



CHALMERS
UNIVERSITY OF TECHNOLOGY



UNIVERSITY OF GOTHENBURG

Are these numbers real?

Using GANs for improving quality control in smart manufacturing

Master's thesis in Computer science and engineering

KARL GRIPHAMMAR

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2022

MASTER'S THESIS 2022

Are these numbers real?

Using GANs for improving quality control in smart manufacturing

KARL GRIPHAMMAR



UNIVERSITY OF
GOTHENBURG



CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2022

Are these numbers real?
Using GANs for improving quality control in smart manufacturing
KARL GRIPHAMMAR

© KARL GRIPHAMMAR, 2022.

Supervisor: Yinan Yu, Department of Computer Science and Engineering
Advisor: Samuel Scheidegger, Aixia AB
Examiner: Carl-Johan Seger, Department of Computer Science and Engineering

Master's Thesis 2022
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in L^AT_EX
Gothenburg, Sweden 2022

Are these numbers real?

Using GANs for improving quality control in smart manufacturing

KARL GRIPHAMMAR

Department of Computer Science and Engineering

Chalmers University of Technology and University of Gothenburg

Abstract

Smart manufacturing refers to the use of digitalization for improving and automating manufacturing processes. One use case is artificial intelligence (AI) used in quality control, which can reduce production costs and heavy labor. Training AI models requires large amounts of annotated data, which can be costly to obtain. This study aims to examine whether generative adversarial networks (GANs) can be used for improving an image classification model, which is commonly used in smart manufacturing. The data used in the study consists of cropped photographs of characters from serial numbers on automotive engine parts on a production line, which can be used to link the parts to certificates used for quality control. The GANs are trained on the real images. The generated images are then sampled to a mixed set of synthetic and real images, on which a convolutional neural network (CNN) is trained. In this study, we sample two small subsets of the total dataset, and investigate how the size of the dataset affects the performance. Further, we also show that this data can be well-represented by a low-dimensional subspace. This property is used for developing specific methods for sampling synthetic data. The study finds that using GANs for augmenting datasets can increase the performance of the CNN significantly, even when the original dataset is small. Using sampling methods based on subspaces is shown to have a positive effect when the number of added synthetic samples is low, but random sampling yields a higher performance otherwise.

Keywords: Machine learning, data science, deep generative models, GAN, quality control, smart manufacturing, subspaces.

Acknowledgements

I would like to express my deep gratitude to Yinan Yu and Samuel Scheidegger for providing excellent supervision and help during this thesis. I have learned a lot from our discussions: from subspaces and applied research, to how beer recipes can be artificially generated. A special thanks also to Carl-Johan Seger for providing guidance and support during the project.

This thesis would not have been possible without the constant support from my beloved wife Nina. Finally, many thanks to my two children, who apparently decided together that the thesis itself was not challenging enough.

Karl Griphammar, Gothenburg, June 2022

Contents

List of Figures	xi
List of Tables	xiii
1 Introduction	1
1.1 Problem	2
1.2 Goals	3
1.3 Previous studies	4
2 Theory	5
2.1 Deep learning	5
2.1.1 Convolutional neural networks	6
2.2 Generative Adversarial Networks	7
2.2.1 The objective function	9
2.2.2 The architecture	11
2.2.3 Evaluation	14
3 Methods	17
3.1 Dataset	17
3.1.1 Preprocessing	18
3.2 Setup of experiments	20
3.2.1 Principal Component Analysis	24
3.3 Architecture	29
3.4 Evaluation	31
3.4.1 Test data	32
3.5 Comparing architectures	32
4 Results	35
4.1 Initial evaluation of the GANs	35
4.2 Experiments	38
4.2.1 Random sampling of synthetic data	38
4.2.2 Subspace sampling	43
5 Conclusion	49
5.1 Future work	50
Bibliography	53

List of Figures

2.1	A simplified overview of a neural network	6
2.2	An overview of a convolutional layer. The figure shows the first convolution the filter does in the top-left part of the input. In the next step, the filter will slide one stride to the left in the input, and will output a different value in the feature map.	7
2.3	A schematic overview of the GAN	7
2.4	A schematic overview of the CGAN	9
2.5	Schematic comparison of the discriminator in the ACGAN, ProjGAN, and ReACGAN. How the discriminator is conditioned on the class information constitutes the main differences in architecture between the three GANs	13
3.1	Example of serial number from machine part.	18
3.2	Random samples from real dataset. Each row contains four random images from each class.	19
3.3	Distribution of sample sizes for each image class in the real dataset (N = 110 660)	20
3.4	Plot of image width and height of each image per class in the real dataset (N = 110 660). The red line shows an estimated regression line, showing the estimated height as a function of width.	21
3.5	Mean and standard deviation of pixel values for each image and class in the real dataset (N = 110 660)	22
3.6	Flowchart describing the experiment process. The first decision is the size of the set of real images. The GAN is trained on this dataset, and a sample of the synthetic data is mixed with the real data to form a mixed training set. This is used to train a control network, which is then evaluated on the test set.	23
3.7	Histogram of L2 norms of projections, using 682 principal components. 26	
3.8	Random images from below the 10 th percentile of L2 norms using 682 principal components.	27
3.9	Random images from above the 90 th percentile of L2 norms using 682 principal components.	28
3.10	Schematic overview of the discriminator	29
3.11	Schematic overview of the generator	30

3.12	Random samples from the trained GANs. Each row shows eight samples from the GANs in the following order (from top to bottom): ACGAN, BigGAN, BigGAN (diffaug), ReACGAN, ReACGAN (diffaug)	34
4.1	Random samples of generated data from ACGAN, trained on 2200 samples.	36
4.2	Random samples of generated data from ACGAN, trained on 11 000 samples.	36
4.3	Random samples of generated data from ReACGAN, trained on 2200 samples.	37
4.4	Random samples of generated data from ReACGAN, trained on 11 000 samples.	37
4.5	Performance of classifier network when adding randomly sampled images generated by the ACGAN. Size of real training set: 11 000. The classifier network is evaluated on the test set (11 000 test images). The dashed line shows the performance of the classifier network when using only real data. Number of added samples are expressed in logarithms with base 10.	39
4.6	Performance of classifier network when adding randomly sampled images generated by the ReACGAN. Size of real training set: 11 000. The classifier network is evaluated on the test set (11 000 test images). The dashed line shows the performance of the classifier network when using only real data. Number of added samples are expressed in logarithms with base 10.	40
4.7	Performance of classifier network when adding randomly sampled images generated by the ACGAN. Size of real training set: 2 200. The classifier network is evaluated on the test set (11 000 test images). The dashed line shows the performance of the classifier network when using only real data.	41
4.8	Performance of classifier network when adding randomly sampled images generated by the ReACGAN. Size of real training set: 2 200. The classifier network is evaluated on the test set (11 000 test images). The dashed line shows the performance of the classifier network when using only real data.	42
4.9	Accuracies from augmented training set using images generated by the ACGAN. 11 000 real images were used.	44
4.10	Accuracies from augmented training set using images generated by the ReACGAN. 11 000 real images were used.	44
4.11	Accuracies from augmented training set using images generated by the ACGAN. 2 200 real images were used.	46
4.12	Accuracies from augmented training set using images generated by the ReACGAN. 2 200 real images were used.	46

List of Tables

3.1	The sum of explained variance per class and number of principal components (in percent).	25
3.2	Number of total parameters in each respective network. The backbone in all GANs are very similar, with the exception being the ReACGAN which includes projections and embeddings of class information, and hence more parameters than the other GANs.	31
3.3	Evaluation scores from the initial comparison of GANs	32
4.1	Overview of the different datasets used in the experiments. The smaller training sets are randomly sampled from the full training set. This is different from the full dataset described in the method chapter, since this includes both the full training set and the test set.	35
4.2	Evaluation scores of the ACGAN and the ReACGAN trained on 1000 and 200 real images per class respectively. GAN-train scores when using only real data are included as a benchmark.	38
4.3	Accuracies of classification network for all experiments, expressed in percentages. Each row shows the accuracies when adding the number of synthetic samples to the real images in the training set shown in the second column. Each column shows the accuracies when using the number of real images in the training set shown in the second row. The highest accuracy for each GAN and each number of real images used is bolded.	47

1

Introduction

The past two decades have been characterized by a rapid development of advanced digitalization and Internet technologies, and the implementation of such technologies in industrial production is sometimes called the fourth industrial revolution, or Industry 4.0 [1]. Industry 4.0 can be characterized by two major development directions: an application-pull and a technology-push in industrial practice. The past decades have seen changes such as decentralization of markets and production, shorter development periods, and in general an increased demand for more efficient and flexible production. At the same time, technology development has led to increased automation and digitalization, and has made production and logistics more efficient in terms of physical space due to smaller computers.

The new production technologies lead to a constantly growing mass of produced data, which in turn has motivated the use of machine learning methods for addressing challenges in industrial production and logistics [2]. Examples of use-cases of machine learning and artificial intelligence in smart manufacturing processes today are real-time data processing and analysis, fault detection, and predictive maintenance [3]. Identifying mechanical faults is a crucial part of the manufacturing process. Problems can occur when machines used in production starts to wear down, or if produced parts are defect in some way. Failing to identify such problems may lead to large consequences for the safety and for the profitability of the production. Hence, it is of interest of the manufacturing company to identify mechanical faults in a reliable way. Since the work is monotonous, it is desirable to automate this process. In the past decade, deep learning models have been a popular approach to this task [4]. These models have often performed well due to their feature learning capacity and non-linear mapping ability. Convolutional neural networks (CNNs) have been especially popular, e.g., by setting up cameras for monitoring the production, and to let the network determine whether the produced parts are acceptable or not. One approach to computer vision for quality control, which has been more common in recent times, is to use generative adversarial networks (GANs) for data augmentation [5]. Since GANs were introduced, there have been many developments that have improved its use for data augmentation tasks. Some models try to reduce the need for data when training the GAN [6, 7], while other implementations try to incorporate more information in the GAN to better control the generation [8, 9].

1.1 Problem

The background of this project is the task of automating quality control of industrial manufacturing, and introduce traceability of mechanical parts. The manufacturing company in question produces automotive engine parts, using a production line with manual quality assurance. The purpose of this process is to assure that the mechanical parts are not defect in any way, and each part must be controlled for physical damages or other defects. However, manual quality assurance is time-consuming, expensive, and potentially harmful for the worker due to heavy lifts. Further, the earlier the quality assurance can be implemented in the manufacturing process, the more cost-reducing it will be. To make this process more efficient, a camera has been set up on the production line. The camera takes a photo of the serial number on each machine part, and each character in the serial number is identified using an object detection network.

Object detectors using convolutional neural networks are generally grouped into two types: *one-stage detectors* and *two-stage detectors*. In two-stage detectors, the object detection is treated as a classification problem. In the initial stage, a number of bounding boxes are sampled from the image. This can be done with, for example, region proposal methods. In the second stage, an image classifier is applied to predict objects in the bounding boxes, or if there is only background in the box. Only non-background boxes are treated as detections in the end. An example of this approach is the R-CNN [10].

One-stage detectors, such as YOLO [11], are a type of methods where the object detection is done in a single feedforward pass. Instead of treating each proposed region independently, all predictions in an image are done all at once by treating it as a regression problem. The input image is divided into grids, which each contains a number of base boxes. In each grid, the model will predict the actual bounding box as well as the most likely object to appear in that bounding box, given a set of classes. This computation is done in a single CNN, instead of splitting up the tasks as in the two-stage detectors.

In general, one-stage detectors are faster than two-stage detectors which are often more computationally expensive, but two-stage detectors are often more accurate. Since accuracy is an important quality in smart manufacturing, the object detection approach used in the setting of this project is a two-stage detector. Specifically, the image classifier will be of special interest for this project (the classifier will henceforth be referred to as the *control network*).

When a serial number is detected and classified, the machine part can be linked to a specific certificate via the serial number. The certificate stores relevant data, such as which machine the specific part was made from, what type of compound the part consists of, etc. This setup allows for a more efficient fault control, since it is fully automated in practice, and any defect that is observed can be linked to all other machine parts from the same batch. This also allows the fault monitoring to be done earlier in the process, which is beneficial from a logistics point of view. Hence, this use of machine learning for fault control is more cost-effective, less time-consuming, and ergonomically better for workers, compared to manual quality control.

Classifying mechanically printed digits and letters can, however, be error-prone. The

needle that prints the digits and letters will eventually be worn out, which causes the printings to be less clear, and photographs of the machine parts might be over- or underexposed. Errors in this process can be expensive, since machine parts might be linked to the wrong certificate, which in turn can lead to costly errors downstream in the process.

The overall aim of this project is to improve the performance of the control network, and minimize the number of misclassifications. The performance of the control network can be improved by better training. For example, images that might be hard to label correctly can be identified in order to specifically train the control network on these hard cases. However, industrial data can be expensive to annotate, and the difficult cases are especially hard to collect. For this reason, models tend to suffer from overfitting and do not perform as well on unseen test data. A common problem when using deep learning models in industrial quality control is that the models tend to mostly work on the training data only [12]. Traditional data augmentation, such as adjusting brightness or rotating angles, tend to only have a very small effect on performance [6].

1.2 Goals

This project aims to use GANs to generate synthetic data, with the purpose of improving the training of the control network. Since data can be difficult and expensive to obtain, the solution has to achieve both high performance and efficiency. The goal is to examine how synthetic data can be used to achieve high accuracy of a classification network, while only using a limited set of real images. It will be shown in this thesis that the data can be described as being well-represented in a low-dimensional subspace, and we will attempt to develop a method that ideally can be generalized to other datasets with similar properties.

The questions that this project attempts to address can be defined as follows:

- Can a GAN be used for improving the performance of the control network tasked with classifying digits and letters on machine parts? While previous studies have shown that GANs can be used for data augmentation in classification tasks within industrial settings [13, 14, 15], we specifically aim to develop a technique that is suited for data that can be described as being well-represented in a low-dimensional subspace.
- How does the size of the dataset of real images affect the performance of the control network? While the ideal is to use a large dataset, real data is expensive to collect. Hence, we want to examine if only a small set of real data can be used while still maintaining high accuracy.
- Can the performance of the control network be improved by using different sampling techniques? Specifically, can images be sampled, that are more difficult for the control network to classify?

1.3 Previous studies

Deep learning has been a common method for implementing artificial intelligence in smart manufacturing processes. Some application areas of deep learning have been descriptive analytics for quality inspection, fault assessments, and predictive models for defect prognosis [16]. Examples of these applications are neural networks for surface defect detection on different textures [17], recurrent neural networks (RNNs) used for long-term prognosis of machine health status [18], and using long short-term memory networks (LSTMs) for predicting remaining useful life of machines [19]. Specifically, in quality control, neural networks for image classification is a common method for identifying presence or absence of faults. One example is El Hachem et al. [20], who used a CNN for classifying whether images of motor parts contained the right screws or if they were missing. They showed that this deep learning model achieved a high accuracy, and performed better than humans. Hence, applications of deep learning models like the studies mentioned suggest that high performance of quality control can be achieved, while decreasing cost and encumbering labor for workers.

There are many variations of applications using GANs for augmenting datasets by adding synthetic data. GANs have been used for augmenting medical data, such as images of brain scans [21, 22] or x-ray images of lungs [23]. Similar studies have been done in industrial settings. Shao, Wang Yan [24] used a conditional GAN for generating images to augment an unbalanced dataset, where the task was to classify mechanical sensor signals. They showed that adding synthetic samples could help to balance the dataset, and that the performance of the classification model increased by adding more generated data to the training set. Niu et al. [25] suggested a new generation method for generating defect images using defect-free images of mechanical parts. The goal was to improve a neural network tasked with anomaly classification, and their suggested method was a GAN used for generating specific defects on real defect-free images. Their results suggested, in line with previously discussed studies, that using GANs for augmenting small training sets improves the performance and reduces the error rate of the classification network.

A study with a similar setting to this project was Pinetz, Ruisz, Soukup [26] who used a Wasserstein GAN (WGAN) for generating digits when trained on banknote serial numbers. They specifically studied the effect and stability of training a GAN with a limited dataset, and showed that GANs trained on very small datasets can improve the accuracy of a classifier network. They also showed that training GANs on both real and previously generated synthetic data improved the performance even further.

A common theme for the studies mentioned in this section is that while they all suggest that GANs are effective methods for data augmentation, there is no standard method and the implementation tends to be data-driven. Data augmentation using deep generative models can thus be considered an open research question. What this study adds is also whether there are sampling techniques especially effective for data represented by low-dimensional subspaces. No study have been found that has brought this specific question into the light.

