 generated image distributions are not one-to-one (Figure 47). The PD shows some good overlap with the real distribution, but there is a part of the generated PD distribution that is not near the real distribution. The performance of HC seems to be worse than that of PD, as there is almost no overlay between the real and generated distributions.

The latent space interpolation videos for both StyleGAN3 models can be accessed from

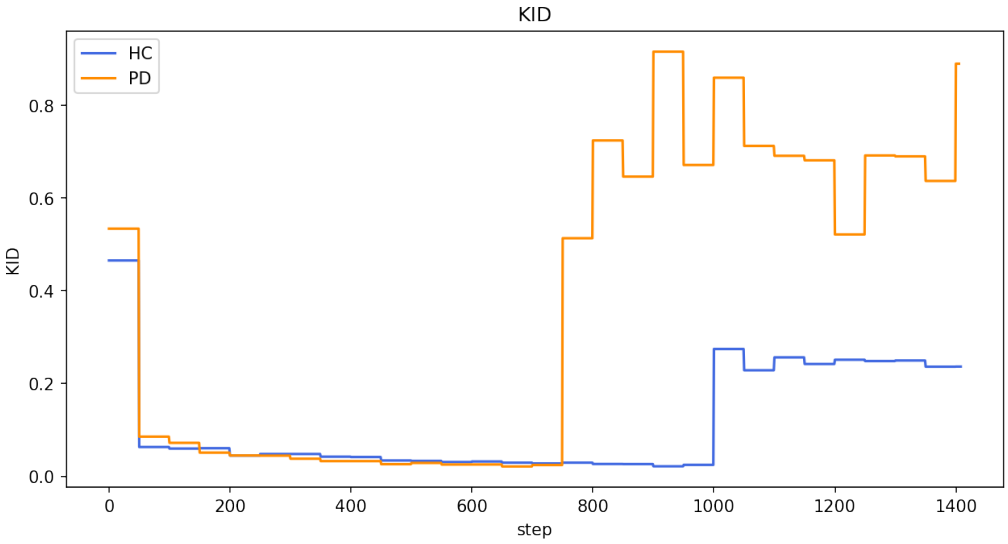


Figure 44. StyleGAN3 KID

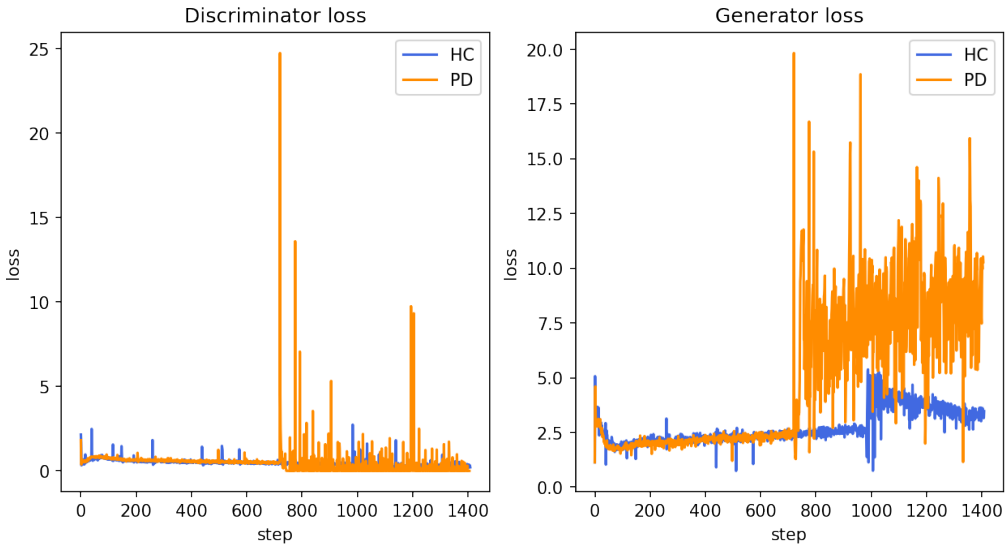


Figure 45. StyleGAN3 (left) discriminator and (right) generator loss

Google Drive [3]. StyleGAN3 interpolations also show that the PD model interpolates more smoothly. From the videos it can also be seen that the generator produces some subtle image artefacts that look like a pattern of lines in the background (most clearly visible in the PD interpolation around *00:30*).

6.1.4 Projected GAN

Projected GAN models trained for 4500 steps with each model seeing 17.7×10^6 images. The best KID of the HC model was **0.001264** (step 1100) and **0.0009285** (step 3950) for

[3] `https://drive.google.com/drive/folders/1DFT-9D50J37mLkHvKCBl_xvp_fLDnrog?usp=sharing`

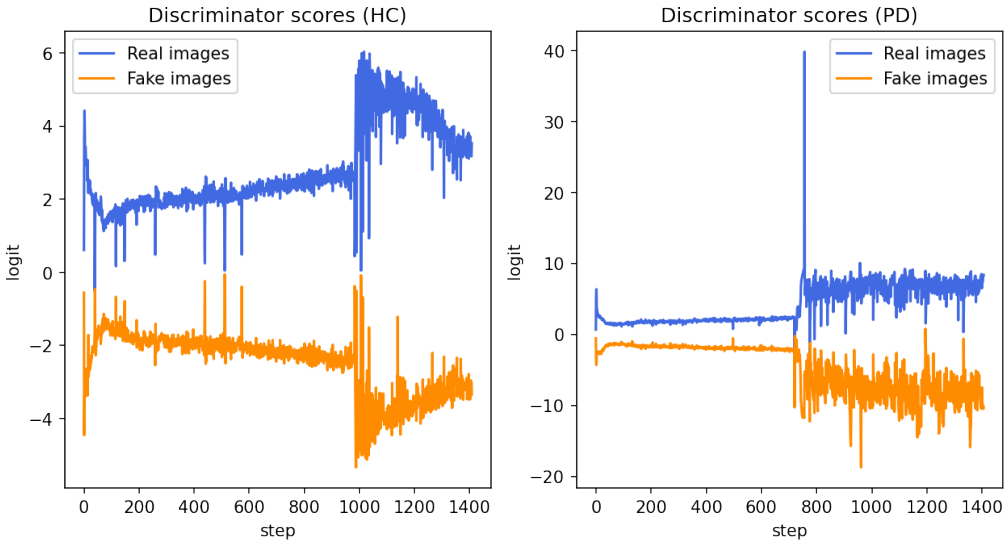


Figure 46. StyleGAN3 HC (left) and PD (right) discriminator scores

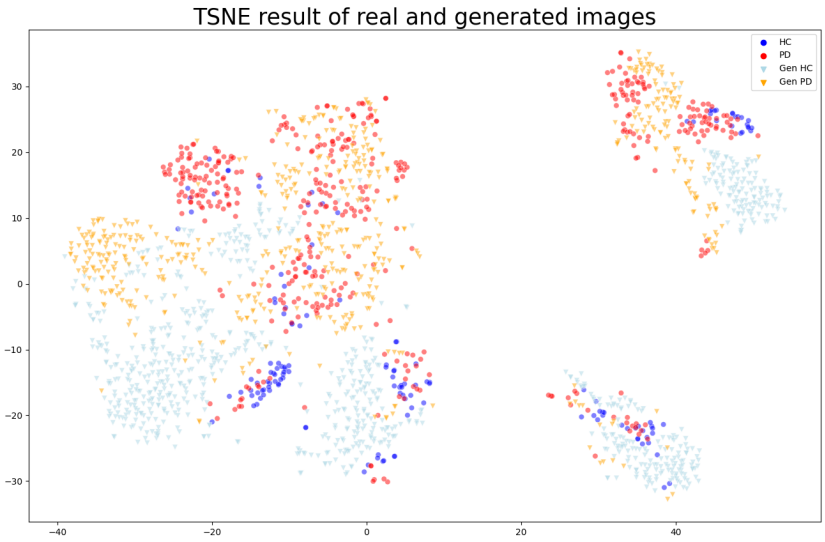
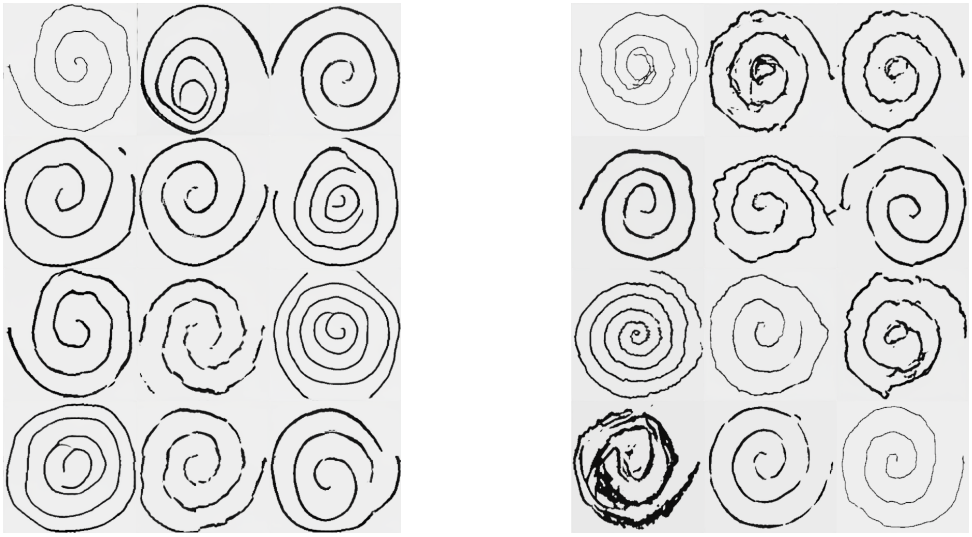


Figure 47. StyleGAN3 t-SNE

the PD model. Handpicked samples from the best checkpoints of Projected GAN are shown in Figure 48, and the larger grid can be viewed in Appendix 2 (Figures 67 and 68).

This architecture seemed to fix the issue of inconsistent background colour and seems to generate a larger diversity of images, but some level of mode collapse can also be seen in this model. Furthermore, the model does not always seem to generate a clear Archimedean spiral; sometimes the image contains concentric circles or two spirals that are intertwined. This was also present in the other architectures but to a much smaller degree.