2

Theory

This chapter provides an overview of the theory behind GANs and its developments since it was introduced, as well as a brief overview of evaluation methods for GANs.

2.1 Deep learning

Deep learning is a field of machine learning that deals with artificial neural networks with representation learning [27]. The idea is that a neural network takes some input and learns a more abstract representation consisting of a series of mappings, described by different layers in the network. The *deep* in deep learning refers to that the data is transformed through a number of so-called *hidden layers*, that each learns increasingly abstract features from the input data. A schematic overview of this can be seen in Figure ??

The concept of artificial neural networks is often said to be analogous with biological neural networks, where neurons receive inputs from its dendrites, and transmits output signals along its axons. In artificial neural networks, the hidden layers consist of a number of neurons that in its basic structure takes an input, which is then multiplied by its corresponding weight, and then passed through an activation function. This can be formalized as:

$$a_j^l = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right) \quad (2.1)$$

where a_j^l is the activation of the j^{th} neuron in the l^{th} layer. In a typical feedforward neural network, the input is passed through all hidden layers to the output layer. The output layer does not often have an activation function. Instead, the output often represents the class score, e.g. in the case of a classification-task.

The neural network learns the mapping from input x to output y by finding the parameters θ (which in the previous example are w and b) that do this best. This is done by defining a loss function, e.g.:

$$J(\theta) = \frac{1}{2n} \sum_x (y - \hat{y})^2 \quad (2.2)$$

This is an optimization problem, since the goal is to minimize the loss. The standard algorithm used for training neural networks is called *backpropagation*, which computes the gradient of the loss function with respect to the weights of the network.

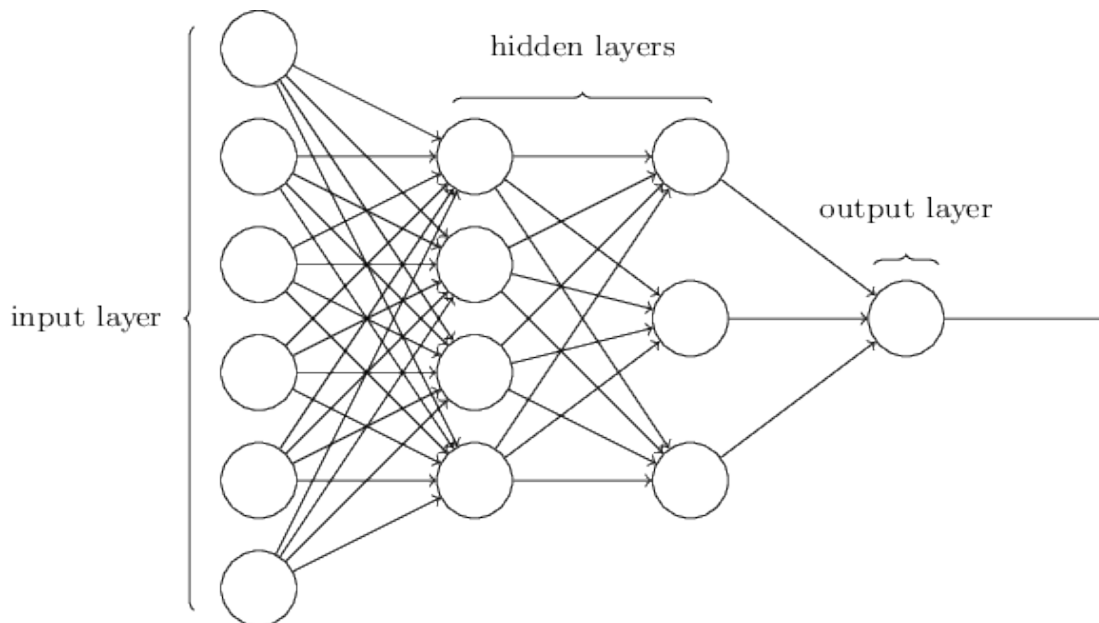


Figure 2.1: A simplified overview of a neural network

2.1.1 Convolutional neural networks

CNNs are special types of neural networks for analyzing data with a grid-like topology, such as images or time-series data. The reason for this is that spatial relations are taken into account when data is being processed in the network [27]. What characterizes CNNs are *convolutions*, which are special types of linear operations. A convolutional layer in the CNN takes an input and uses a filter for performing convolution operations. The filter slides over the input across the spatial dimensionality, and a scalar product is calculated from the filter kernel and the input. The output of this operation is called the *feature map* or *activation map*. This process is illustrated in Figure 2.2.

Another important part of the CNN is the pooling layers, which function as dimensionality reduction. Similar to the convolutional layers, the pooling layer slides a filter over the feature map. But instead of performing computations using weights similar to the filter kernel, the pooling layer aggregates the values of the feature map. Two common types of pooling are *max pooling* and *average pooling*. A max pooling layer takes the max value from each slide of the filter over the feature map, while average pooling computes the average. The purpose of this operation is to reduce complexity and mitigate problems with overfitting.

Finally, the fully-connected layer is a layer in the neural network where each neuron is connected to all neurons in the previous layer. This layer computes class scores, which, for example, can be used for image classification.

CNNs have been particularly successful in practical applications, and several CNN architectures have been given a name due to how well they have performed in their respective task. Some of the most well-known are: LeNet [28], VGGNet [29], and ResNet [30].

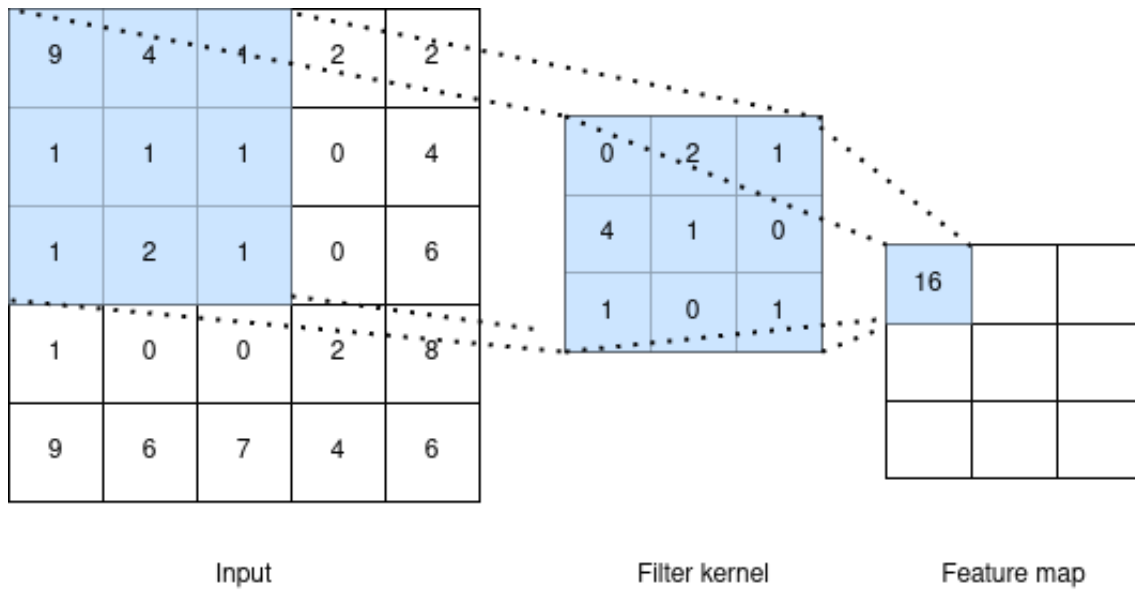


Figure 2.2: An overview of a convolutional layer. The figure shows the first convolution the filter does in the top-left part of the input. In the next step, the filter will slide one stride to the left in the input, and will output a different value in the feature map.

2.2 Generative Adversarial Networks

The GAN was introduced in 2014 to improve the performance of current deep generative models [5]. In the GAN, a generative model (denoted G) is trained against a discriminative model (denoted D), using an adversarial process. The goal of the generative model is to generate synthetic data as similar to the real training data as possible, while the goal of the discriminative model is to determine whether the data is real or if it is synthetic. The end goal of the network is to generate data such that the discriminative model is unable to distinguish it from real data.

When training a GAN, the two models G and D are trained simultaneously based

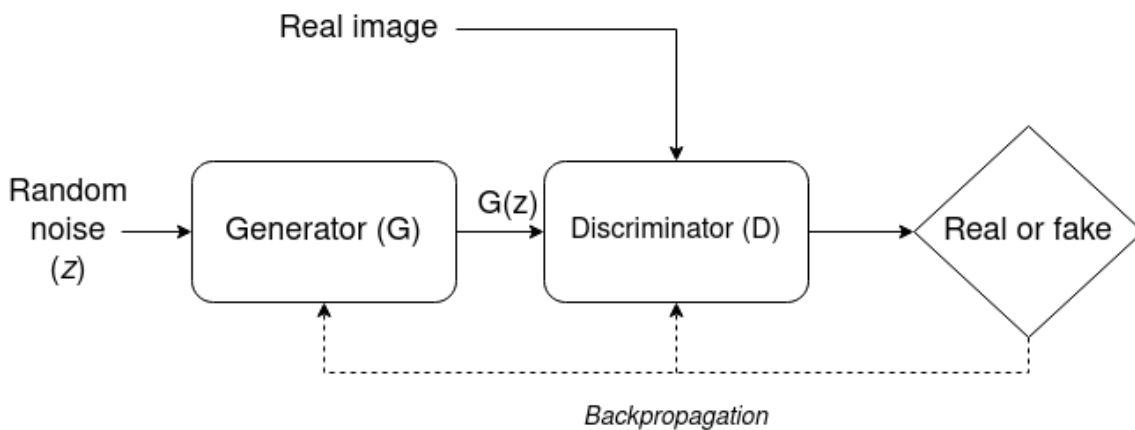


Figure 2.3: A schematic overview of the GAN

on a minimax game, meaning that one agent is trying to minimize the value function of the GAN, while the other tries to maximize it. The generator learns the mapping of a noise vector $z \sim p_z(z)$ to data space as $G(z; \theta_g)$. The discriminator $D(x; \theta_d)$ takes the data x as input and determines whether x comes from the real data p_r or the generative distribution p_g . A schematic overview of this process can be seen in Figure 2.3.

D is trained to maximize the probability of correctly classifying the label of both the real training data and the output from the generator, while G is trained to minimize $\log(1 - D(G(z)))$. The minimax game with value function $V(G, D)$ can then be stated as:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[1 - \log D(G(z))] \quad (2.3)$$

Goodfellow et al. [5] showed that the minimax game described above has a theoretical global optimum for $p_g = p_r$. This means that if the training is successful, G and D will eventually reach a point where they cannot improve anymore. At this optimum, the discriminator is unable to differentiate between the real data and the synthetic data from the generator, since the two distributions are equal.

Although adding classes as input to both the generator and the discriminator was mentioned in the original paper, the GAN was initially unconditioned and there were hence no proper way of controlling the output of the generated data. To remedy this shortcoming, the conditional Generative Adversarial Net (CGAN) was introduced, in which it was possible to condition the model with additional information and thereby steer the data generation process [31]. In the CGAN, the generator and discriminator are conditioned on auxiliary information, such as class labels. A schematic overview can be seen in Figure 2.4. The objective function for the CGAN can be stated as:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z \sim p_z(z)}[1 - \log D(G(z|y))] \quad (2.4)$$

This objective function is similar to equation 2.3, but the input is now conditioned on the extra information y . The CGAN allows the network to be used for supervised tasks, and has since its introduction been a standard implementation in later conditional GANs.

Along with the conditionality of the CGAN, the use of convolutional layers in the GAN is now a standard implementation, introduced with deep convolutional generative adversarial networks (DCGAN) [32]. The architecture used in the original GAN implementation had been a combination of ReLU and sigmoid activation functions in the generator, and maxout activations in the discriminator [5, 31]. The objective function was the same as in the GAN implementation, but the architecture introduced by DCGAN consisted of only convolutional layers in both the generator and the discriminator. This allowed the network to be deeper, and hence improved the performance of GANs and made the training more stable. Using convolutional layers also resulted in the generator learning semantically meaningful features from the images. This allowed some control over the generation of synthetic data, either by manipulating weights in specific convolutional layers, or by vector arithmetic of the noise vector used in the generator [32]. It was, for example, to some extent possible to control whether generated images of rooms would depict windows, or whether generated faces would smile or not.

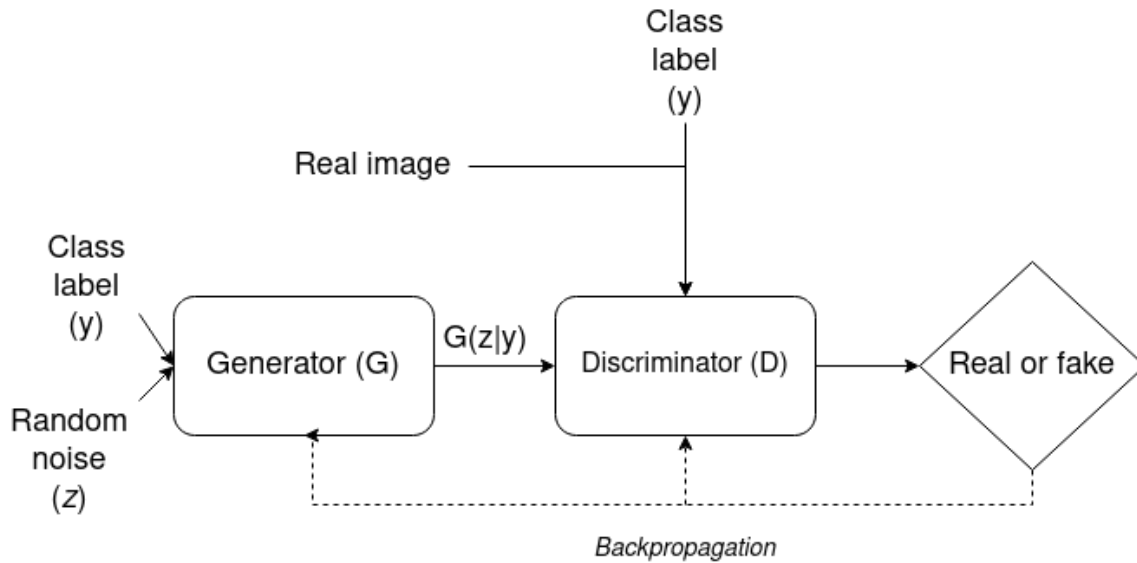


Figure 2.4: A schematic overview of the CGAN

While the CGAN and DCGAN became standard implementation of subsequent GANs, there have been a countless number of improvements and variations of the framework since the GAN was initially introduced. Two major reasons for this are that the design of the two models (G and D) in the GAN has few limitations, and that even though it has been shown that the minimax game has a theoretical global optimum, this has turned out to be difficult in practice for several reasons [33]. The flexibility of the generator in particular has resulted in a widespread use of GANs in many domains. Similarly, the many difficulties of training a GAN have lead to different variations of the framework that attempts to overcome these problems. Hence, two major trends in GAN research are how GANs can be used for real-world applications, and how training GANs can be improved [34].

The following two subsections will provide a short overview of different GANs, which will provide a ground for the choice of GAN used for this thesis. While no formal taxonomy of GANs exist, survey papers [34, 35, 33] tend to divide the existing frameworks into two variants: GANs that focus on the objective function, and GANs that focus on the architecture, or the structure of the network. The overview presented here will adhere to this taxonomy. First, different methods for altering the objective function shown in equation 2.3, or by choosing a specific loss function, will be discussed. The purpose of this is to improve the performance of training the GAN. Second, different frameworks that focus on changing the architecture of the network will be discussed. The purpose of these is often to use for specific applications, but also for improving image quality or stabilize the training.

2.2.1 The objective function

As mentioned, it was pointed out already when the GAN was introduced that although the objective function described in equation 2.3 has a theoretical global optimum for $p_g = p_r$, there are some practical issues with this [5]. In the original GAN, the discriminator was maintained near its optimal state during training, and

the generator slowly converged with each update. Otherwise, if the generator is trained too much relative to the discriminator, there is a risk of collapse and the generator will generate output with little or no variation. However, it turned out that this was rather inherent to the original objective function [36]. When using the objective function described in Equation 2.3, improving the discriminator will eventually lead to vanishing gradients, meaning that the generator will not improve at all, or to inaccurate updates of the discriminator which results in highly unstable training. If D behaves perfectly by classifying all data correctly, the loss will fall to zero and G will receive very little feedback. On the other hand, if D behaves badly by making poor predictions, the feedback provided to G will be very inaccurate and not meaningful. A common failure case as a result of this is *mode collapse*, meaning that the generator keeps producing the same output since it is unable to learn the complexity and variations of the real distribution. Given that these problems are inherent to the objective function, there has been an argument for the need of redesigning the loss function rather than changing the architecture [34].

One solution to the problems with vanishing gradients and the risk of collapse was the WGAN [37]. Two reasons for this are, first, that the WGAN replaces the binary classifier of the discriminator in the original GAN with a critic that maps the prediction to a continuous space. Second, the WGAN uses earth mover’s distance – a measure of distance between two distributions – as the loss measure for optimization. It can be shown that this allows a sequence of probability distributions to converge, even if they do not overlap. The new loss function in the WGAN is K -Lipschitz continuous, meaning that it is everywhere continuous differentiable. Hence, it does not suffer from vanishing gradients, as was one of the problems with the original objective function. The result was a training that was not limited by the performance of the discriminator, and hence provided a more stable training. The WGAN critic was, however, designed to enforce a Lipschitz constraint using weight clipping [37]. As a result, the training became slower, and the WGAN was not suited for very deep networks. To remedy this, the WGAN-GP was suggested, which made use of gradient penalties in the loss function instead of weight clippings [38]. This was shown to stabilize the training further, and allowed for stronger modeling performances.

An alternative solution to the WGAN was to use spectral normalization, as suggested by Miyato et al. [39], instead of weight clipping or gradient penalties. They found that the previous approaches were heavily dependent on the generative distribution. Thus, changes in this distribution can destabilize the effect of regularization methods such as weight clipping or gradient penalty. For example, a high learning rate might have such destabilizing effect. Instead, constraining the spectral norm in each layer of the generator in order to satisfy the Lipschitz constraint would provide a more stable training, since this regularization is data independent [39]. This was shown to generate images of higher quality and more diversity compared to previous regularization methods, when evaluated on the CIFAR-10 and the ImageNet datasets.

2.2.2 The architecture

The following section will present some GANs described in the literature, that focus on improving the architecture of the network. Due to the abundance of different architectures, this will be a non-exhaustive presentation that will focus on the most significant contributions. The focus here will also be on GANs that are in some way conditional on class information, since these have been shown to achieve better results and are more relevant for this project. While conditional GANs share some similarities, there are significant differences in how the conditioning is done in the network.

As described previously, the idea of conditioning the GAN on class information was initially presented in the CGAN [31], hence allowing supervised data generation. This was done by concatenating the class information y to the input of the generator and discriminator. To improve the quality of image generation of the CGAN, the auxiliary classifier GAN (ACGAN) was introduced [40]. In contrast to the CGAN, where class information was fed to the discriminator, the ACGAN proposed to instead reconstruct the class label by adding an auxiliary decoder network that outputs the class label. The argument was that providing additional tasks to a model would improve the performance on the original task. The discriminator would now output two probability distributions: over sources and over the class labels, $P(S|X), P(C|X) = D(X)$. This would rearrange the objective function from the CGAN as following:

$$L_S = \mathbb{E}[\log P(S = \textit{real}|X_{\textit{real}})] + \mathbb{E}[\log P(S = \textit{fake}|X_{\textit{fake}})] \quad (2.5)$$

$$L_C = \mathbb{E}[\log P(C = c|X_{\textit{real}})] + \mathbb{E}[\log P(C = c|X_{\textit{fake}})] \quad (2.6)$$

Where L_S is the log-likelihood of the correct source, and L_C is the log-likelihood of the correct class. With this objective function, D is trained to maximize $L_C + L_S$, while G is trained to maximize $L_C - L_S$.

The ACGAN was later shown to suffer from diversity problems of the synthetic data as the number of classes increases [40, 41], meaning that there would be less variation between synthetic images within a given class. Two models that were a direct development of the ACGAN are the twin auxiliary classifiers GAN (TAC-GAN) [41], and the auxiliary discriminative classifiers GAN (ADC-GAN) [42]. The TAC-GAN identified that the original AC-GAN was unable to handle class distributions that overlap, which led to data generation where class label Y had a deterministic relationship to X . In other words, there would be little variation in the intra-class image generation. The proposed solution was to add a second auxiliary classifier that would attempt to determine the class label of the synthetic data. The generator would then compete with this second classifier, resulting in more variation in the synthetic data, since this would be harder for the classifier to label [41]. The ADC-GAN in turn identified that the TAC-GAN turned out to be unstable. Instead of the twin auxiliary discriminator, the ADC-GAN proposed that a discriminatory auxiliary classifier. Instead of letting the auxiliary classifier only recreate the class label as in the original AC-GAN, the proposed classifier would also determine whether

the data was real or synthetic [42]. The argument was that this would more accurately learn the real joint data-label distribution, hence increasing both fidelity and diversity of the generated samples.

The frameworks developed from the CGAN and ACGAN added the embedded class information by concatenating it to the input in both the generator and discriminator. Miyato & Koyama argued that this was a rather arbitrary way of incorporating conditional information, and given that the generator is dependent on the performance of the discriminator, this affects the performance of the network [43]. As an alternative, they suggested a conditional GAN with a projection-based discriminator (ProjGAN). In this framework, the class information was added as an inner product of the embedded conditioning variable and a feature vector of the input image, rather than a concatenation. This way, the loss function will take the distance between the generative distribution and the target distribution, which was shown to achieve a better performance [42] compared to the more arbitrary concatenation of the CGAN. A schematic comparison between the ACGAN and the ProjGAN can be seen in Figure 2.5.

The Self-Attention Generative Adversarial Networks (SAGAN) later introduced self-attention mechanisms to GANs, and was built upon the projection-based GAN [44]. The self-attention allowed GANs to model long range, multi-level dependencies across image regions, and thus improving with generating fine-detailed samples. GANs are ordinarily composed of a number of convolutional layers, where the convolutions process information in local regions. Instead, SAGAN uses self-attention modules, and in each layer, input $x \in^{R \times C}$ are transformed into two feature spaces used to calculate the attention. The self-attention is used in both the generator and the discriminator. SAGAN was shown to perform better than both ACGAN and the projection-based discriminator when evaluated using conditional image generation on ImageNet [44].

The BigGAN was an extended version of SAGAN, specifically developed for large scale training [45]. The major changes from SAGAN included adding skip connections, which meant that the noise vector z was added to multiple layers in the generator rather than just the initial layer, and truncating the noise vector when generating samples. It was shown that BigGAN improved the performance significantly compared to SAGAN, and especially when generating images of higher resolution.

Kang et al. [46] proposed the Rebooted Auxiliary Classifier GAN (ReACGAN) as a further development of both classification-based GANs, such as the ACGAN, and projection-based GANs. They suggested that the cross-entropy loss used in the ACGAN suffers from poor class predictions in early stages of training, due to unbounded feature input vectors, especially when the number of classes in the dataset increases. This leads to exploding gradients, and makes the ACGAN unable to both discriminate between real and synthetic images, and classifying categories of images. The proposed solution in the ReACGAN was to normalize the feature embeddings onto a unit hypersphere, thus mitigating the exploding gradient problem.

The second addition to the ReACGAN was to expand the cross-entropy loss to a Data-to-Data Cross-Entropy (D2D-CE) loss. Both classification-based GANs and projection-based GANs are regarded to use loss functions that are considering only

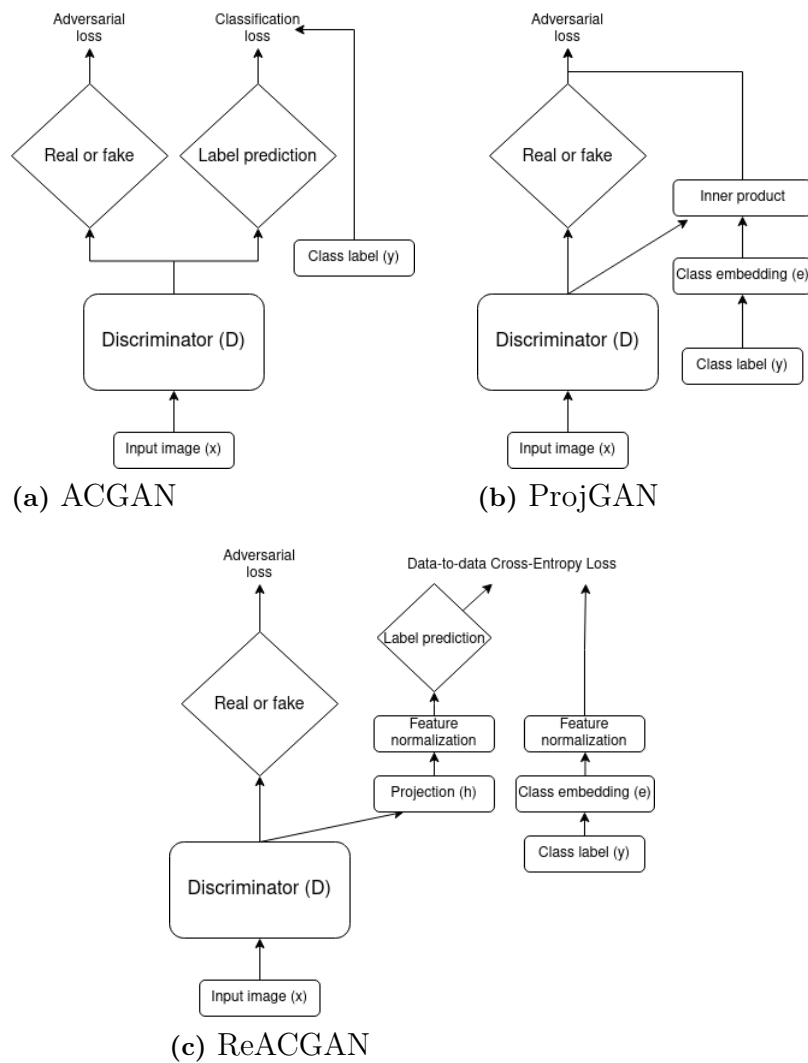


Figure 2.5: Schematic comparison of the discriminator in the ACGAN, ProjGAN, and ReACGAN. How the discriminator is conditioned on the class information constitutes the main differences in architecture between the three GANs

data-to-class relations [47]. That is, both the embeddings and the projections are based on the images and the class labels, hence the respective loss functions are considered to be pair-based. The D2D-CE loss would also take data-to-data relations into considerations, which would increase the intra-class variations [46]. This means that the ReACGAN simultaneously tries to discriminate whether a given image comes from the real distribution or not, and maximize similarities between positive samples (samples within the same mini-batch with same labels) and minimize similarities between negative samples. These two properties of the ReACGAN - the feature normalization and the D2D-CE - were shown to achieve better results when comparing with other GANs and evaluating against different datasets. An overview of the ReACGAN is provided in Figure 2.5.

2.2.3 Evaluation

Evaluating a GAN can in the simplest sense be described as determining the quality of the generated data. Reasons for evaluating GANs might be to compare two or more networks, or to evaluate how well the GAN succeeds in a particular task. In practice, there are many ways to do this. In a survey paper over GAN evaluation measures, Borji [48] listed 29 different common methods, divided into qualitative and quantitative methods. In the following subsection, three different quantitative methods will be discussed, followed by a short mention of qualitative methods.

The two most common quantitative methods for evaluating GANs are the Inception score (IS) [49] and the Fréchet Inception distance (FID) [50]. IS is a measure of how well generated images are classifiable and diverse with respect to their class labels, and has been shown to correlate well to human judgement [49]. When computing IS, an Inception model pre-trained on the ImageNet dataset is used for getting conditional label distributions $p(x|y)$ from the synthetic data to be evaluated. The score is computed as following:

$$IS(G) = \exp\left(\mathbb{E}_{x \sim p_g}[D_{KL}(p(y|x)||p(y))]\right) \quad (2.7)$$

where D_{KL} is the KL-divergence between the conditional class distribution $p(y|x)$ and the marginal class distribution $p(y)$. In meaningful generated images, the conditional class distribution should have low entropy. On the contrary, a model that generates images with high intra-class variation should have a marginal class distribution with high entropy. Hence, a GAN that generates highly classifiable and varied images should receive a high IS. IS does not take real images into consideration, and hence does not measure how similar the synthetic data are to the real data.

FID also uses a pretrained Inception model, but measures instead how well the GAN can approximate the real image distribution [50]. A set of synthetic data and real data are embedded into a feature space by one of the layers in an Inception model. Both the real $p_r(x)$ and the synthetic $p_g(x)$ data will be modeled as Gaussian distributions by the embeddings, with respective means and covariances. FID is computed as the Fréchet distance between the two Gaussian distributions:

$$d^2((m_r, C_r), (m_g, C_g)) = \|m_r - m_g\|^2 + (C_r + C_g - 2(C_r C_g)^{\frac{1}{2}}) \quad (2.8)$$

where $(m_r, C_r), (m_g, C_g)$ denotes the mean and covariance of the real and generated image distributions, respectively. If the synthetic data is very similar to the real data, this distance will be small. Hence, a low FID score is desirable, representing high image quality.

While both IS and FID have been shown to correlate with human judgment, and are inversely correlated with each other, there are some disadvantage to these methods for evaluating GANs. One problem is that both IS and FID relies on an Inception model pretrained on ImageNet, which can be disadvantageous for other datasets [51].

An alternative to IS and FID is to compute the distance between synthetic and real data in terms of precision and recall, represented as GAN-train and GAN-test scores [51]. These methods both require a neural network for image classification. When computing GAN-train, the neural network is trained on synthetic data generated by a GAN, and is then evaluated on a set of images from the real dataset. This would be similar to a recall measure, in the sense that a good performance would show that the generated images are diverse. Conversely, GAN-test is computed by training the neural network on real data, which is then evaluated on the synthetic data. This would be similar to a precision measure, since a good performance would indicate that the synthetic distribution is a good approximation of the real image distribution.

Finally, it is worth mentioning that many other evaluation methods exist, and some are more specific than others. While it is beyond the scope of this research overview, some examples of this are methods for detecting overfitting in GANs (such as nearest neighbors) or methods for detecting mode collapse in GANs [48].

3

Methods

This chapter will provide an overview and analysis of the data used in this project, as well as a description of the experiments to be conducted. All models used in this project are implemented using PyTorch [52]. The codebase of the GAN implementation comes from StudioGAN [53] with adjustments.

3.1 Dataset

The dataset is provided by Sansera via Aixia, and consists of cropped images from serial numbers and batch numbers printed on mechanical machine parts. The serial numbers are photographed with an online setup, using cameras attached to a robot arm. When a machine part is in a specific place on a production line, the robot arm triggers the camera, which activates a flash and photographs the machine part. The photographs are then transferred to a cloud storage for further processing. An example of a photograph of a machine part is provided in Figure 3.1. This figure also shows the next step of the data processing, where an object detector is used for identifying digits and letters in the serial and batch numbers. A bounding box is drawn around each character in the serial number and batch number, and the bounded part of the image is then given a predicted label. The serial- and batch numbers are then used to link each machine part to specific certificates, that can be used for quality control and traceability.

The data used in this project consists of the cropped parts in the bounding boxes, exemplified in Figure 3.1. This makes the problem simpler, since the GANs otherwise would have to generate whole serial numbers. By cropping each digit, the problem becomes more restricted and leaves less room for errors. The varying conditions in the photographs, e.g. shifting lightning conditions or worn down printing equipment that makes digits blurry, make it more difficult for the object detection network to correctly label each digit and letter. The dataset contains 110 660 images from 11 different classes, representing the different characters in the serial- and batch numbers. The characters that appear in these numbers are the digits 0-9 and the letter *W*. Random samples of images depicting each character are shown in Figure 3.2.