(a) (b)

Figure 48. Hand-picked examples from Projected GAN best checkpoint. (a) HC, (b) PD.

Figures 49 and 50 show the results of k-nearest neighbours. AS with the previous architectures the Projected GAN models manage to generate images that are novel but take much of their inspiration from the real data. One thing to note is that it is much harder to find areas in the generated spirals that are really close to some part of a real image.

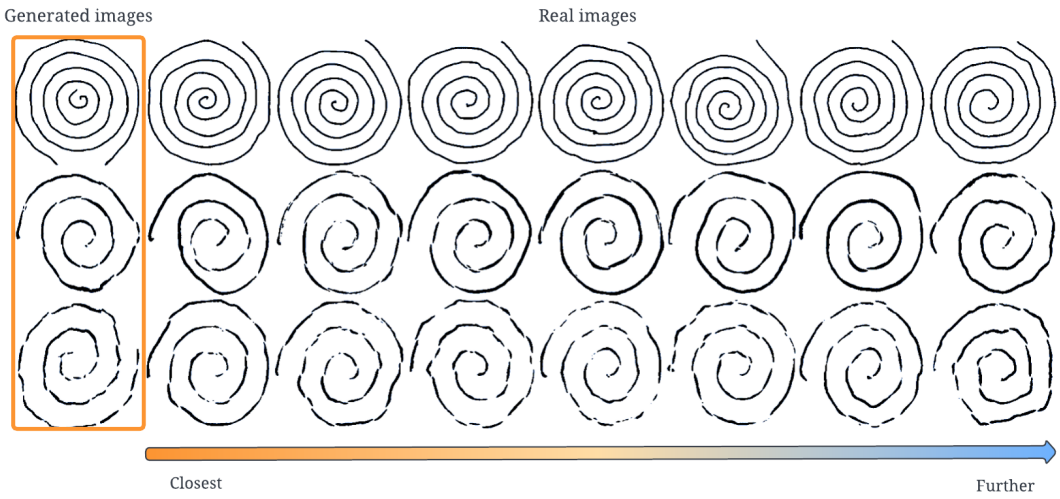


Figure 49. Projected GAN HC k-nearest neighbour

KID scores for Projected GAN models are an order of magnitude lower than those of the other architectures (Figure 51). After a steep drop at the start, both models KID remained stable. Only a marginal increase in KID can be seen in the HC model at the end of training.

From the loss graphs (Figure 52), after an increase in generator loss at the start of training, it started to drop steadily, suggesting that the models start to converge. Projected GAN

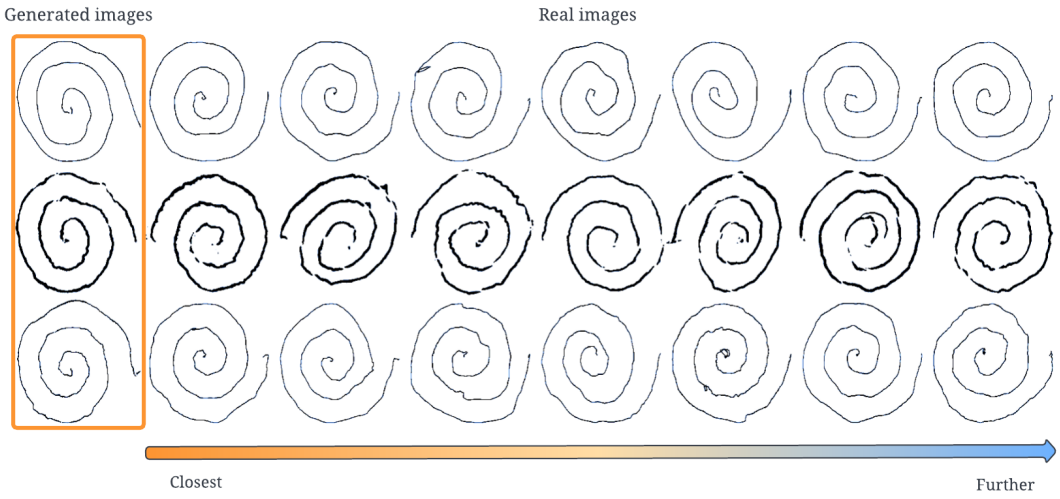


Figure 50. Projected GAN PD k-nearest neighbour

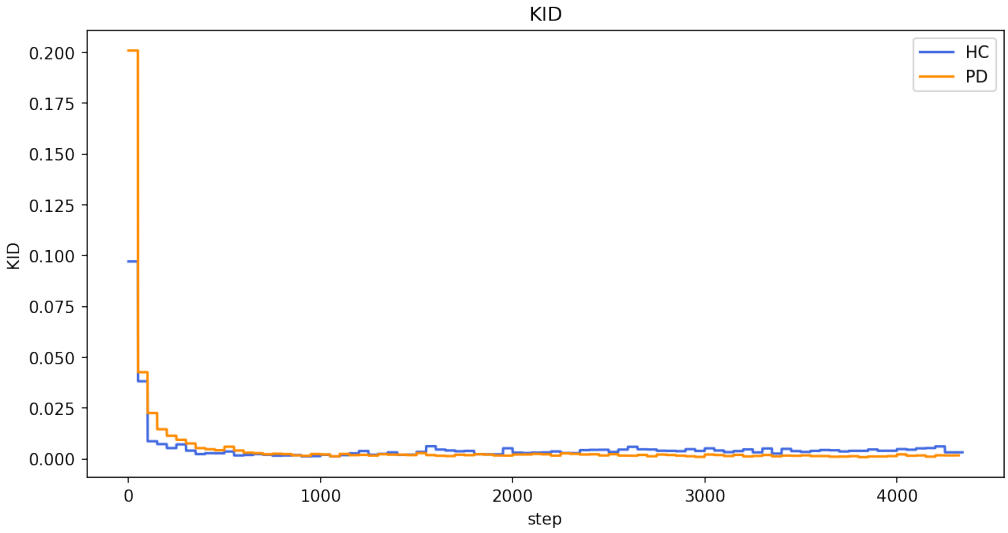


Figure 51. Projected GAN KID

is the only architecture where this occurred. Discriminator loss decreased throughout the training process, as was the case with every other GAN architecture tested. This might lead to a failure mode if the model was further trained.

The discriminator scores in Figure 53 show that the discriminator becomes better at identifying real images, as the training process continues, but worse at identifying fake images. In a sense, it means that the discriminator is really sure of what is a real image, but becomes more and more unsure how to identify what is fake. This might be related to overfitting, because real data sets are quite small, the discriminator might just memorise them. Furthermore, the scores are nowhere near convergence, where the model would have no idea how to differentiate between the images.

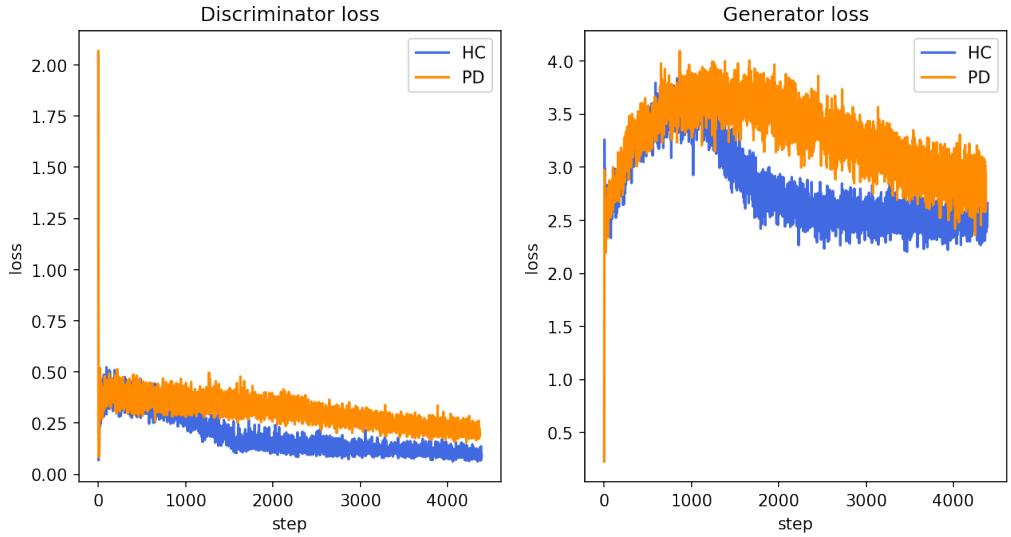


Figure 52. Projected GAN (left) discriminator and (right) generator loss

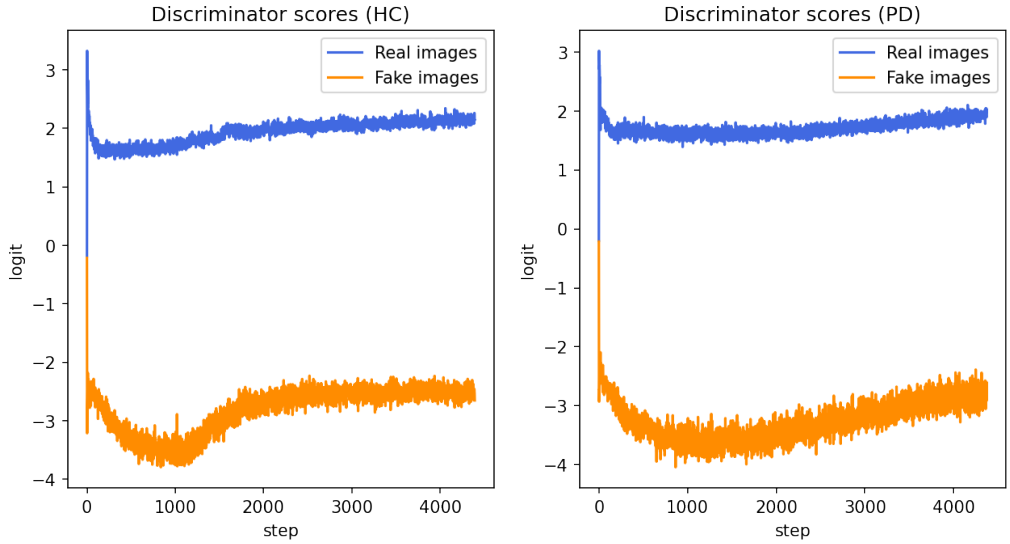


Figure 53. Projected GAN HC (left) and PD (right) discriminator scores

T-SNE plot of the generated distribution of Projected GAN (Figure 54) seems to indicate much better generative performance compared to the other architectures. Both the HC and PD models overlap quite well with their real counterparts, PD slightly better than HC. All generated images form a cluster around their respective class, and there are no separate clusters of generated images in the distribution. Additionally, there seems to be some indication that the real images of HC and PD are more separated from each other than before.