The distribution of images per class is shown in Fig 3.3. Because of the higher relative frequency of some classes, there is an imbalance in the dataset. The effect of an imbalanced training set when training a GAN has been shown in previous studies to be negative [54, 55]. These studies suggested that the generated data from minority classes were often influenced by the majority classes, and hence of

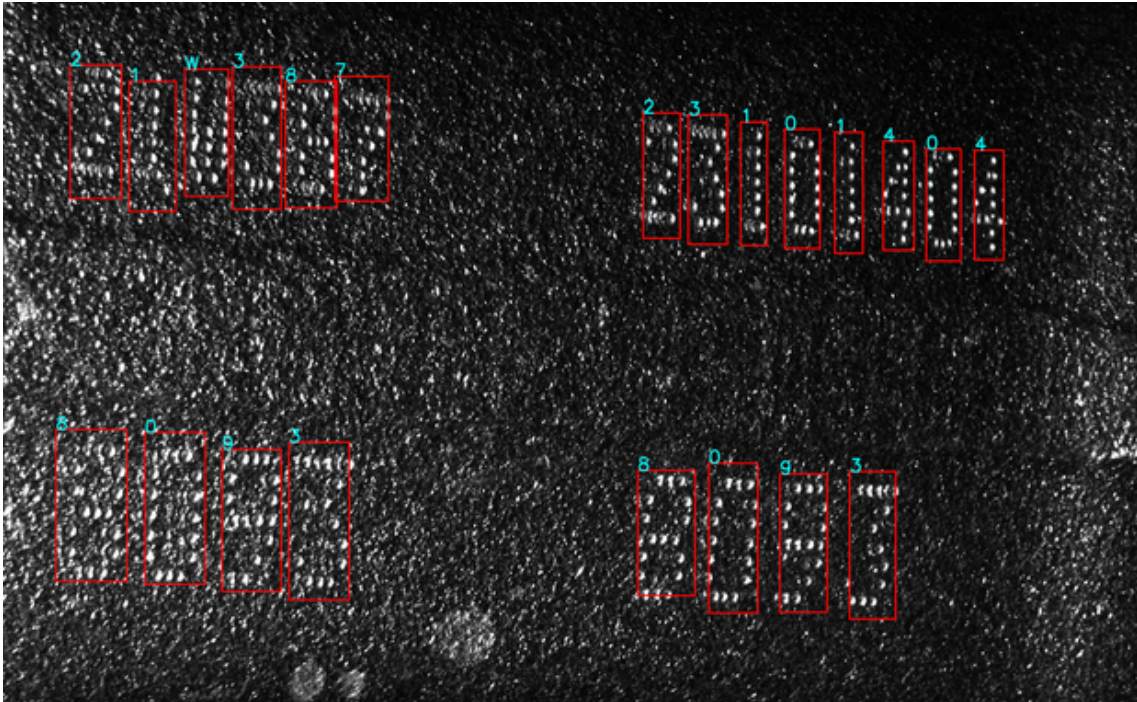


Figure 3.1: Example of serial number from machine part.

lower quality.

3.1.1 Preprocessing

The preprocessing of the data for both the GAN training, and training the CNN, consists of converting the images to tensors, rescaling them in the range $[0, 1]$, and then normalize them to match the output of the generator. The images were originally grayscale, but are converted to three channels, since this is what the GANs and the CNNs are adjusted for. Because the convolutional networks in the GAN requires the images to be of similar size, the training data are resized to 64×32 pixels. The size of the images in the dataset was originally determined by the size of the bounding boxes from the objection detector. Hence, the sizes of the images are not consistent. This means that some images are slightly distorted and drawn out. Figure 3.4 shows the ratio between width and height as well as the estimated height as a function of width for each image and each class, represented by a red line. An estimated slope of 1 would indicate that the ratio on average is equal among all images in that class. This means that reshaping images from class '2' distorts the images equally on average, while class '1' or 'W' likely are more affected by resizing the images.

Figure 3.5 shows the plotted mean and standard deviation of the pixel values for each image, per class, after the preprocessing. Very dark images would have a low mean, while brighter images will have a higher mean. Figure 3.5 thus shows that there is a larger spread of pixel means in the images in some classes compared to others. For example, the pixel means of images showing the digit '0' have a larger spread compared to images from class 'W'. This would indicate that there is a larger

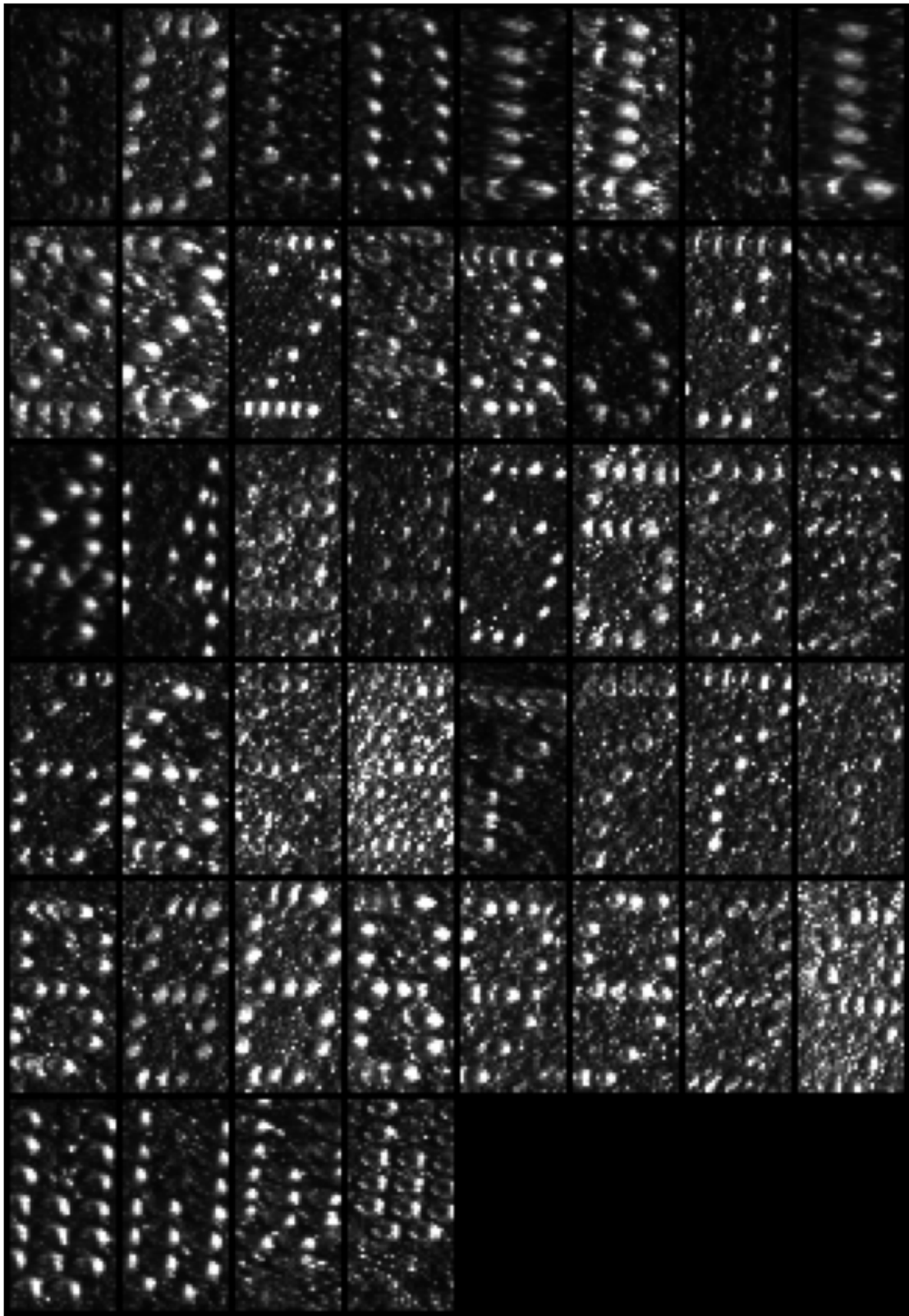


Figure 3.2: Random samples from real dataset. Each row contains four random images from each class.

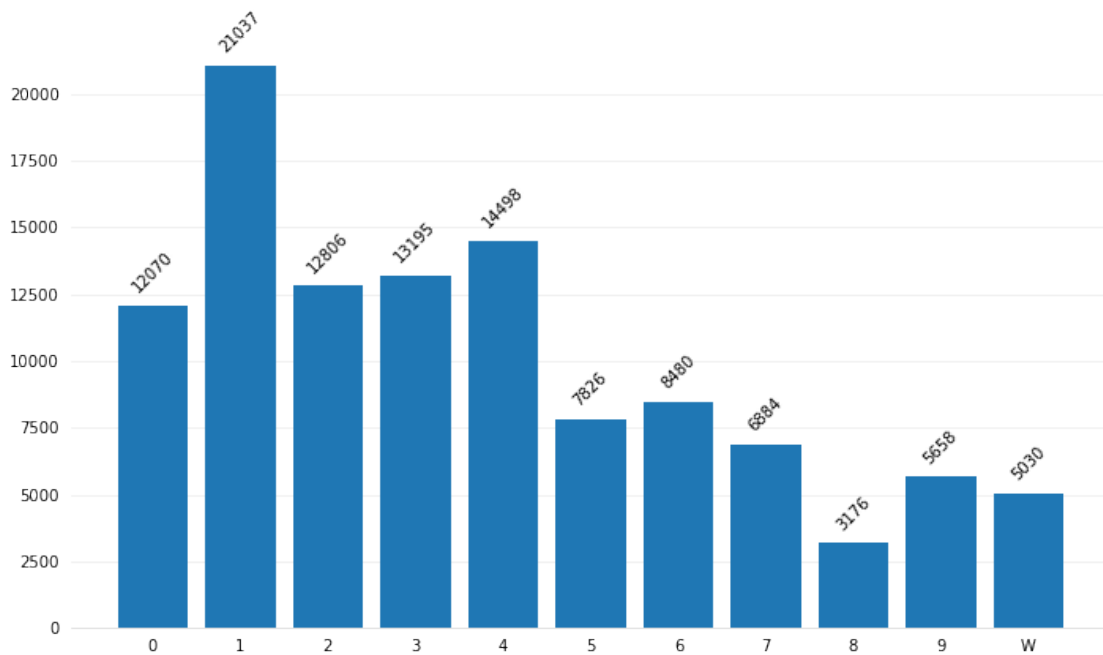


Figure 3.3: Distribution of sample sizes for each image class in the real dataset ($N = 110\,660$)

variation in some classes in terms of brightness in the images. Intuitively, classes with a larger spread of mean values of pixels would indicate more variation in the output. However, while the mean of pixel values in an image indicates the brightness level of an image, it does not say anything of the variation of the motif of an image, i.e. where the digits are located in relation to the center of the image.

3.2 Setup of experiments

The purpose of the study is to investigate how the image classification of the object detection network can be improved, and specifically when training data is limited. The general setup for the experiments conducted, is that a GAN is trained on a set of real images, after which a number of synthetic images are generated with the aim of being as similar to the real data as possible. The real images and the set of synthetic images are then mixed into a training set, and a CNN (from here referenced to as the *control network*) is trained on this dataset for image classification. After training, the CNN is evaluated on a previously unseen set of real images. An overview of the experiment process can be seen in Figure 3.6.

One research question is how much the GAN can improve the control network when training data is limited. To investigate this question, the GANs are trained once on 2 200 real images, i.e. 200 images per class, and once on 11 000 real images, i.e. 1000 real per class. Each time, the GAN is trained for 30 000 steps. After the training, 98 560 synthetic images are generated from each GAN, which constitute the full set of generated images. To see if the effect on performance of the control network is dependent on the number of synthetic images added, a varying number of generated

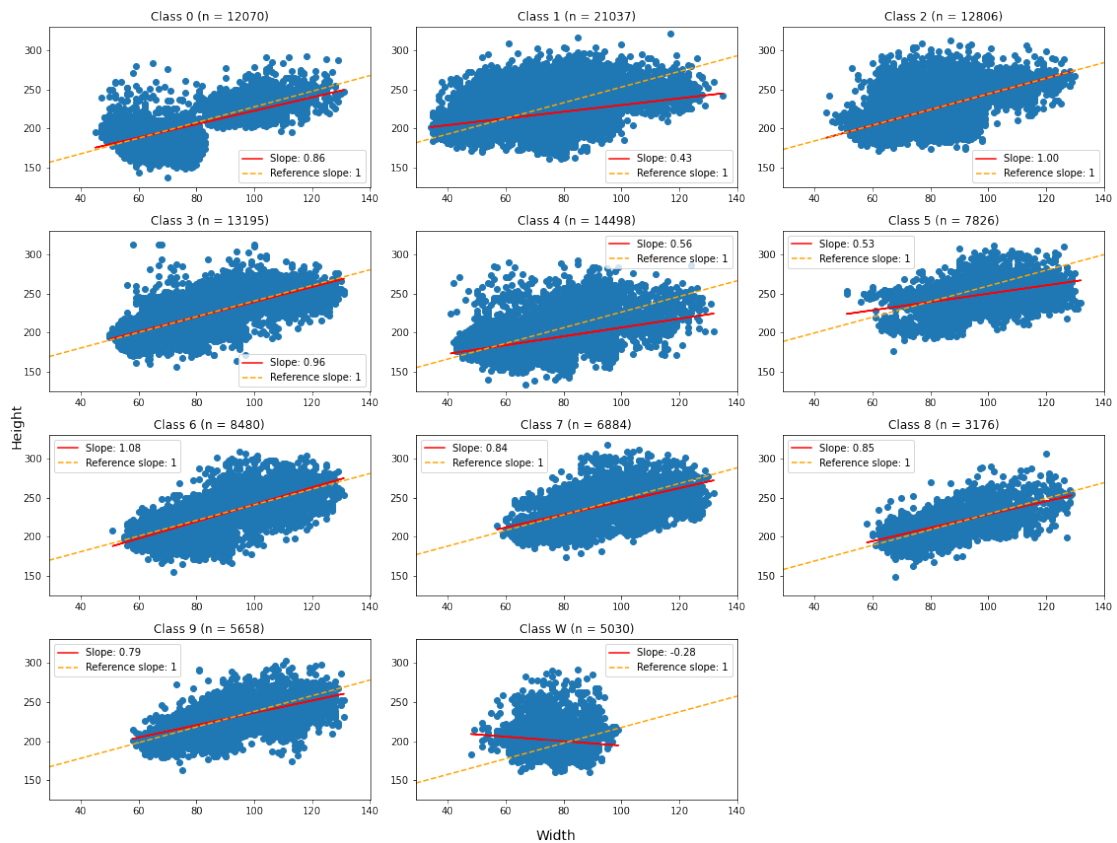


Figure 3.4: Plot of image width and height of each image per class in the real dataset (N = 110 660). The red line shows an estimated regression line, showing the estimated height as a function of width.

3. Methods

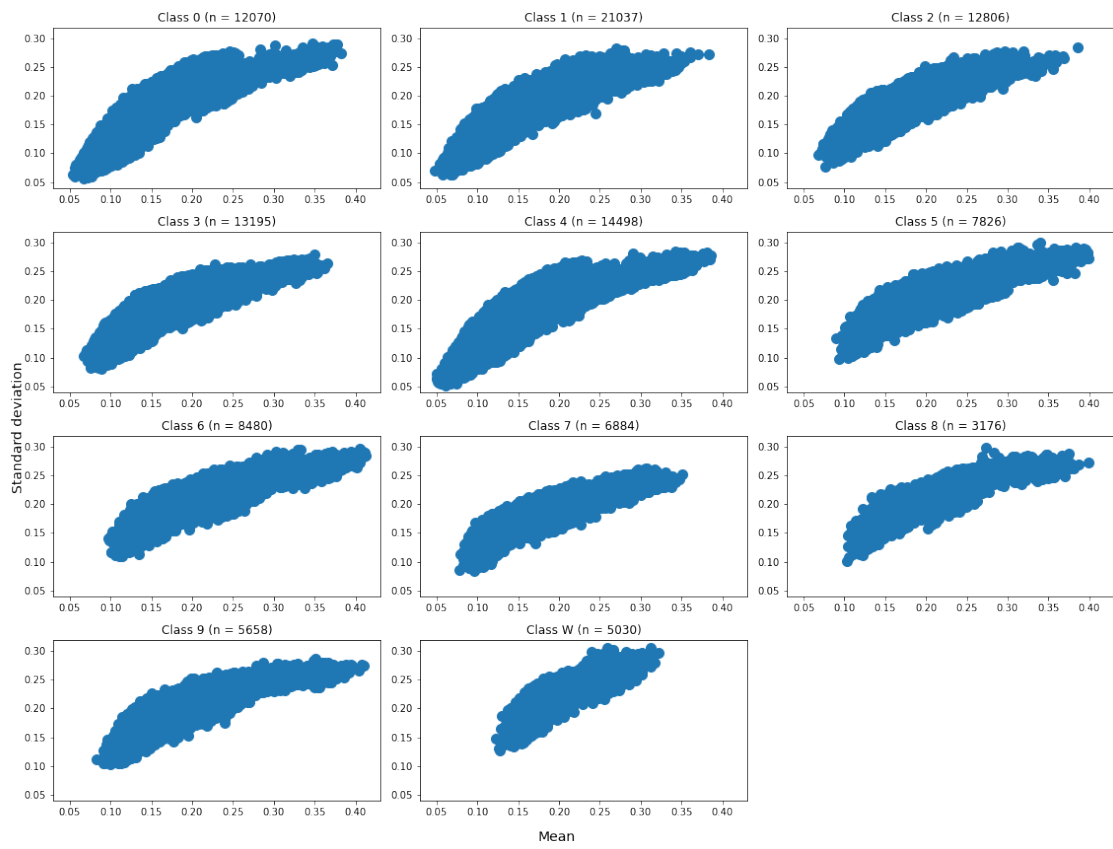


Figure 3.5: Mean and standard deviation of pixel values for each image and class in the real dataset ($N = 110\ 660$)

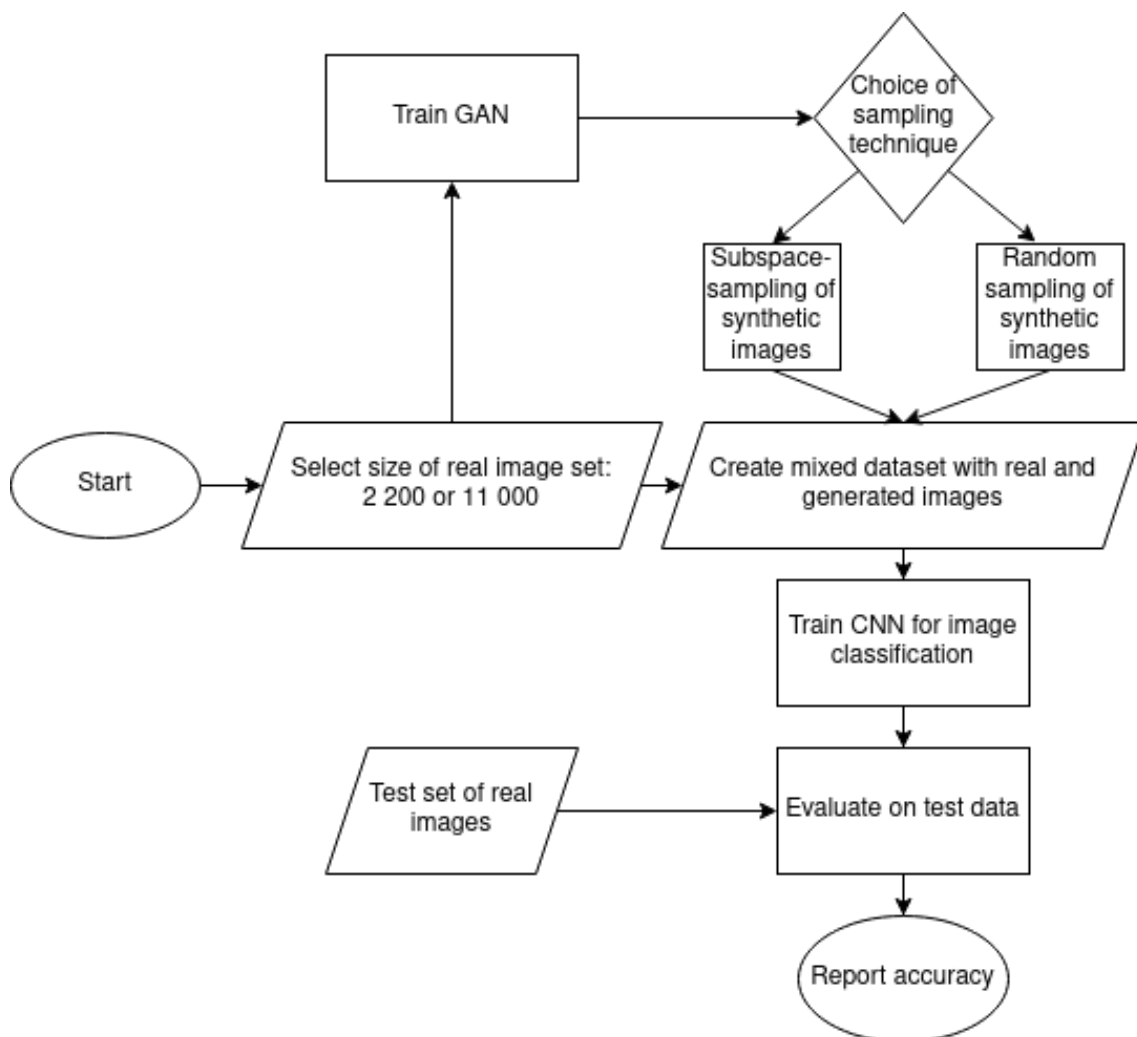


Figure 3.6: Flowchart describing the experiment process. The first decision is the size of the set of real images. The GAN is trained on this dataset, and a sample of the synthetic data is mixed with the real data to form a mixed training set. This is used to train a control network, which is then evaluated on the test set.

images is added to the mixed training set. For each GAN, and each size of real training set, the experiment is repeated when adding a total of 1 100, 5 500, 11 000, 49 280, and 98 660 synthetic images to the mixed training set respectively. The total set of synthetic images are distributed equally between the different classes. While the original training set is unbalanced, using a balanced distribution of synthetic images has the advantage of controlling for potential effects of an unbalanced dataset on the classification network.

In Section 3.2.1, we show that this dataset can be described as being well-represented with a low-dimensional subspace. The intuition is that there is little variation in the images with respect to the motif and the placement of the characters in each image, and that a subspace with much fewer dimensions can represent the data fairly well. This property should generally be true for data in smart manufacturing, since images from production lines, for instance, should contain little variation compared to other natural images. A research question in this thesis is if a method that specifically suits data with this property can be used for sampling synthetic images to improve the performance of the control network.

To investigate if the type of synthetic images added to the training set has any effect on performance of the control network, the experiments are divided into two groups. The difference between the two groups of experiments is how the synthetic images are sampled from the full set of generated images. In the first group, the synthetic images are randomly sampled from each class in the full set of synthetic images, with no consideration to any image properties. In the second group of experiments, the sampling was made with the aim of specifically targeting synthetic images that might be hard for the control network to classify. To achieve this, a principal component analysis (PCA) is made on the real training set used in each experiment, in order to find a subspace that spans the principal components. This can then be used to find images that could be considered hard cases or outliers. This method is further explained in Section 3.2.1. To explore the effect of this sampling technique in more depth, three variations of the sampling technique are used: *random sampling*, *proportional sampling*, and *outlier sampling*.

3.2.1 Principal Component Analysis

PCA is a process often used for dimensionality reduction by finding principal components that explain as much variation in the data as possible. The principal components are linear combinations of the original features in the dataset. In the PCA process using p principal components, the first principal component is computed by finding the linear combination that explains the most variance in the original features. The second component is computed by finding the linear combination that explains the maximum variance when the first component is controlled for, and is hence orthogonal to the first principal component. This process is then iterated p times. The result is a subspace that spans the p principal components, and the dimensions of the original data can be reduced by projecting the data on this subspace. Formally, this can be described as a singular value decomposition of the data matrix \mathbf{X} :

Table 3.1: The sum of explained variance per class and number of principal components (in percent).

Number of components	0	1	2	3	4	5	6	7	8	9	W
512	89.78	84.86	83.00	83.25	89.11	81.67	82.92	79.52	86.32	83.72	89.35
682	92.49	88.51	87.20	87.15	91.98	86.06	87.08	84.48	90.47	88.02	92.41
1024	95.99	93.45	92.79	92.55	95.66	92.15	92.76	91.37	95.61	93.71	96.23

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{W}^T \quad (3.1)$$

where Σ are the singular values of \mathbf{X} , and \mathbf{U} and \mathbf{W} are orthogonal matrices. More importantly, \mathbf{W} forms the orthogonal basis for the principal components. It can be shown that the transformation

$$\mathbf{T}_L = \mathbf{X}\mathbf{W}_L \quad (3.2)$$

maps the data to the subspace that spans the L first principal components, which can further be shown to minimize the so-called squared reconstruction error:

$$\|\mathbf{X} - \mathbf{X}_L\|_2^2 \quad (3.3)$$

This can be used for finding data points with large reconstruction errors, which in this case intuitively should correspond to images that are very different from the rest of the images. The method for doing so in this study can be described as finding subspaces for each respective class by using PCA on the training set, and compute the L_2 -norms to find the data points with the largest reconstruction error. The heuristic used here is to sort the reconstruction errors, and find images above the 90th percentile. These are considered to be *hard cases*, i.e. images that intuitively should be more difficult for the control network to classify.

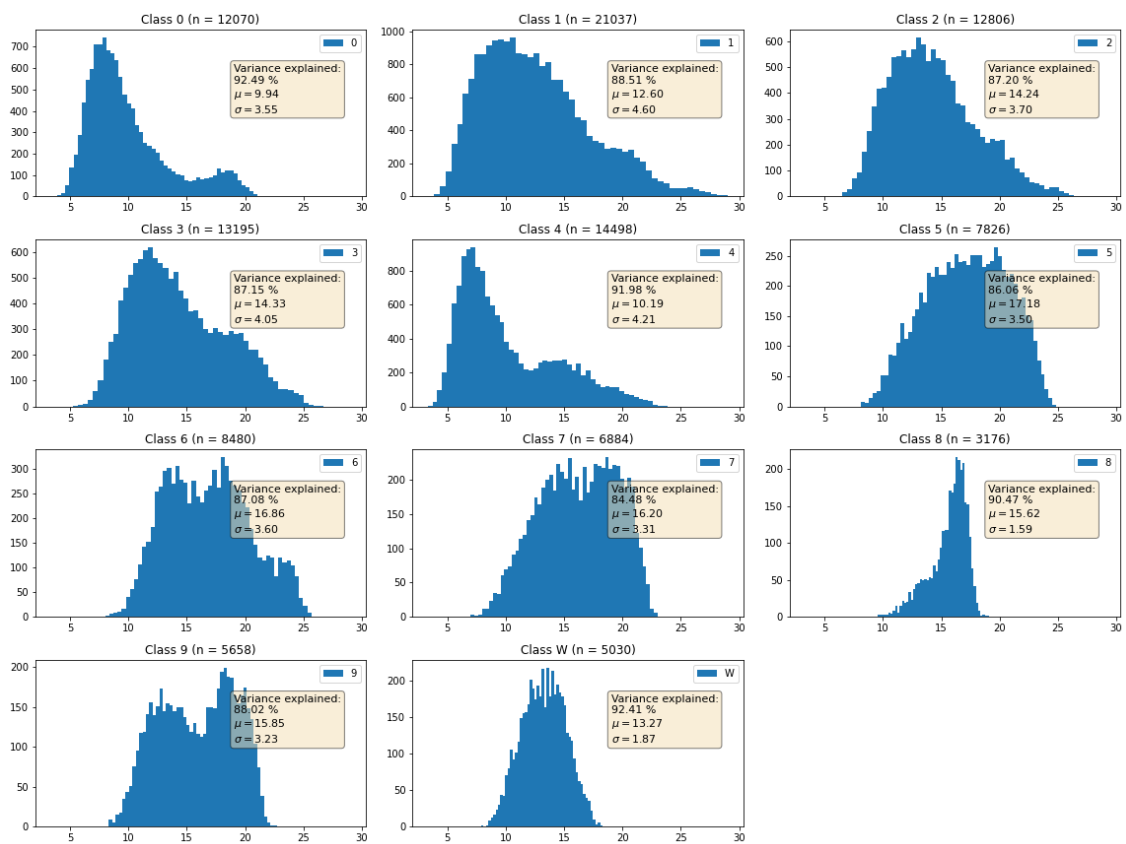
After preprocessing, each image has a size of 64 x 32 pixels, and by that 64 x 32 = 2048 features. Using PCA, the number of features is reduced to see how much variation a subspace explains. Table 3.1 shows how much variation in the dataset the respective number of principal components explains. For example, 682 principal components (around 1/3 of the original features) explains around 90% of the variation in the dataset. Hence, we argue that this dataset can be well-represented by a low-dimensional subspace.

To explore how this can describe the full training set, the L2 norm of the projection on the subspace that spans the 682 principal components was computed and plotted in Figure 3.7. Each histogram shows the distribution of L2 norms for the respective class, and represents the distance from a data point to the subspace that spans the principal components. It can be observed that the first five classes have a wider range of the norms, while other classes, such as 8 and W, are more centered. Even though the mean of the norms are not much higher compared to other classes, it means that the data points are about the same distance from the subspace. Hence, we can expect less variation in these classes.

To visualize this, eight images per class from the subset below the 10th percentile of computed norms using 682 principal components. This is shown in Figure 3.8.

3. Methods

Figure 3.7: Histogram of L2 norms of projections, using 682 principal components.



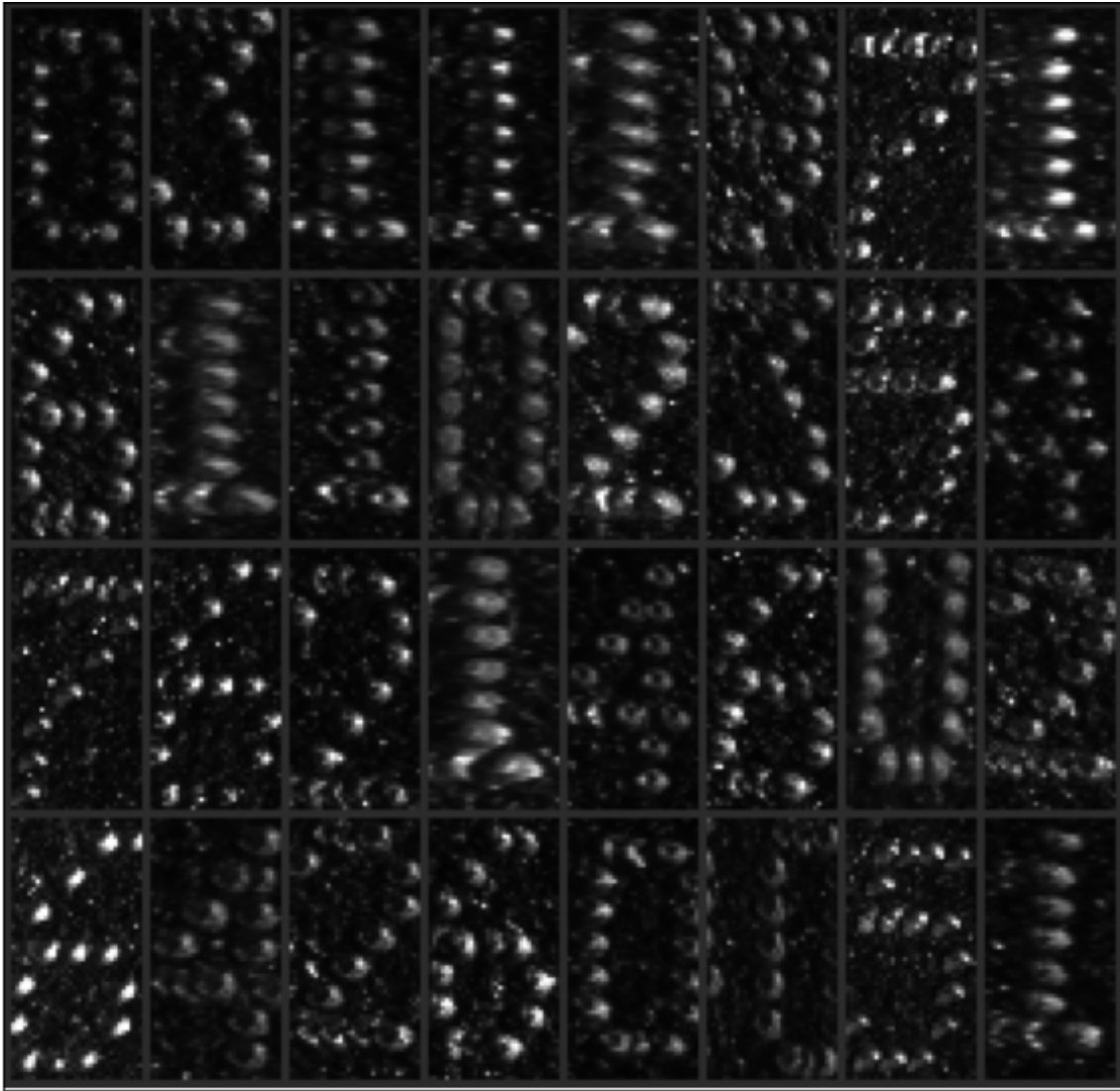


Figure 3.8: Random images from below the 10th percentile of L2 norms using 682 principal components.