The latent space interpolation videos for both Projected GAN models can be accessed

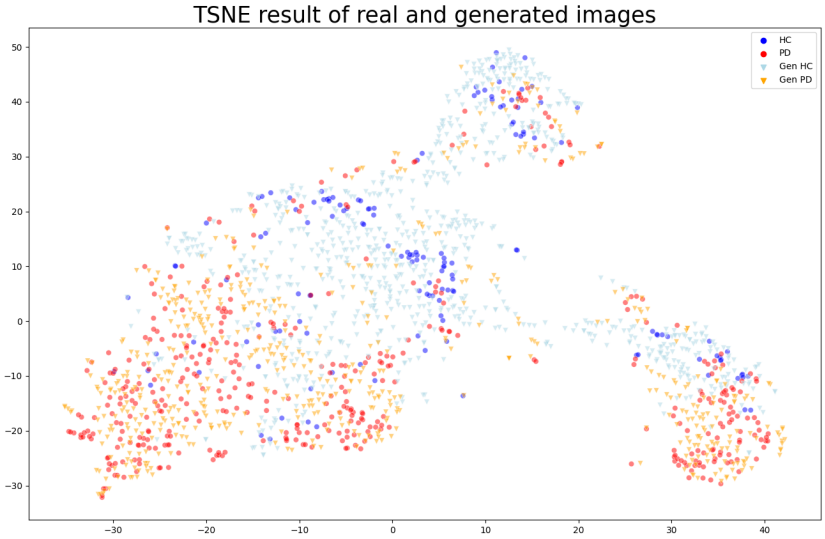


Figure 54. Projected GAN t-SNE

from Google Drive [4]. The interpolations for this architecture are the most varied. This comes with the downside of the generated shape not always being a spiral. Here, the HC interpolates more smoothly than the previous architectures. At the same time, PD interpolates smoothly, but at a faster pace.

6.1.5 Summary

Table 8 summarises the KID scores of the GANs. Based on KID scores and qualitative analysis, Projected GAN seems to capture the distribution the best. Therefore, it seems that it is the most suitable for improving the classification performance of CNNs.

Table 8. Best KID scores of each GAN architecture.

GAN	**KID (↓)**	
	HC	**PD**
StyleGAN2-ADA	0.01416	0.01054
StyleGAN2-ADA + LeCam	0.01826	0.02517
StyleGAN3	0.02148	0.02113
Projected GAN	0.001264	0.0009285

[4] https://drive.google.com/drive/folders/1DFT-9D50J37mLkHvKCBl_xvp_fLDnrog?usp=sharing

6.2 CNN classifier results analysis

This section will analyse the performance of the CNN architectures and augmentation methods. Tables 9 and 10 give an overview of how CNN models performed with each augmentation method on the test set. In the following subsections, each of the augmentation methods will be evaluated.

Table 9. CNN test set sensitivity. Mean scores over five runs. **Bold** indicates the best results and underlined values the second best for a CNN model.

Augmentation method	Sensitivity (%)					
	AlexNet	ResNet	VGG	Inception v3	Xception	DenseNet
None	88.0	94.3	92.6	92.0	90.9	30.3
Traditional	**91.4**	93.7	89.7	92.6	93.1	**53.7**
StyleGAN2-ADA	85.1	90.9	94.3	90.3	93.7	28.6
StyleGAN2-ADA + LeCam	88.6	95.4	93.7	**94.3**	95.4	29.7
StyleGAN3	89.1	94.3	94.9	93.1	93.7	17.1
Projected GAN	90.3	**96.6**	**95.4**	92.6	**96.6**	45.1

Table 10. CNN test set specificity. Mean scores over five runs. **Bold** indicates the best results and underlined values the second best for a CNN model.

Augmentation method	Specificity (%)					
	AlexNet	ResNet	VGG	Inception v3	Xception	DenseNet
None	73.1	68.0	66.9	69.1	65.7	97.7
Traditional	72.0	**76.0**	**73.1**	**76.6**	**72.6**	92.6
StyleGAN2-ADA	**75.4**	71.4	68.0	**76.6**	68.0	97.1
StyleGAN2-ADA + LeCam	68.6	69.1	66.9	69.1	63.4	93.1
StyleGAN3	73.7	72.0	72.0	72.6	65.7	**100.0**
Projected GAN	68.0	69.7	65.1	66.3	59.4	96.0

6.2.1 Baseline evaluation

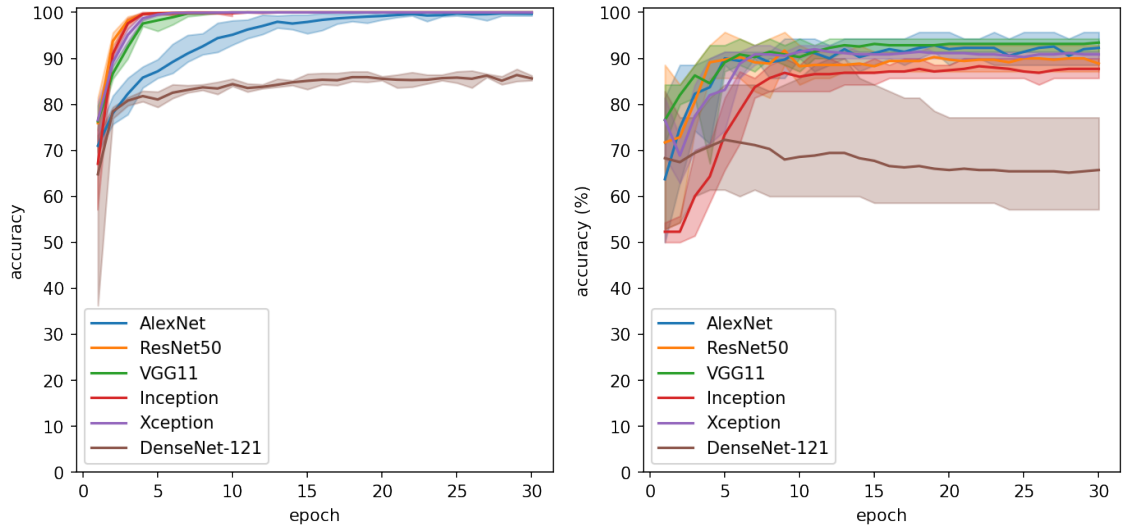
The original training set was used to establish a baseline that shows how CNN architectures perform without augmented data. The best sensitivity was produced by ResNet, which managed to achieve a sensitivity of 94.3% and a specificity of 68.0%. AlexNet produced the highest specificity with 73.1%. The other architectures produced similar but slightly worse results, with one exception DenseNet. DenseNet was the only clear outlier in performance, with by far the highest specificity 97.7% and the lowest sensitivity 30.3%.

The accuracy of the training set and the validation set, throughout the model training, shows

that DenseNet is the only model where the performance of the validation set deteriorates as the training progresses (Figure 55). The other models have similar accuracy curves for both the training and the validation set.

Excluding DenseNet, the accuracy of the AlexNet training set increases slower than the others, and there is no correlation between that training accuracy and the validation accuracy of AlexNet, as the validation accuracy performance does not correspond to the training accuracy. Secondly, while the training accuracy of ResNet increased rapidly in the first few epochs, the validation accuracy does not increase in tandem with it and takes more epochs to stabilise. ResNet had, on average, the lowest validation accuracy, but performed the best on the test set.

Every architecture, with the exception of DenseNet, manages to achieve 100% accuracy on the training set. It indicates that the models have probably memorised the training set and are prone to overfitting.



(a) (b)

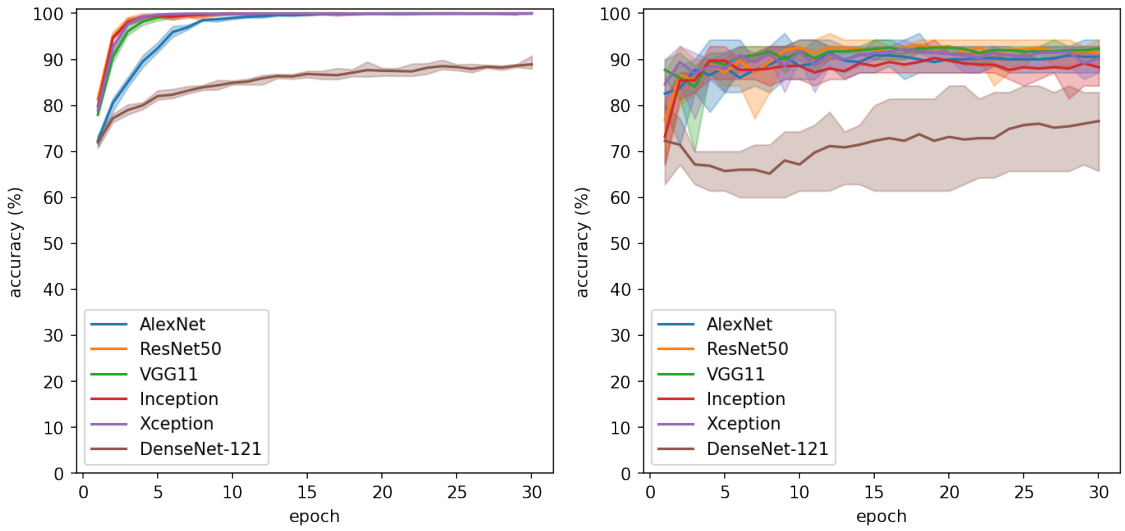
Figure 55. Baseline CNN accuracy. (a) Train set, (b) validation set.