Conversely, Figure 3.9 shows eight random images per class from the subset above the 90th percentile. As can be seen in the figures, images with high reconstruction errors tend to be very bright and noisy, while images with low reconstruction error are easier for the human eye to distinguish.

The subspaces computed for each class are what the two of the three variants of sampling techniques, described in Section 3.2, are based on. Random sampling means that synthetic images are sampled randomly from each class, with no respect to the subspace. Outlier sampling means that synthetic images are sampled randomly from only the subset above the 90th percentile of the norm of the projection vectors. I.e., only the hard cases are sampled and added to the training set. Finally, proportional sampling means that half of the synthetic images are sampled from the subset above the 90th percentile, and half of the images are sampled from the rest of the synthetic images.

The proportional sampling is also repeated when using subspaces from the full set.

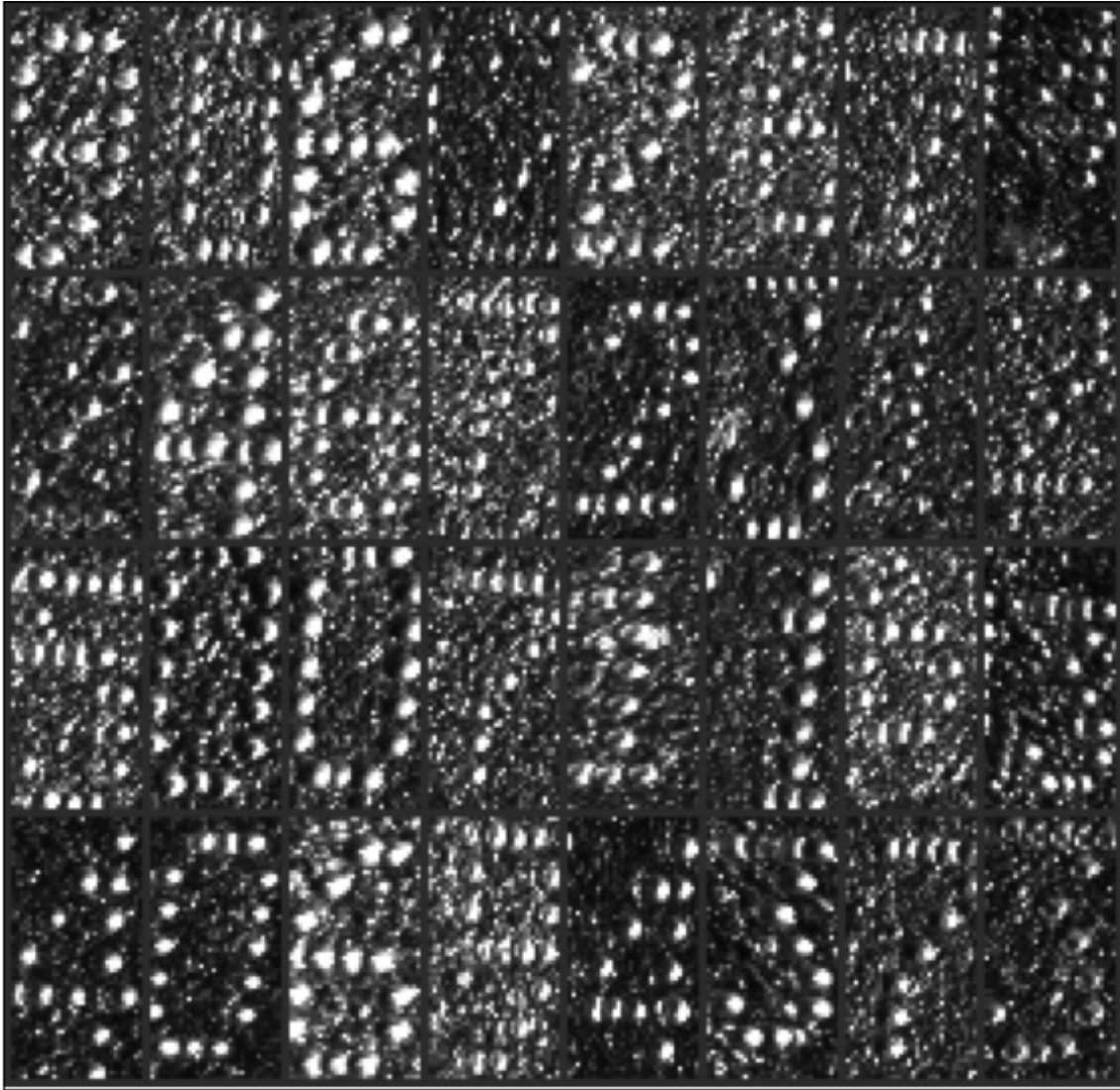


Figure 3.9: Random images from above the 90th percentile of L2 norms using 682 principal components.

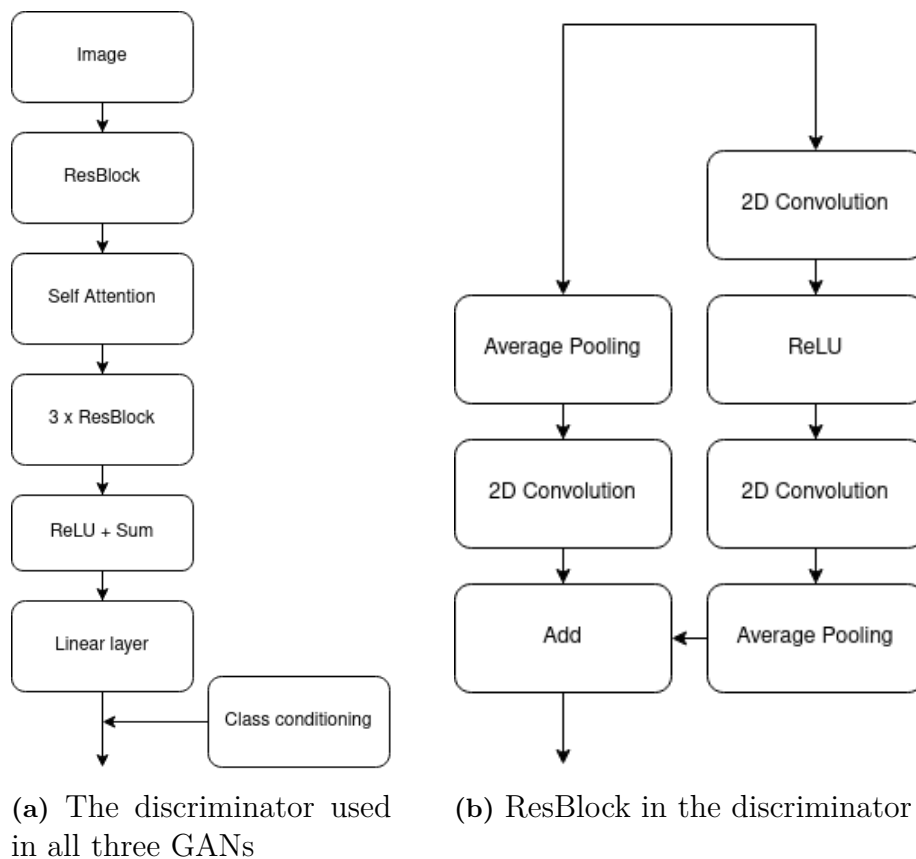


Figure 3.10: Schematic overview of the discriminator

In these cases, the sampling is done similarly as the proportional sampling described above, but the subspace is computed from the full set of real images. This can be considered the ground truth of the subspace, which contains less bias. Intuitively, this would provide a less biased reference for what synthetic images can be considered hard cases.

3.3 Architecture

To motivate the choice of architecture for the GAN, three different models will be tested for comparison. The three models are the ACGAN [40], BigGAN [45], and ReACGAN [46]. These three models are described in Section 2.2.2, and the motivation for choosing them is their important contributions in the GAN literature, as well as showing good results for conditional image generation.

A schematic overview of the architecture can be seen in Figures 3.10 and 3.11. The backbone used in all three GANs is a ResNet style architecture, described in [45, 44]. The difference in architecture between them is how the class information is conditioned in the discriminator and the generator, respectively. The generator in the ACGAN concatenates the class label to the noise vector z , while the BigGAN and the ReACGAN uses class embeddings as illustrated in Figure 3.11a. The class conditioning in the discriminator is the major difference between the BigGAN and

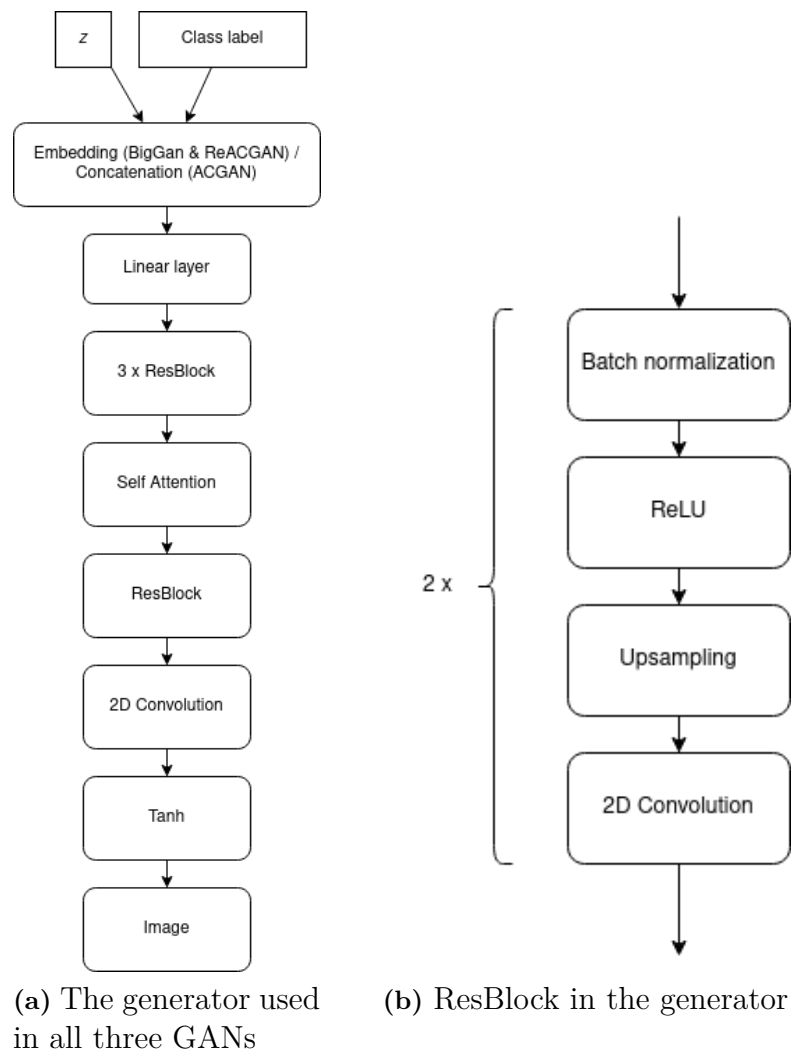


Figure 3.11: Schematic overview of the generator

Table 3.2: Number of total parameters in each respective network. The backbone in all GANs are very similar, with the exception being the ReACGAN which includes projections and embeddings of class information, and hence more parameters than the other GANs.

	ResNet-18 [30]	BigGAN [45]	ACGAN [40]	ReACGAN [46]
<i>Generator</i>		24 308 804	24 308 804	24 308 804
<i>Discriminator</i>		43 992 866	43 992 866	45 164 834
<i>Classification network</i>	11 182 155			

the ReACGAN. The BigGAN uses a projection-based class condition that takes the inner product of the class information and the feature vector from the discriminator, while the ReACGAN uses a projection layer on the class information as well as an embedding of the labels, and following that, feature normalization. Aside from the architecture, the major differences between the three GANs presented here is the loss in the adversarial training. While the ACGAN uses cross-entropy loss, the BigGAN uses hinge loss, and the ReACGAN uses D2D-CE (described in more depth in section 2.2.2). The total number of parameters used in the respective GANs are described in Table 3.2.

3.4 Evaluation

When evaluating the experiments, three methods are used:

- Visual inspection: Random samples from the generated images are inspected in order to determine the quality of the training. While this method is subjective, it is a simple way of ruling out results that are clearly bad.
- GAN-train and GAN-test: As described in 2.2.3, a CNN is trained for image classification with the real only data, and the generated only data respectively. GAN-train means that the CNN is trained on the generated only data, and evaluated on a specific test set. In the GAN-test, the CNN is trained on real data only, and evaluated on the full set of synthetic images. This method provides an indication of if the synthetic data has captured the mode of the real data, as well as if there is enough variation in the synthetic data to classify the real data sufficiently.
- Evaluation using the control network: Since the overall task is to improve the performance of the control network, the experiments are evaluated by mixing real and synthetic data to train a CNN. The control network is then evaluated on the test set. By comparing this to a benchmark performance, it is possible to evaluate the effect of augmenting the training set with synthetic data.

The CNN used for image classification in all experiments is based on the architecture of ResNet-18 [30], but without using pre-trained weights. This CNN is supposed to act as the control network, but is not actually part of the object detector used in the original task of quality assurance. The total number of parameters in the CNN is described in Table 3.2.

Table 3.3: Evaluation scores from the initial comparison of GANs

	GAN-train ¹	GAN-test ²	IS	FID
ACGAN	85.92	99.38	3.08	184.31
ReACGAN	78.30	96.02	3.02	235.16
ReACGAN (DiffAug)	74.86	99.59	2.4	198.91
BigGAN	72.94	90.69	2.36	297.59
BigGAN (DiffAug)	65.09	91.3	2.56	168.52
Real data	95.31	–	–	–

¹ GAN-train means a CNN trained on a set of synthetic images and evaluated on a test set of real images. The score is the reported accuracy expressed in percentages.

² GAN-test means a CNN trained on a set of real images and evaluated on the synthetic images. The score is the reported accuracy expressed in percentages.

3.4.1 Test data

The test set used for evaluation of the experiments consists of 1 000 randomly sampled images per class from the full dataset of real images, i.e. a total of 11 000 images. The images in the test set are not included in any training sets.

3.5 Comparing architectures

While the ReACGAN has showed to often outperform the other GANs when evaluated on different datasets [46], there is no obvious theoretical preference to what GAN would work best for the data used in this project.

The GANs that are initially chosen for this project are trained on the real training set for 30 000 steps. Each network is trained with a batch size of 128, and with a similar setup of hyperparameters to provide a fair comparison, since a thorough optimization of hyperparameters for each network would have been too time-consuming. The BigGAN and the ReACGAN are both trained with and without differential augmentation [56]. This is a method for improving the data efficiency of GAN training by applying differentiable augmentations on both the fake and the real data, and has been shown to improve the stability and performance of GAN training. However, this method is adjusted for BigGAN, and hence was not used on ACGAN in this test.

The networks are evaluated using GAN-train and GAN-test, described in Section 3.4. For completion, IS-score [49] and FID-score [50] are computed. While this is the most common way to evaluate GANs in the literature, it might not correspond well to real-world tasks [51]. Finally, the results are compared to the benchmark performance of a classifier trained on the real dataset. For this comparison, 1000 images are randomly sampled from each class, and evaluated on the test set.

The results from the comparison are shown in Table 3.3. The GAN-train score of the real data shows an accuracy of 95.31% when evaluated on the test set. This

can be thought of as the upper bound of the performance of the GAN-train on the GANs, since a perfectly trained GAN indicates that the synthetic distribution is equal to the real data distribution. As can be observed in Table 3.3, none of the trained GANs come close to this upper bound, indicating that the GANs are not perfectly trained. Some reasons for this might be mode-droppings, where the generated data is not as diverse as the real data, or that the classifier network does not learn properly from the synthetic data [51]. The GAN that performs best in this comparison is the ACGAN with a GAN-train accuracy of 85.92%. The main difference between this implementation of ACGAN and BigGAN is the conditioning method of the discriminator, and it seems that a projection-based discriminator is not advantageous in this case.

The GAN-test scores shows how well the distribution of the generated data matches the real distribution. The GAN-test is not performed on the real data, as this would in practice be the same as GAN-train on the real data. Table 3.3 shows that all GANs achieved a GAN-test score fairly close to the GAN-train score of the real data, meaning that in all cases the real distribution is captured quite well.