6.2.2 Augmentation evaluation

Traditional augmented data achieved better specificity with each architecture compared to baseline. With the exception of AlexNet and DenseNet, where the specificity decreased by 1.1% and 5.1% respectively. The specificity of the other models increased by 6–7%. Sensitivity was more of a mixed bag. Three of the six models saw an increase, the most significant being AlexNet 3.4% and Xception 2.2%. Inception v3 saw a marginal increase of 0.6%. The sensitivity of VGG decreased the most by 2.9% and ResNet decreased by 0.6%. Furthermore, four of the six architectures achieve their best specificity score, and two

architectures record their best sensitivity score, when used with traditional augmentation.

The accuracy of training and validation (Figure 56) tells a similar story as with the baseline data set. One key difference being DenseNet, it is the only CNN architecture where both training validation accuracy show growth after 30 epochs, indicating that training longer might improve the results of the test set. Every other architecture seems to hit a plateau before the 10th epoch.



(a) (b)

Figure 56. Traditional augmentation CNN accuracy. (a) Train set, (b) validation set.

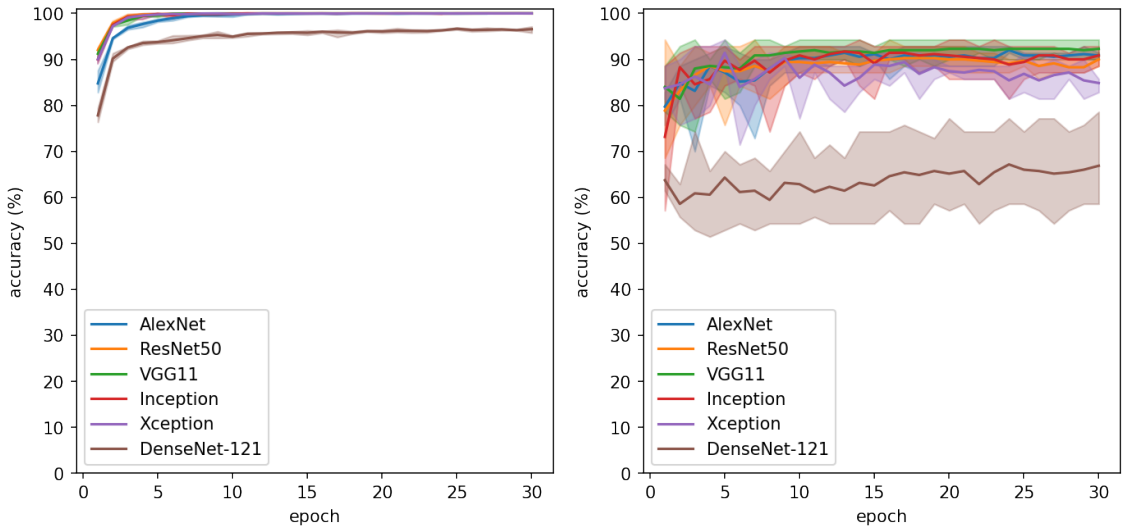
6.2.3 StyleGAN2-ADA augmentation evaluation

StyleGAN2-ADA augmentations lead to two of the CNN architectures posting their best specificity score. Compared to baseline, StyleGAN2-ADA posts 2–7% higher specificity and 2–4% lower sensitivity, excluding DenseNet. DenseNet is once again an outlier, when compared to other CNN architectures, and performs slightly worse than the baseline.

When comparing traditional and StyleGAN2-ADA augmentation, StyleGAN2-ADA outperforms the sensitivity twice out of the six times, but these models do not manage to match the specificity of traditional augmentation methods. The other CNN model has a lower sensitivity of around 2–7%, with the exception of DenseNet, where the sensitivity is lower by 25%. Specificity is higher in two CNN models by approximately 3–5% and is equal in the case of Inception v3. The specificity of the other three models is lower by around 5–7%.

The training accuracy of the StyleGAN2-ADA augmented data sets increases to more than 90% for all CNN architectures after the first epoch, indicating that something in the

generated data makes it easy for CNN to classify (Figure 57). The validation accuracy is generally really similar to the traditional augmentation accuracy (Figure 56) but slightly lower than the baseline and traditional augmentation.



(a) (b)

Figure 57. StyleGAN2-ADA augmentation CNN accuracy. (a) Train set, (b) validation set.

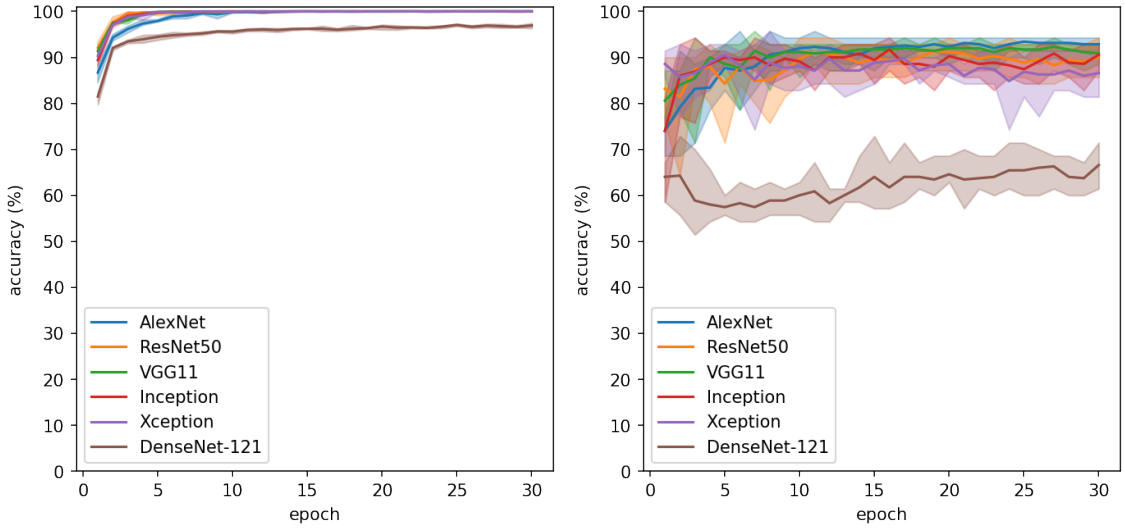
StyleGAN2-ADA generated data seem to encode some information about the differences between PD and HC. It did improve the performance of most of the CNN architectures over the baseline and in some cases even outperformed traditional augmentation. StyleGAN2-ADA augmented images are most likely held back by the lack of diversity (partial mode collapse) and the generator not being able to capture the original image distribution, which leads to the generators creating images outside of the distribution and confusing the classifier.

6.2.4 StyleGAN2-ADA + LeCam augmentation evaluation

StyleGAN2-ADA + LeCam managed to achieve the best test set sensitivity with Inception v3, out of all augmentation methods. In most cases, StyleGAN2-ADA + LeCam provided a increase in sensitivity, around 0.5–4%, when compared to the baseline. Specificity matches or slightly increases the baseline with three CNNs. The other three CNN see a drop of around 2–5%.

The only times that StyleGAN2-ADA + LeCam improves on the traditional augmentation is the sensitivity of ResNet, VGG, Inception v3 and Xception and the specificity of DenseNet. All other metrics are lower than traditional augmentation. Compared to StyleGAN2-ADA, the results indicate that the use of StyleGAN2-ADA + LeCam results in better sensitivity but worse specificity.

The training and validation accuracy of StyleGAN2-ADA + LeCam (Figure 58) is similar to StyleGAN2-ADA. One notable difference being the standard deviation of DenseNet which is smaller.



(a) (b)

Figure 58. StyleGAN2-ADA + LeCam augmentation CNN accuracy. (a) Train set, (b) validation set.

Again, the generators lack the ability to generate a sufficiently diverse set of meaningful images. On the basis of the sensitivity and specificity scores, this seems to be more severe in the case of HC images than for PD images. As with StyleGAN2-ADA, partial mode collapse and image artefacts deteriorate the performance of the classification task.

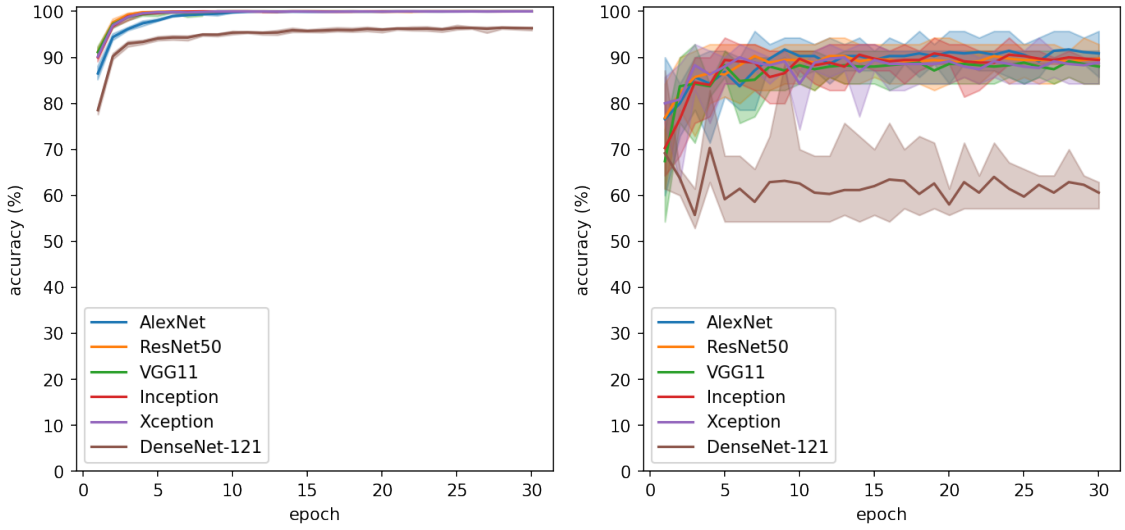
6.2.5 StyleGAN3 augmentation evaluation

StyleGAN3 results exceed or match the baseline, with the exception of DenseNet. DenseNet is once again an outlier, having a very high specificity and very low sensitivity.

Compared to traditional augmentation, using StyleGAN3 generated data, with ResNet, VGG or Inception v3, matches or increases the sensitivity by 0.5–2% and decreases the specificity by 7–9%. AlexNet saw a decrease in sensitivity of 3.4% and an increase in specificity of 1.1%. The sensitivity and specificity of Xception decrease by 1.7% and 6.9% respectively.