To conclude this comparison of GANs, the ACGAN was shown to perform best, followed by the ReACGAN. Samples of the generated data from all GANs are shown in Figure 3.12.

We can see that using differentiable augmentation does not increase the performance in this test. However, the training set was quite large, and this method is developed when training GANs on a limited amount of data. Thus, both the ACGAN and the ReACGAN with differentiable augmentation will be used in the future tests, since these will be performed using a smaller training set.

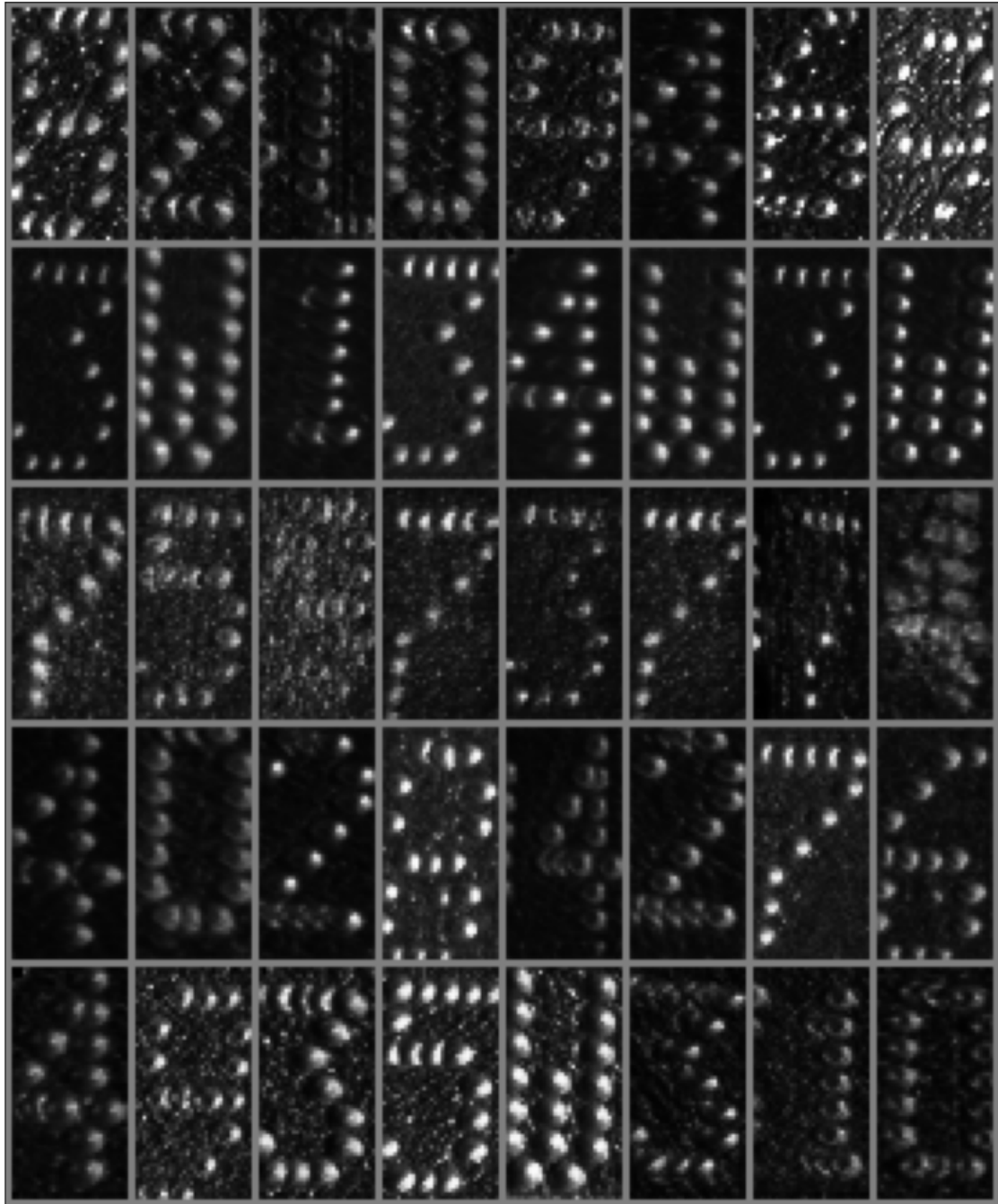


Figure 3.12: Random samples from the trained GANs. Each row shows eight samples from the GANs in the following order (from top to bottom): ACGAN, BigGAN, BigGAN (diffaug), ReACGAN, ReACGAN (diffaug)

4

Results

This section will first present an initial evaluation of the performance of the GANs, and following that, a presentation of the results from the conducted experiments. An overview of the different datasets used in the experiments are shown in Table 4.1.

4.1 Initial evaluation of the GANs

Table 4.2 shows the evaluation of the different GANs when trained on 11 000 and 2 200 images, respectively, as well as the performance of the classifier network when using only real training data. When conducting the experiments, 8 960 images per class were generated by each GAN (98 560 images in total) to be used for augmenting the real training data.

As observed in Table 4.2, both the ACGAN and the ReACGAN scored high on GAN-test, meaning that a CNN trained on only real images can accurately predict the synthetic data. Note that the GAN-train scores of the real data is different from what is reported in Table 3.3. The reason for this is that the classification network used in Table 3.3 is trained on the full training set, while the ones used in Table 4.2 are trained on the smaller training sets, and hence received slightly lower scores.

The results from Table 4.2 suggests that the synthetic images are realistic approximations of the real dataset. Looking at the GAN-train scores, which provide an indication of how well the synthetic data captures the real distribution, the ReACGAN performs better than the ACGAN. This is more significant when the GANs are trained on fewer real images, and one reason for that is likely that the ReACGAN is trained using differentiable augmentation [56].

This difference can also be observed in Figures 4.1, 4.2, 4.3 and 4.4 showing random samples from each GAN. The ReACGAN seems slightly better at capturing the noise

Table 4.1: Overview of the different datasets used in the experiments. The smaller training sets are randomly sampled from the full training set. This is different from the full dataset described in the method chapter, since this includes both the full training set and the test set.

	Total images	Number of images, classwise										
		0	1	2	3	4	5	6	7	8	9	W
<i>Small training set (200)</i>	2 200	200	200	200	200	200	200	200	200	200	200	200
<i>Small training set (1000)</i>	11 000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
<i>Full training set</i>	99 660	11 070	21 037	11 806	12 195	13 498	6 826	7 480	5 884	2 176	4 658	4 030
<i>Test set</i>	11 000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000

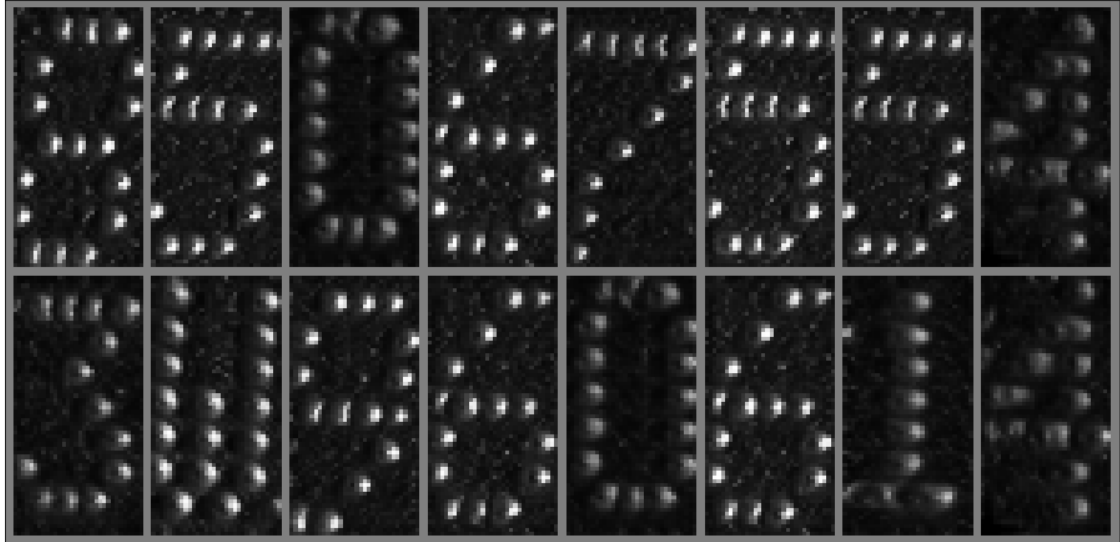


Figure 4.1: Random samples of generated data from ACGAN, trained on 2200 samples.

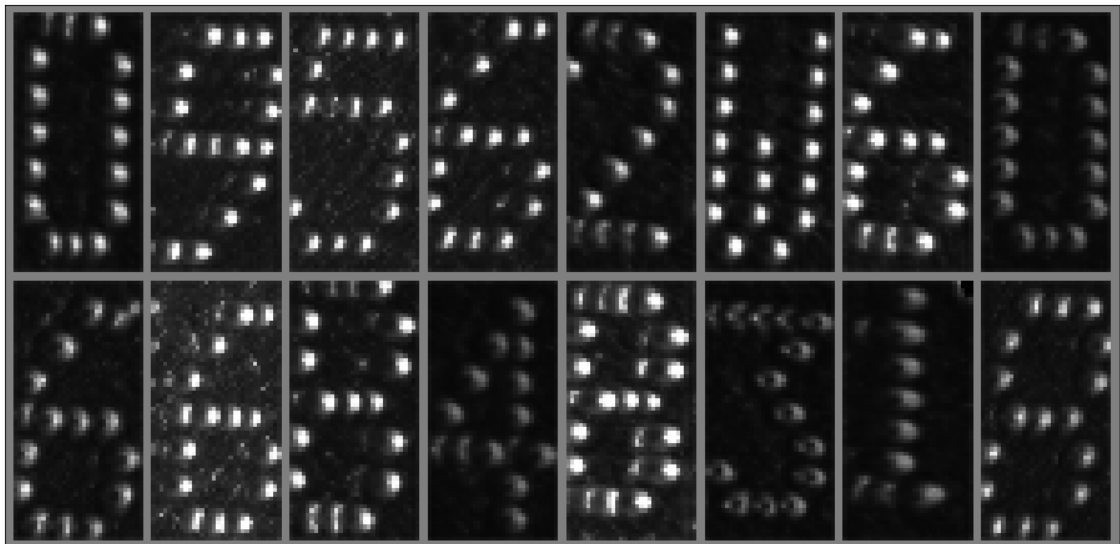


Figure 4.2: Random samples of generated data from ACGAN, trained on 11 000 samples.

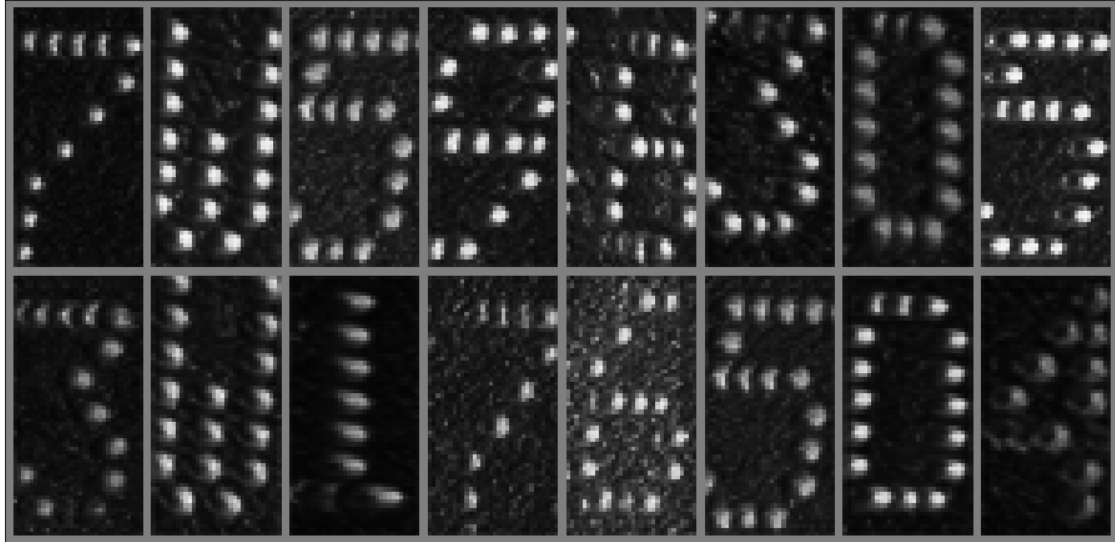


Figure 4.3: Random samples of generated data from ReACGAN, trained on 2200 samples.

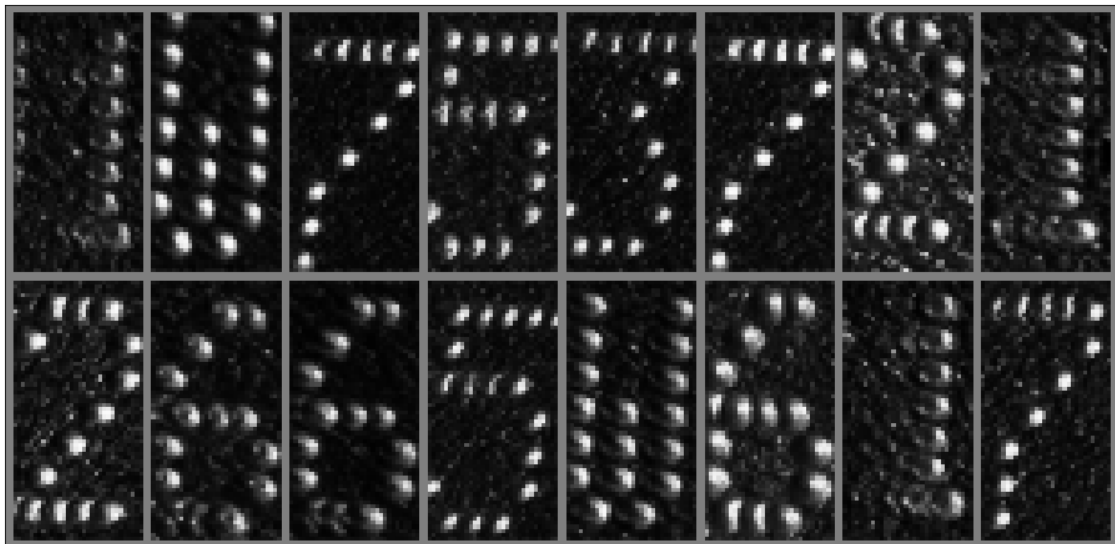


Figure 4.4: Random samples of generated data from ReACGAN, trained on 11000 samples.

Table 4.2: Evaluation scores of the ACGAN and the ReACGAN trained on 1000 and 200 real images per class respectively. GAN-train scores when using only real data are included as a benchmark.

		<i>IS</i>	<i>FID</i>	<i>GAN-train</i>	<i>GAN-test</i>
ReACGAN	200	2.92	112.60	51.03%	90.42%
	1000	3.06	71.51	69.05%	99.63%
ACGAN	200	2.38	201.31	34.50%	100%
	1000	2.77	192.75	68.48%	99.68%
Real data	200	–	–	85.57%	
	1000	–	–	95.06%	

in the background, as the ACGAN mostly outputs a very dark background without much noise. It is, however, difficult to observe any major differences between the same GAN trained on different size of the training set.

4.2 Experiments

The first group of experiments were conducted by randomly sampling a number of generated images from the set of images generated by each GAN, and add these to the real images to construct a training set. In the second group of experiments, the generated images are sampled with respect to the length of the projection vector of each respective image on the subspace that spans the principal components of the image set (further described in 3.2.1). The purpose of this is to specifically augment the training set with generated images that can be considered more difficult to classify. For each experiment, the classifier network is trained on the augmented training set and evaluated on the fixed test set. The number of added synthetic samples per class are 100, 500, 1000, 4480, and 8960 respectively.

4.2.1 Random sampling of synthetic data

Figure 4.5 shows the results of adding a varying size of synthetic images randomly sampled from the full set of images generated by the ACGAN [40]. E.g., in the first experiment, a total of 1 100 images are added to the real image set, and in the second experiment a total of 5 500 images are added, etc.

The figure shows the increase in performance when adding synthetic data to a training set with 1000 real images per class, and also the accuracy of the classifier network when only real images are used, represented by the dashed line. Adding only 100 images per class did not yield any increase in performance, but adding more fake images to the training set achieved a higher accuracy. The highest achieved accuracy was 99.17% when adding 8 960 images per class (a total of 98 560 fake images), which is a slight increase from 98.34% when adding a total of 49 280 fake images. Using only real data in the training set achieved an accuracy of 95.06%, meaning that adding 8 960 images per class increased the performance with over 4 percentage units.