Compared to StyleGAN2-ADA and StyleGAN2-ADA + LeCam, StyleGAN3 beats the sensitivity with AlexNet and VGG. Regarding specificity, using StyleGAN3 augmentations with ResNet and VGG outperforms the other StyleGAN-based augmentations. Otherwise, the StyleGAN3 results are second in terms of performance, out of the three options.

The StyleGAN3 training set and validation set plots (Figure 59) are similar to previous StyleGAN-based augmentation methods. The training set is learnt in a few epochs and the validation hits a plateau around 90% accuracy. One visible difference is that DenseNet validation accuracy does not show a growing trend.



(a) (b)

Figure 59. StyleGAN3 augmentation CNN accuracy. (a) Train set, (b) validation set.

The lack of diversity in StyleGAN3 generated images prevents it from consistently outperforming traditional augmentation methods in both test metrics, usually only beating the sensitivity score. On the other hand, the baseline is beaten consistently, which is something that cannot be said about the other StyleGAN-based augmentations. Indicating that StyleGAN3 generates better quality images that have more meaningful features.

6.2.6 Projected GAN augmentation evaluation

Out of all the augmentation methods, using Projected GAN with ResNet, VGG and Inception v3 produces the highest sensitivity. Furthermore, achieving the highest overall sensitivity of 96.6%. DenseNet is again the outlier of the CNN architectures.

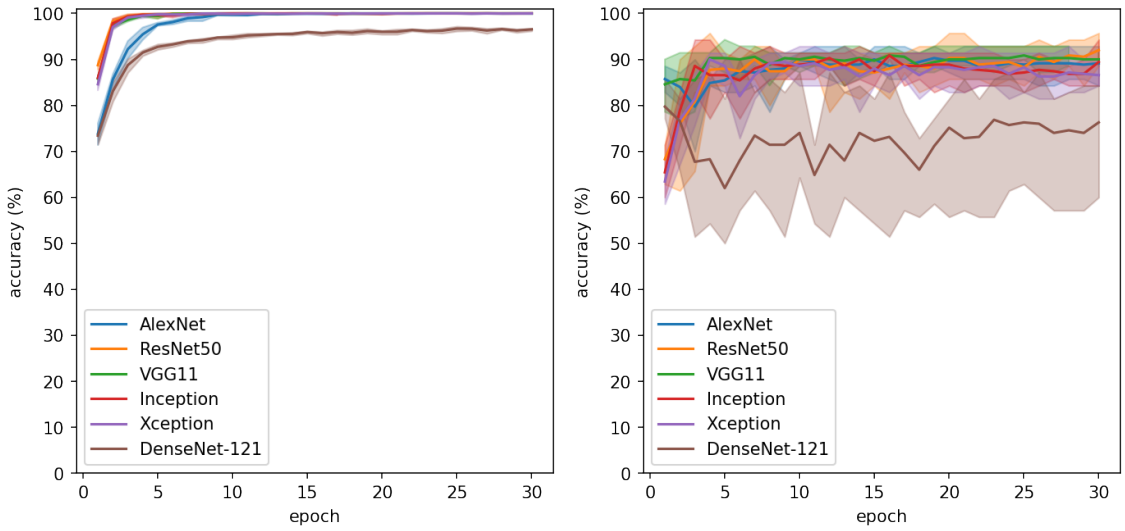
Projected GAN manages to beat the sensitivity of all the baseline CNN models by around 0.6–5.7% but falls behind in specificity in all of them with the exception of ResNet and DenseNet.

The high sensitivity scores of Projected GAN augmentation mean that the sensitivity of traditional augmentation is beaten by 2.9–5.7% in the case of ResNet, VGG and Xception and matched in the case of Inception v3. AlexNet and DenseNet lose out to the sensitivity of traditional methods but are the second best after them. The specificity is lower in all

cases, except DenseNet, by approximately 4–13.2%.

Compared to the other GAN augmentation methods, Projected GAN sensitivity is higher everywhere except Inception v3. However, the specificity is the lowest of the GAN-based methods.

Training accuracy (Figure 60a) shows a lower starting point at the beginning of training and a steady increase throughout training. Showing that the CNN models have to do a bit more work to correctly learn how to identify the classes, when compared to the other GAN augmentation methods. The validation accuracy hovers a bit below 90% for most CNN models. DenseNet validation accuracy, once again, is the only one that grows until the end of training.



(a) (b)

Figure 60. Projected GAN augmentation CNN accuracy. (a) Train set, (b) validation set.

Overall, Projected GAN augmentation resulted in better sensitivity and worse specificity. Showing that Projected GAN has trouble creating meaningful HC images but does a better job at creating PD images. The increase in sensitivity might be the result of the PD generator not producing as many artefacts and a more diverse distribution of images than other GANs.

6.3 Discussion

All GAN architectures performing worse for HC images is expected. Taking into account that there were almost 3 times less training data available than for PD images. Further data collection would help alleviate the problem. On the other hand, the KID score of StyleGAN architectures degrades quicker with PD data. This could be caused by the PD

training set having a larger variety of images, as a PD drawing might have severe tremors or nearly no visible tremors. The HC drawings are usually similar to each other. When the generator starts producing a smaller set of images, the features extracted from the generated images and the real images to calculate the KID start to differ more, leading to a worse KID score. Another explanation would be that the augmentations from the discriminator augmentations start to leak so much that it impacts the KID score.

Regarding the discriminator augmentations used in StyleGAN, it can be noted that the StyleGAN generators produce images with varying contrast and illumination compared to the training set, where every image has the same colour values for the spiral and the background. This seems also to be a case of the augmentations leaking into the generated images. Changing the discriminator augmentation pipeline might solve these problems. Projected GAN uses the FastGAN configuration in this thesis, which does not use image augmentation in the training process and, therefore, cannot leak into the generated distribution.

From the experiments carried out in this work, it can be seen that GAN augmentation methods seem to have more of an impact on the sensitivity of classification, while traditional augmentation has effects more the specificity of classification. This might be due to the lack of HC data to train the GAN models, which led the HC generator to perform not as well as the PD generator. Overall, for generating additional training data, two of the GAN architectures stand out. StyleGAN3 seems to improve both test metrics most consistently, out of all GANs, often improving the sensitivity over traditional methods and achieving the second best specificity after traditional methods. Projected GAN, on the other hand, consistently showed the highest sensitivity out of all the augmentation methods.

The training accuracy that increases more quickly with GAN generated data might be caused by the use of two generators to generate data. The generated data might include generator-specific artefacts that make the classification trivial for CNNs. For example, the colour values might not exactly match between generators, or there is some pattern in the images that is not easily noticeable with the naked eye. Another contributing factor is the limited variety of images generated by GANs, as exact of near copies of images do not give any new information to the CNN. For this reason, adding more generated images to the training set would most likely not improve the classification performance.

Selecting the GAN checkpoint based on the KID score might also not be the most optimal. StyleGAN3 had the highest KID score, but managed to outperform GAN models with lower KID scores in classification. Furthermore, Projected GAN had by far the best KID scores, but this did not translate into better classification performance across the board.

One thing to note is that the traditionally augmented images have more variance in the position, rotation, and scale of the spiral, while the GAN generated images are positioned in the centre of images, and in some cases are always rotated in the same way. One way to quickly increase the variance of generated images in the future would be to apply traditional augmentations to them. Furthermore, minimal hyperparameter tuning was done for GANs in this work, as training takes days. This, coupled with the fact that GAN training is highly volatile to the selected hyperparameters, suggests that it might be possible to train more optimal GANs for spiral generation.

Traditional augmentation has the advantage of being easier to apply to any training process. Modern tools make defining augmentation pipelines really easy. Additionally, the computing power needed for traditional augmentation is negligible compared to training a GAN. More importantly, the near-instant nature of traditional augmentations results in a faster feedback loop, allowing for more iterations. Modern machine learning frameworks also support data augmentation during training, making it even easier to use data augmentations.

Of the six CNN architectures used, ResNet, VGG, Inception v3 and Xception perform the best. The results for each of the augmentation methods are really close to each other, and there is no constant best that outperforms others with each augmentation method. AlexNet gives slightly lower results than the four previously mentioned models. This makes sense, as AlexNet is the oldest, shallowest, and most rudimentary of all CNN architectures tested. The only consistent outlier throughout all the testing was DenseNet, which consistently performed much worse than other models. However, there were indications that a longer training time might improve the results.

Compared to state-of-the-art solutions using CNNs, CNNs trained with GAN-generated data do not perform better. In this thesis, only the spiral shape was used as input to CNN, while state-of-the-art approaches use data enhancement to add additional information, such as pressure and acceleration, to images [12, 8]. This might be one of the reasons for the lower performance, as the shape of the spiral encodes information only about the tremor and does not encode information about the speed of writing or the pressure on the drawing surface.

Other approaches use multiple different drawing and writing tests to make the final prediction [11], allowing CNN to have more information about the individual than a single drawing approach. One thing to note about [11] is that their single assessment Archimedean spiral CNN (AlexNet) performed worse than the AlexNet trained with GAN generated data in this thesis with a 2.1% increase in sensitivity and 0.8% increase in specificity. Indicting that adding the proposed GAN augmentation method to the mentioned methodologies

could further improve Parkinson's detection performance.

The GAN models trained and used in this thesis have been made publicly available and can be accessed through GitHub [5]. CNN models are also made public on Google Drive [6].

[5] https://github.com/Erikdzo/Parkinsons-Archimedean-spirals-GAN-models

[6] https://drive.google.com/drive/folders/1yMI_V9KgPR46FwVsYOTpdVDgITc0mj3V?usp=sharing

7. Summary

The goal of the present thesis was to analyse the effectiveness of different GAN architectures for Archimedean spiral drawing generation, use the generated data to train CNN models for image classification of Parkinson's disease, and analyse how the GAN-based augmentations affect the performance of the CNNs when compared to no augmentations and traditional augmentations.

The research was based on five Parkinson's patients' handwriting and drawing data sets. In total, the data collected contained 312 images of healthy controls and 618 images of Parkinson's patients.

For image generation four different GAN architectures were used: StyleGAN2-ADA, StyleGAN2-ADA + LeCam, StyleGAN3, Projected GAN. For image classification, six CNN architectures were used: AlexNet, VGG, Inception v3, ResNet, Xception and DenseNet.

The results of image generation show that quantitatively Projected GAN produces an order of magnitude better KID score and the t-SNE analysis shows that it mimics the original distribution the best. Qualitative analysis shows that the highest fidelity images are produced by StyleGAN3 and Projected GAN, with minimal visual artefacts or corruption visible in the generated image. All the GAN models showed signs of partial mode collapse and non-convergence, indicating lack of sufficient training data.

In general, the generated images improved the sensitivity of the classification more than the specificity. The best performing generated images were produced by StyleGAN3 and Projected GAN. Projected GAN produced the highest sensitivity of all the augmentation methods 96.6%, while not improving the specificity over the baseline. StyleGAN3 often produced high sensitivity, exceeding traditional augmentation and second behind Projected GAN result, and was second in specificity only behind the traditional methods. These results indicate that there is potential in using GANs to generate novel training data for the classification task.

Of all CNN models, the best results were produced by ResNet, VGG, Xception and

Inception v3. Between these models, there were no clear winners. DenseNet was the only outlier in terms of performance, but the analysis showed that a change to training duration might fix this. Therefore, more testing needs to be done with this architecture.

The highly sensitive models produced by using GAN generated data show that the proposed methodology can serve as a decision support tool for medical professionals. These highly sensitive models could potentially increase confidence and save resources (time and cost) spent in the initial phases of the diagnosis.

This work opens many further research directions for using GANs as a data augmentation tool. A thorough exploration of the GAN hyperparameter space would allow for GANs that better capture the original data distribution. Also, at what point in the training do the generated data provide the most discriminative power between healthy controls and Parkinson's patients, does it match with the lowest KID score, or is there a better indicator?

Furthermore, different types of handwriting and drawing tests could be used for GAN training. The digitally collected images in the GAN training set could also be enhanced with pressure and velocity information, so that the generator would also learn to generate the added information. Another idea is to train the GANs on an augmented distribution, so that there would be more training data available for the GANs.

Bibliography

[1] J Jankovic. "Parkinson's disease: clinical features and diagnosis". In: *Journal of Neurology, Neurosurgery & Psychiatry* 79.4 (2008), pp. 368–376. ISSN: 0022-3050. DOI: 10.1136/jnnp.2007.131045. eprint: https://jnnp.bmj.com/content/79/4/368.full.pdf. URL: https://jnnp.bmj.com/content/79/4/368.

[2] Jack Chen. "Parkinson's Disease: Health-Related Quality of Life, Economic Cost, and Implications of Early Treatment". In: *The American journal of managed care* 16 Suppl Implications (Mar. 2010), S87–93.

[3] Valery L Feigin et al. "Global, regional, and national burden of neurological disorders during 1990–2015: a systematic analysis for the Global Burden of Disease Study 2015". In: *The Lancet Neurology* 16.11 (2017), pp. 877–897. ISSN: 1474-4422. DOI: https://doi.org/10.1016/S1474-4422(17)30299-5. URL: https://www.sciencedirect.com/science/article/pii/S1474442217302995.

[4] ER Dorsey et al. "GBD 2016 Parkinson's Disease Collaborators. Global, regional, and national burden of Parkinson's disease, 1990-2016: a systematic analysis for the Global Burden of Disease Study 2016". In: *Lancet Neurol* 17.11 (2018), pp. 939–953.

[5] Sven Nomm et al. "Detailed Analysis of the Luria's Alternating SeriesTests for Parkinson's Disease Diagnostics". In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)* (2018), pp. 1347–1352.

[6] Peter Drotár et al. "Evaluation of handwriting kinematics and pressure for differential diagnosis of Parkinson's disease". In: *Artificial Intelligence in Medicine* 67 (2016), pp. 39–46. ISSN: 0933-3657. DOI: https://doi.org/10.1016/j.artmed.2016.01.004. URL: https://www.sciencedirect.com/science/article/pii/S0933365716000063.

[7] Iqra Kamran et al. "Handwriting dynamics assessment using deep neural network for early identification of Parkinson's disease". In: *Future Generation Computer Systems* 117 (2021), pp. 234–244. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2020.11.020. URL: https://www.sciencedirect.com/science/article/pii/S0167739X20330442.

[8] Sven Nomm et al. “Deep CNN Based Classification of the Archimedes Spiral Drawing Tests to Support Diagnostics of the Parkinson’s Disease”. In: *IFAC-PapersOnLine* (2020).

[9] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[10] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2014. arXiv: 1312.6114 [stat.ML].

[11] Clayton R. Pereira et al. “Handwritten dynamics assessment through convolutional neural networks: An application to Parkinson’s disease identification”. In: *Artificial Intelligence in Medicine* 87 (2018), pp. 67–77. ISSN: 0933-3657. DOI: https://doi.org/10.1016/j.artmed.2018.04.001. URL: https://www.sciencedirect.com/science/article/pii/S093336571730369X.

[12] Mohamad Alissa et al. “Parkinson’s disease diagnosis using convolutional neural networks and figure-copying tasks”. In: *Neural Computing and Applications* 34.2 (Jan. 2022), pp. 1433–1453. ISSN: 1433-3058. DOI: 10.1007/s00521-021-06469-7. URL: https://doi.org/10.1007/s00521-021-06469-7.

[13] Chen-Yi Lu, Dan Jeric Arcega Rustia, and Ta-Te Lin. “Generative Adversarial Network Based Image Augmentation for Insect Pest Classification Enhancement”. In: *IFAC-PapersOnLine* 52.30 (2019). 6th IFAC Conference on Sensing, Control and Automation Technologies for Agriculture AGRICONTROL 2019, pp. 1–5. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2019.12.406. URL: https://www.sciencedirect.com/science/article/pii/S2405896319323109.

[14] Yu-Xiong Wang et al. *Low-Shot Learning from Imaginary Data*. 2018. arXiv: 1801.05401 [cs.CV].

[15] Maayan Frid-Adar et al. “GAN-based Synthetic Medical Image Augmentation for increased CNN Performance in Liver Lesion Classification”. In: *CoRR* abs/1803.01229 (2018). arXiv: 1803.01229. URL: http://arxiv.org/abs/1803.01229.

[16] Ali Madani et al. “Chest x-ray generation and data augmentation for cardiovascular abnormality classification”. In: *Medical Imaging 2018: Image Processing*. Ed. by Elsa D. Angelini and Bennett A. Landman. Vol. 10574. International Society for Optics and Photonics. SPIE, 2018, pp. 415–420. DOI: 10.1117/12.2293971. URL: https://doi.org/10.1117/12.2293971.

[17] C. R. Pereira et al. “A New Computer Vision-based Approach to Aid the Diagnosis of Parkinson’s Disease”. In: *Computer Methods and Programs in Biomedicine* 136 (2016), pp. 79–88.

[18] Clayton Reginaldo Pereira et al. “Deep Learning-Aided Parkinson’s Disease Diagnosis from Handwritten Dynamics”. In: *2016 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)* (2016), pp. 340–346.

[19] Muhammed Isenkul, Betul Sakar, and O. Kursun. “Improved Spiral Test Using Digitized Graphics Tablet for Monitoring Parkinson’s Disease”. In: May 2014. DOI: 10.13140/RG.2.1.1898.6005.

[20] Betul Sakar et al. “Collection and Analysis of a Parkinson Speech Dataset With Multiple Types of Sound Recordings”. In: *Biomedical and Health Informatics, IEEE Journal of* 17 (July 2013), pp. 828–834. DOI: 10.1109/JBHI.2013.2245674.

[21] Poonam Zham et al. “Distinguishing Different Stages of Parkinson’s Disease Using Composite Index of Speed and Pen-Pressure of Sketching a Spiral”. In: *Frontiers in Neurology* 8 (2017). ISSN: 1664-2295. DOI: 10.3389/fneur.2017.00435. URL: https://www.frontiersin.org/article/10.3389/fneur.2017.00435.

[22] Abhay Yadav et al. *Stabilizing Adversarial Nets With Prediction Methods*. 2017. DOI: 10.48550/ARXIV.1705.07364. URL: https://arxiv.org/abs/1705.07364.

[23] Martin Arjovsky and Léon Bottou. *Towards Principled Methods for Training Generative Adversarial Networks*. 2017. DOI: 10.48550/ARXIV.1701.04862. URL: https://arxiv.org/abs/1701.04862.

[24] Tero Karras et al. *Analyzing and Improving the Image Quality of StyleGAN*. 2020. arXiv: 1912.04958 [cs.CV].

[25] Tero Karras, Samuli Laine, and Timo Aila. *A Style-Based Generator Architecture for Generative Adversarial Networks*. 2018. DOI: 10.48550/ARXIV.1812.04948. URL: https://arxiv.org/abs/1812.04948.

[26] Tero Karras et al. *Training Generative Adversarial Networks with Limited Data*. 2020. arXiv: 2006.06676 [cs.CV].

[27] Hung-Yu Tseng et al. *Regularizing Generative Adversarial Networks under Limited Data*. 2021. DOI: 10.48550/ARXIV.2104.03310. URL: https://arxiv.org/abs/2104.03310.

[28] Tero Karras et al. “Alias-Free Generative Adversarial Networks”. In: *Proc. NeurIPS*. 2021.

[29] Axel Sauer et al. “Projected GANs Converge Faster”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[30] Y. Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[31] Md. Zahangir Alom et al. “A State-of-the-Art Survey on Deep Learning Theory and Architectures”. In: *Electronics* 8 (Mar. 2019), p. 292. DOI: 10.3390/electronics8030292.

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[33] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. DOI: 10.48550/ARXIV.1409.1556. URL: https://arxiv.org/abs/1409.1556.

[34] Christian Szegedy et al. *Going Deeper with Convolutions*. 2014. DOI: 10.48550/ARXIV.1409.4842. URL: https://arxiv.org/abs/1409.4842.

[35] Christian Szegedy et al. *Rethinking the Inception Architecture for Computer Vision*. 2015. DOI: 10.48550/ARXIV.1512.00567. URL: https://arxiv.org/abs/1512.00567.

[36] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385.

[37] François Chollet. *Xception: Deep Learning with Depthwise Separable Convolutions*. 2016. DOI: 10.48550/ARXIV.1610.02357. URL: https://arxiv.org/abs/1610.02357.

[38] Gao Huang et al. *Densely Connected Convolutional Networks*. 2016. DOI: 10.48550/ARXIV.1608.06993. URL: https://arxiv.org/abs/1608.06993.

[39] Danillo Pereira et al. “A step towards the Automated Diagnosis of Parkinson’s Disease: Analyzing Handwriting Movements.” In: June 2015.

[40] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. *On Aliased Resizing and Surprising Subtleties in GAN Evaluation*. 2021. DOI: 10.48550/ARXIV.2104.11222. URL: https://arxiv.org/abs/2104.11222.

[41] Mikołaj Bińkowski et al. *Demystifying MMD GANs*. 2018. DOI: 10.48550/ARXIV.1801.01401. URL: https://arxiv.org/abs/1801.01401.

[42] Andrew Brock, Jeff Donahue, and Karen Simonyan. *Large Scale GAN Training for High Fidelity Natural Image Synthesis*. 2018. DOI: 10.48550/ARXIV.1809.11096. URL: https://arxiv.org/abs/1809.11096.

[43] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[44] Moises Diaz et al. "Sequence-based dynamic handwriting analysis for Parkinson's disease detection with one-dimensional convolutions and BiGRUs". In: *Expert Systems with Applications* 168 (2021), p. 114405. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2020.114405. URL: https://www.sciencedirect.com/science/article/pii/S0957417420310757.

Appendix 1 - Non-exclusive licence for reproduction and publication of a graduation thesis [1]

I Erik Dzotsenidze

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis "Generative Adversarial Networks as a Data Augmentation Tool for CNN-based Parkinson's Disease Diagnostics", supervised by Elli Valla and Sven Nõmm
   1.1 to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2 to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

10.05.2022

[1] The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 - GAN sample grids

Here are all the grids of the best checkpoints of each GAN architecture. Zoom in is recommended to inspect the spirals.

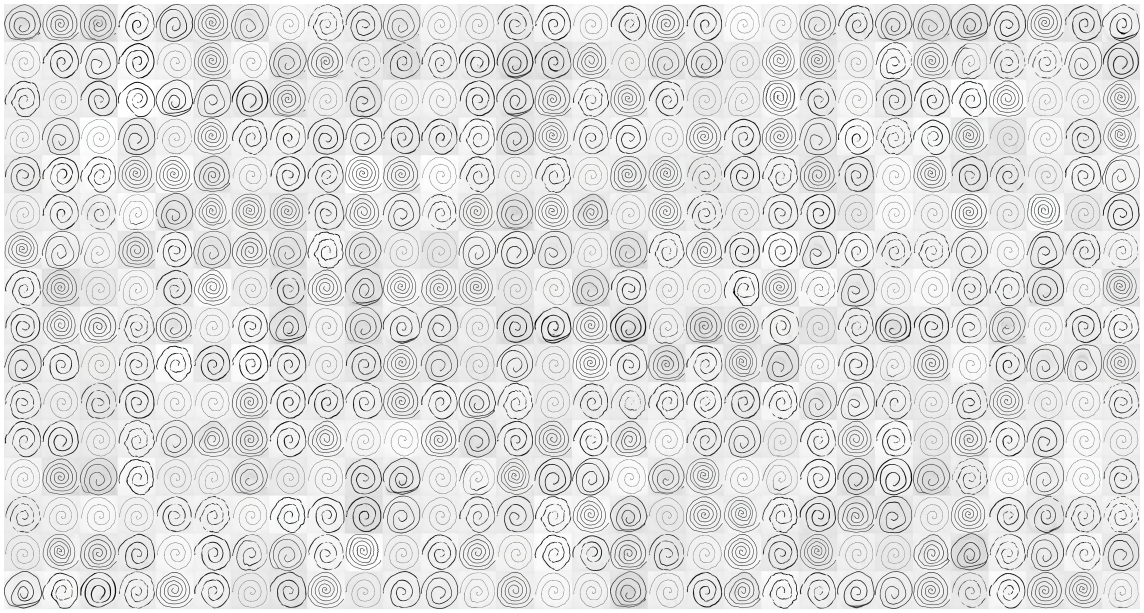


Figure 61. Grid of images from StyleGAN2-ADA HC best checkpoint

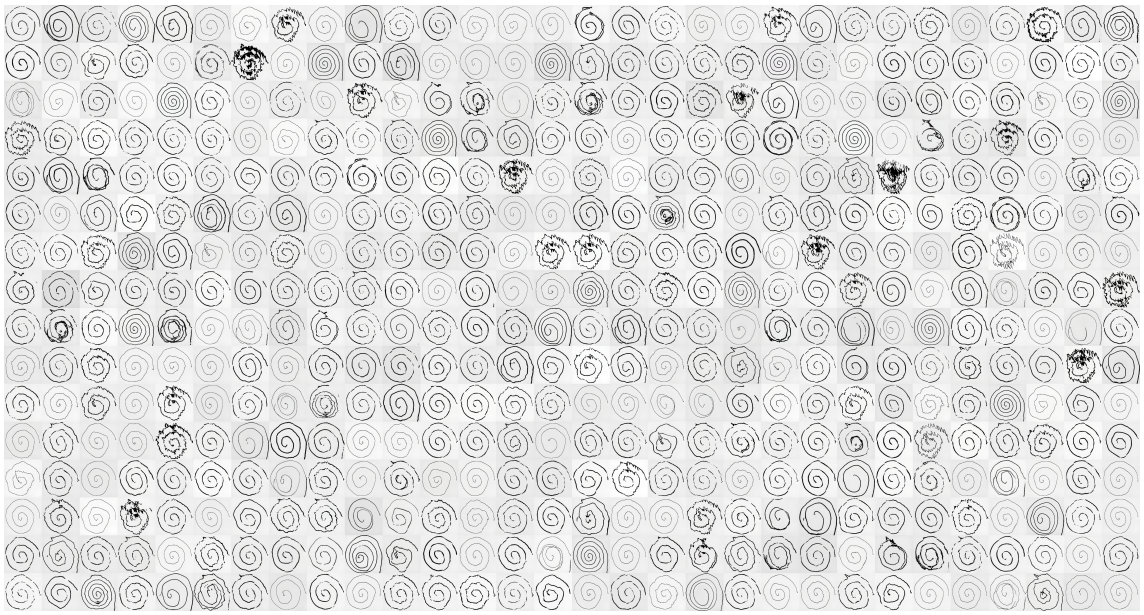


Figure 62. Grid of images from StyleGAN2-ADA PD best checkpoint

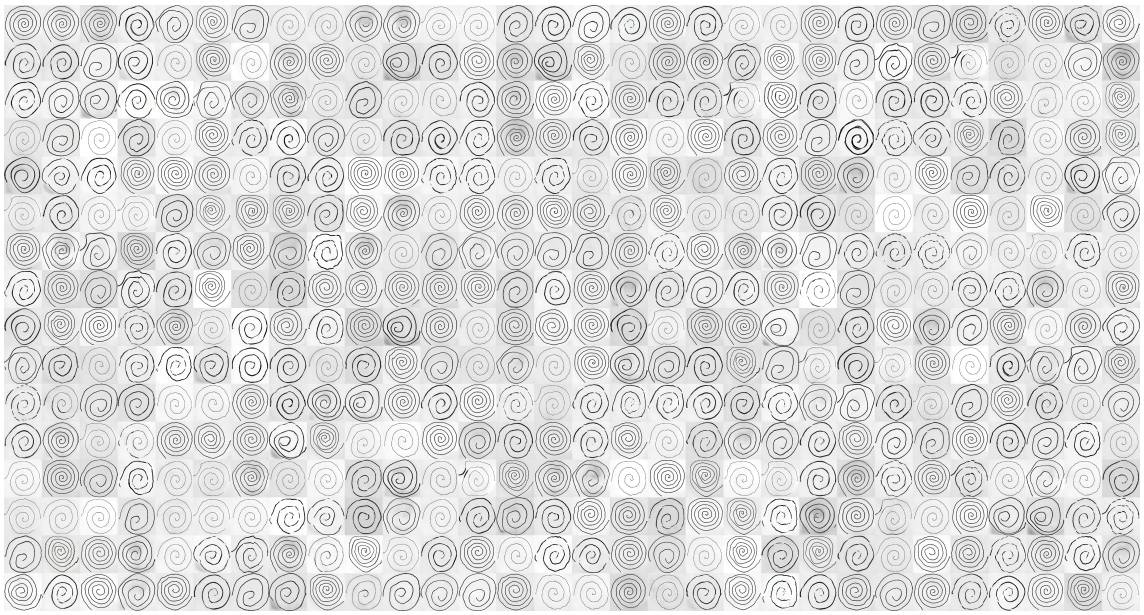


Figure 63. Grid of images from StyleGAN2-ADA + LeCam HC best checkpoint

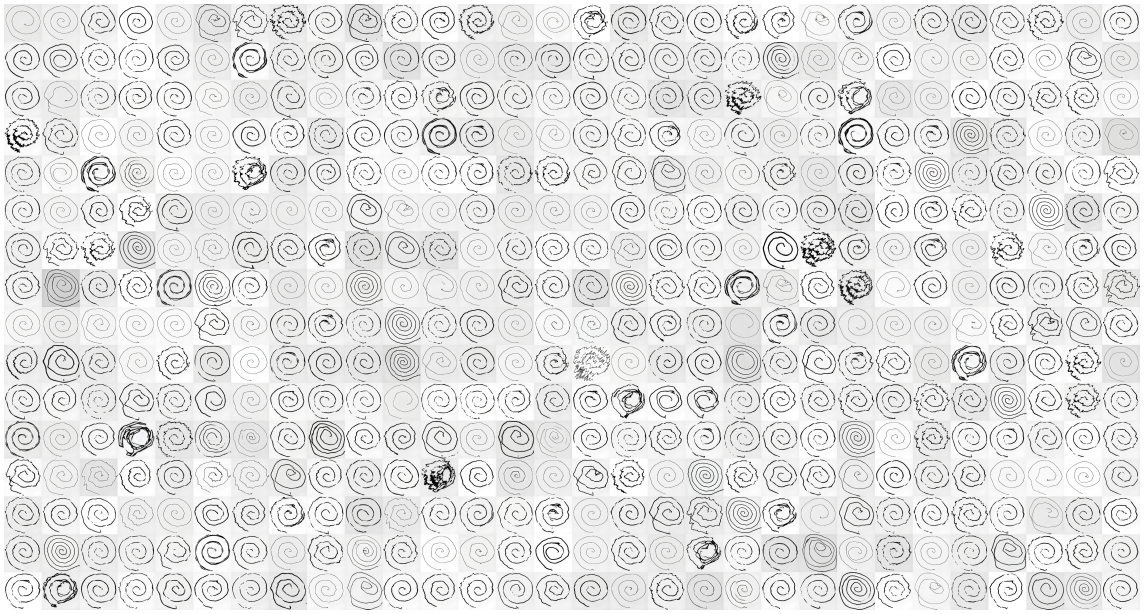


Figure 64. Grid of images from StyleGAN2-ADA + LeCam PD best checkpoint

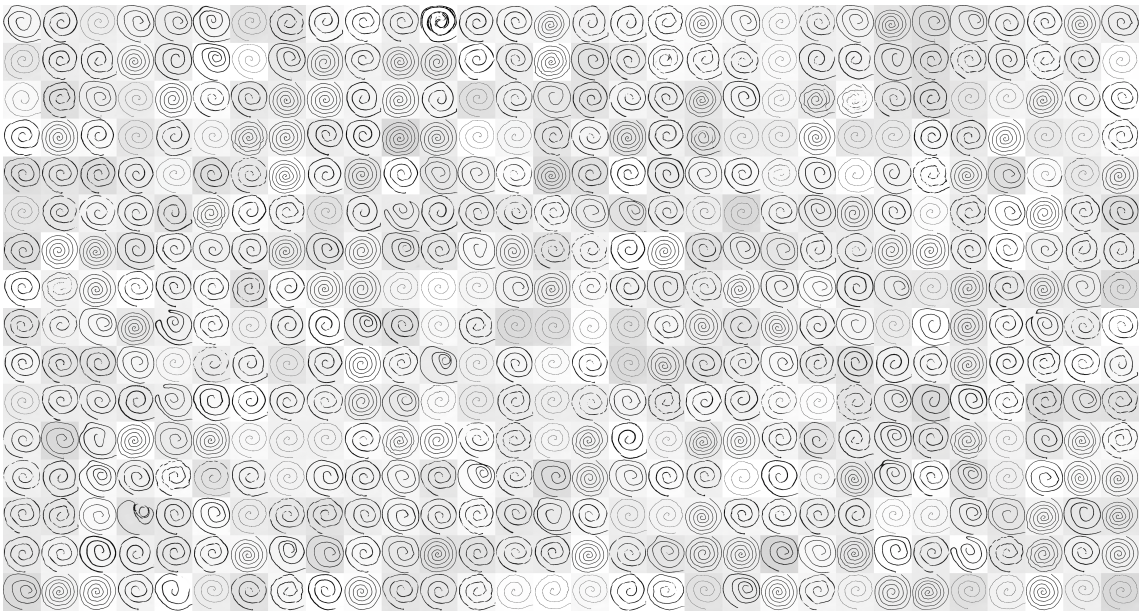


Figure 65. Grid of images from StyleGAN3 HC best checkpoint

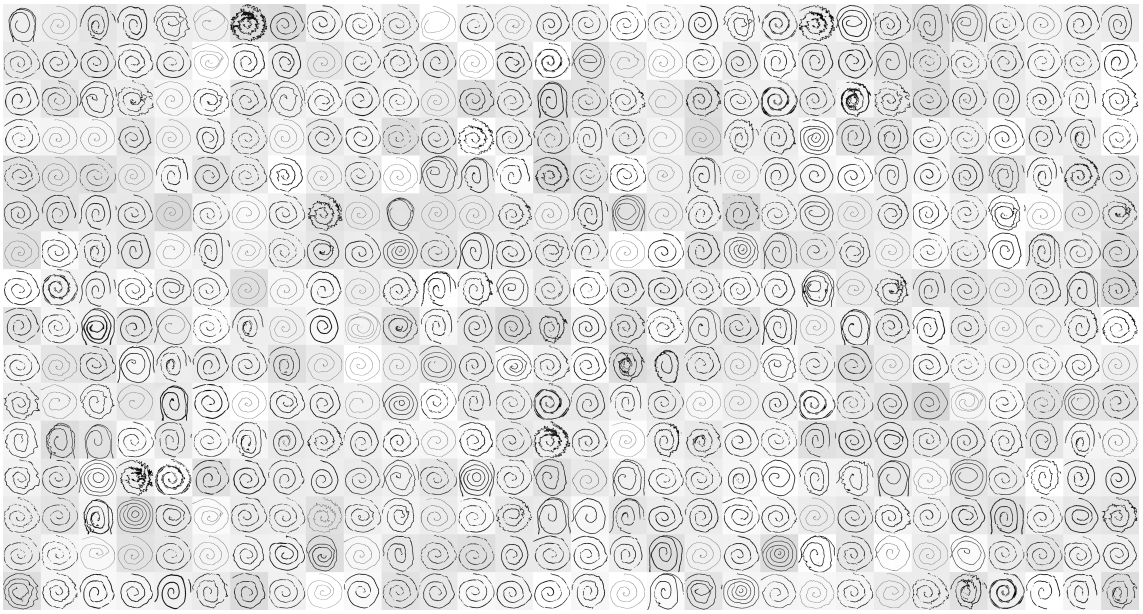


Figure 66. Grid of images from StyleGAN3 PD best checkpoint

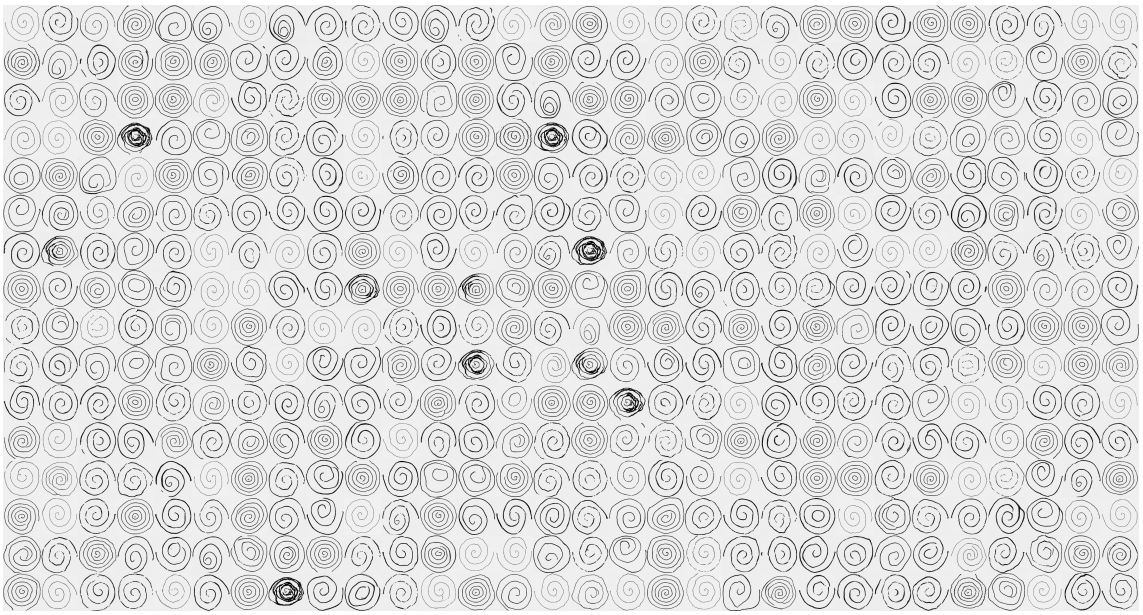


Figure 67. Grid of images from Projected GAN HC best checkpoint

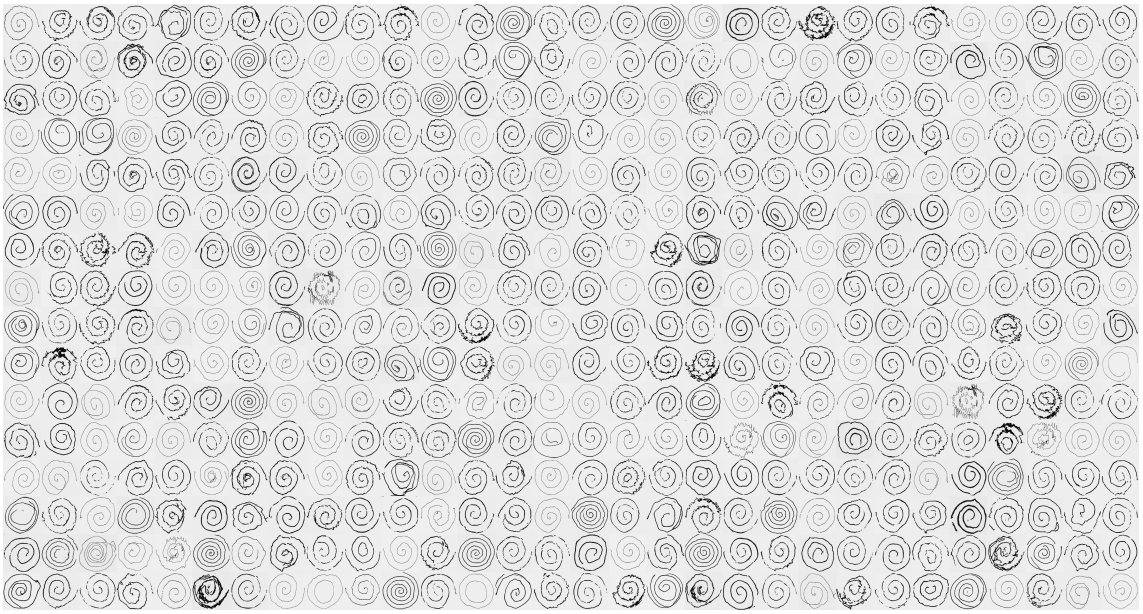


Figure 68. Grid of images from Projected GAN PD best checkpoint