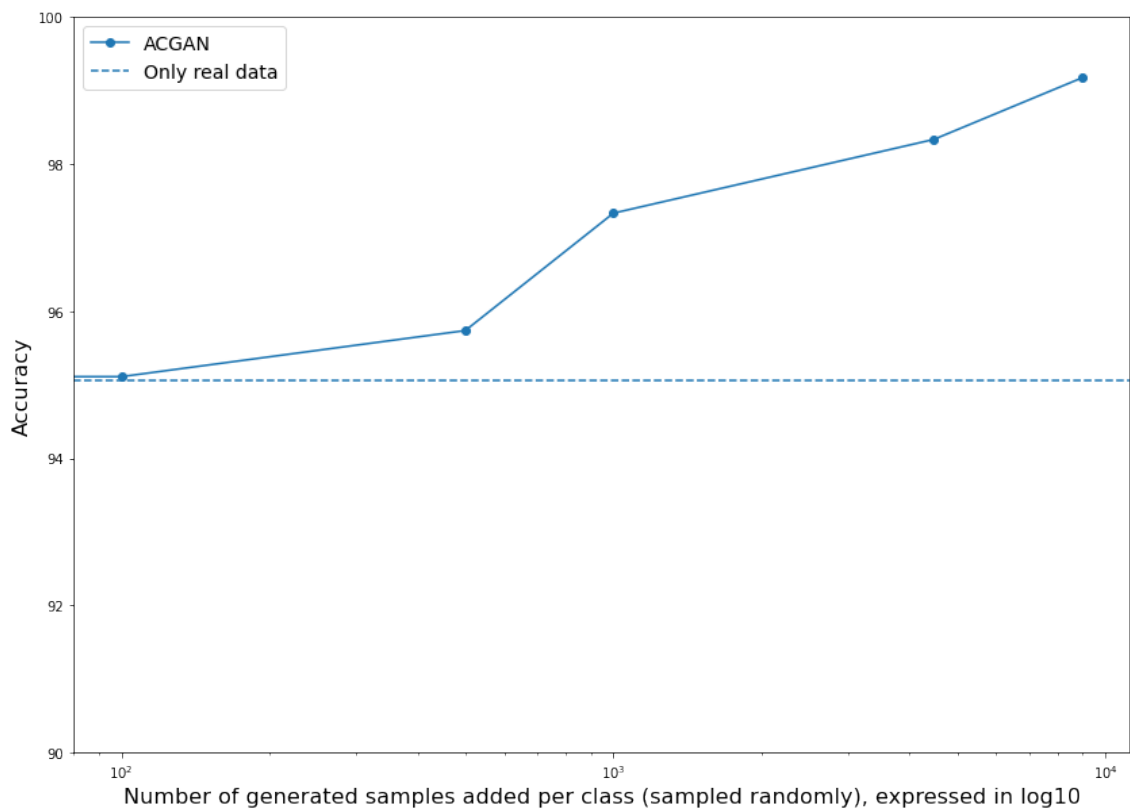


Figure 4.5: Performance of classifier network when adding randomly sampled images generated by the ACGAN. Size of real training set: 11 000. The classifier network is evaluated on the test set (11 000 test images). The dashed line shows the performance of the classifier network when using only real data. Number of added samples are expressed in logarithms with base 10.

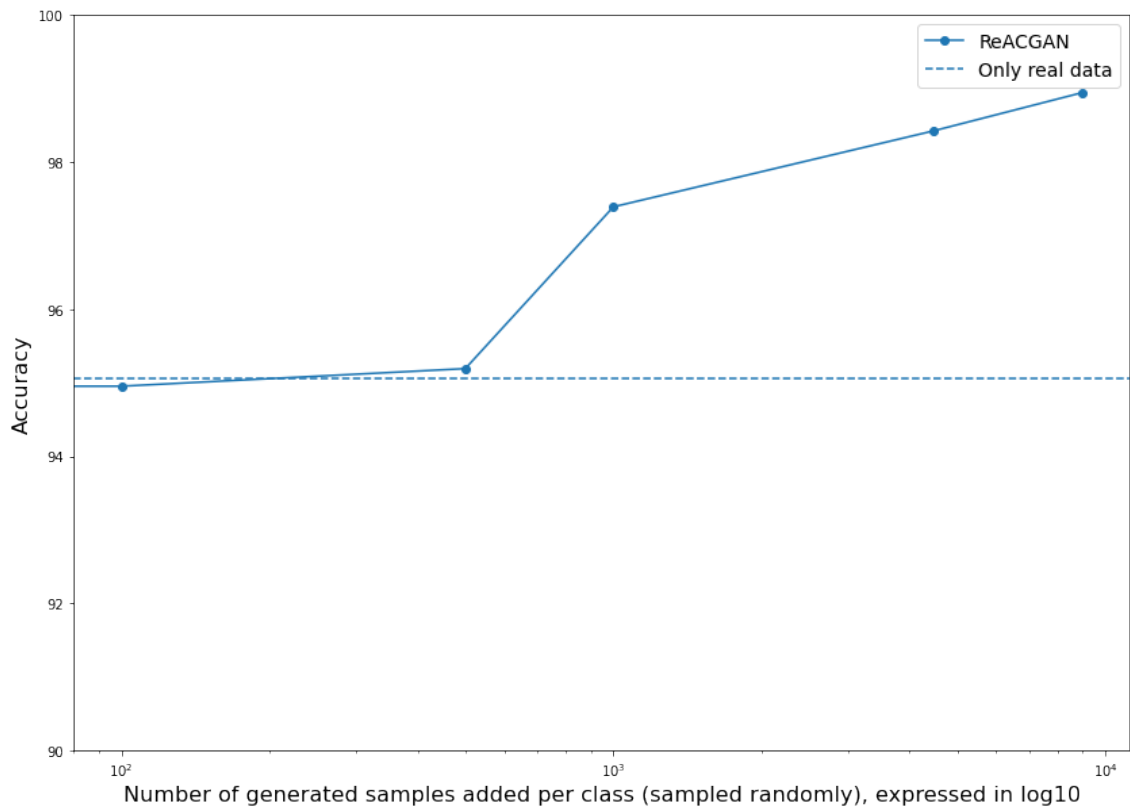


Figure 4.6: Performance of classifier network when adding randomly sampled images generated by the ReACGAN. Size of real training set: 11 000. The classifier network is evaluated on the test set (11 000 test images). The dashed line shows the performance of the classifier network when using only real data. Number of added samples are expressed in logarithms with base 10.

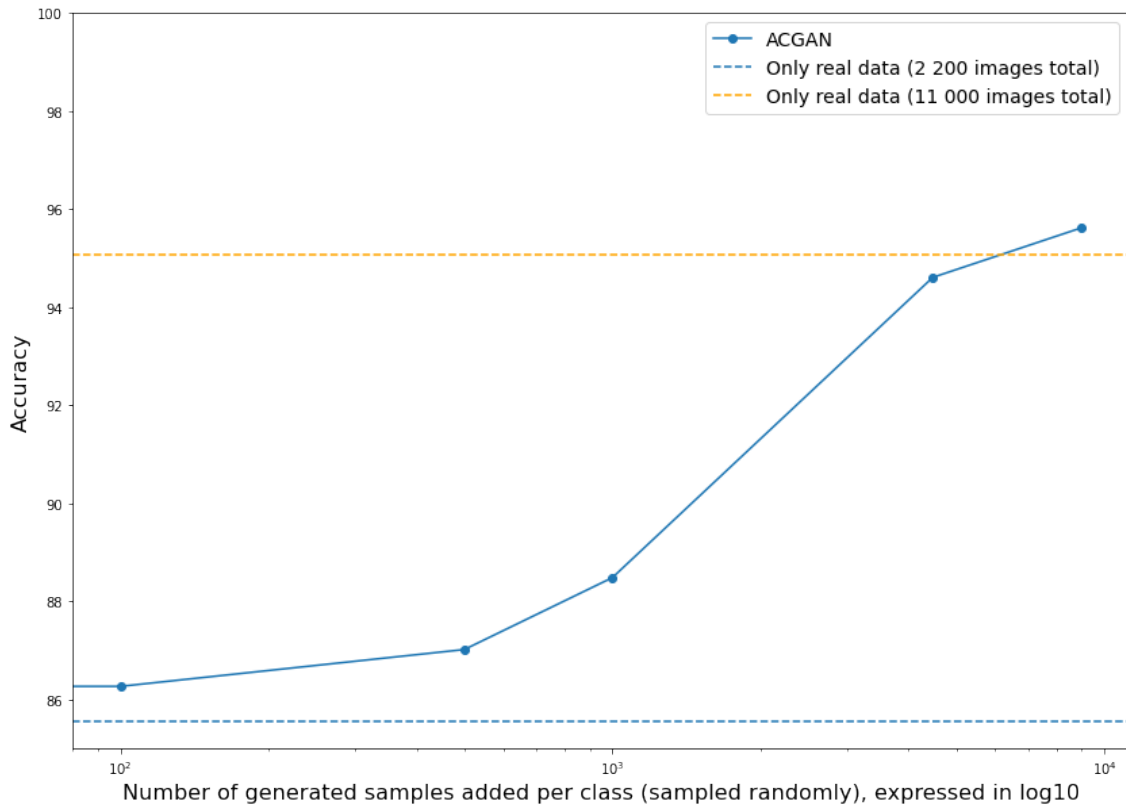


Figure 4.7: Performance of classifier network when adding randomly sampled images generated by the ACGAN. Size of real training set: 2 200. The classifier network is evaluated on the test set (11 000 test images). The dashed line shows the performance of the classifier network when using only real data.

A similar pattern is seen in Figure 4.6, that shows the same experiment when using images generated by the ReACGAN [46]. In this setup, adding 500 fake images per class only had a very minor positive effect, although there was a large increase of accuracy when adding more than that. The highest accuracy was 98.94% when adding 8 890 images per class, which is slightly below what the ACGAN performed. The increase in accuracy when adding a large number of fake images indicates that the GANs managed to learn the underlying distribution of the real data. That is, adding more synthetic data to the training set had a beneficial effect on what the control network was able to learn.

In the second part of this group of experiments, the GANs were trained on 200 real samples per class, i.e. 2 200 training samples in total. Similar to the previous experiment, an increasing number of synthetic data was added to the training set in each respective experiment to see the effect of the augmentation. The classification network was, as before, retrained for the different steps of the experiment, and evaluated on the test set.

Figure 4.7 shows the accuracy of the classification network when augmenting the training set with varying number of generated samples from the ACGAN. The accuracy when using 2 200 (85.57%) and 11 000 (95.06%) real only samples is shown with dashed lines in the figure. The performance increased when adding only 100

4. Results

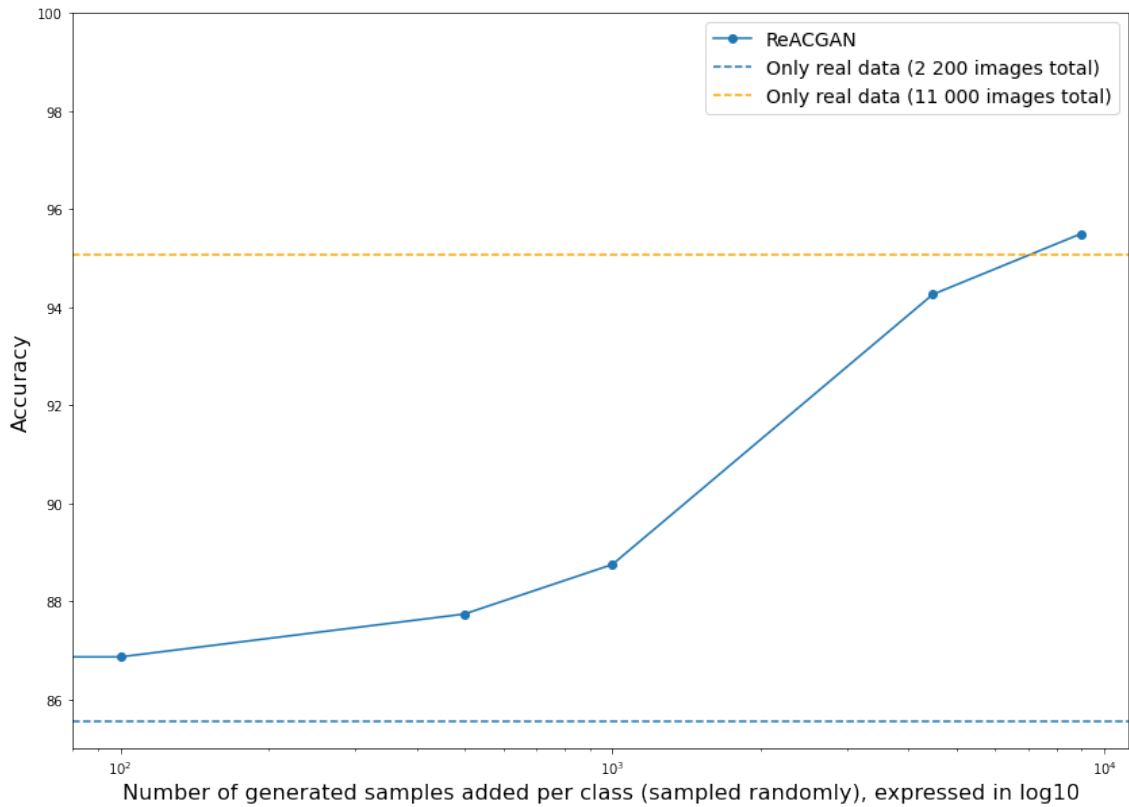


Figure 4.8: Performance of classifier network when adding randomly sampled images generated by the ReACGAN. Size of real training set: 2 200. The classifier network is evaluated on the test set (11 000 test images). The dashed line shows the performance of the classifier network when using only real data.

synthetic samples per class to 86.27%, and increased further when adding more samples. The highest accuracy achieved was 95.62%, when adding 8 960 synthetic samples per class. As can be seen in the plot, this surpasses the accuracy of the control network when trained on 11 000 real images only.

A similar pattern can be seen in Figure 4.8, where the same experiment setup using the ReACGAN is shown. In this experiment, the highest achieved accuracy was 95.50% when adding 8 960 synthetic images per class, which was slightly below what the ACGAN achieved. In both cases, the functions are monotonically increasing but with diminishing returns.

The experiments with adding randomly sampled synthetic data to the training set shows that the performance of the classification network can be increased, especially when adding a large number of fake samples. Both the ACGAN and the ReACGAN achieved accuracies around 99% when trained on 11 000 real images. When trained on only 2 200 real images, both GANs were also able to exceed the benchmark of the model trained on 11 000 real only images.

4.2.2 Subspace sampling

The second group of experiments is based on subspace sampling of synthetic data, instead of the random sampling used in the previous group of experiment described in Section 4.2.1. The aim is to sample synthetic images that specifically might be more difficult for the classification network to label correctly. This method is described in Section 3.2.1, and can be summarized as using PCA for constructing a subspace in the real image data. Assuming the image data can be well-represented by a smaller subspace, this subspace is used for finding images that are farther away from the subspace compared to others. Intuitively, these will be the images that the classifier network will have the most difficulty with labeling correctly.

For these experiments, the norms of the projection vectors of each image on the subspace is sorted by size, and the images above the 90th percentile are considered to be *hard cases*. Hence, the sampling technique used in this group of experiments aims to specifically sample hard cases and adding these to the training set. The sampling techniques used in this group of experiments are proportional sampling and outlier sampling, based on the subsets computed for each class, using the small subsets of real data. These samplings are then repeated using the full set of real data.

Figures 4.9 and 4.10 show the accuracy of the classifier network when adding a varying number of synthetic images to the training set generated by the ACGAN and the ReACGAN respectively. In these experiments, the GANs are trained on 11 000 real images. In both cases, only adding outliers achieved the lowest accuracy. As can be seen in the plots, these functions are not monotonically increasing, meaning that adding a higher number of synthetic images that can be considered outliers sometimes actually decreased the accuracy. The reason for this is likely that adding a large number of hard cases have a negative effect on how the classifier network learns the normal images. Proportional sampling using only the small training set achieved higher accuracy than proportional sampling using the full training set in the case of the ReACGAN when adding a large number of synthetic images, which was not true for the ACGAN. An explanation for this is that the sampled synthetic images from the ReACGAN is more similar to the real distribution, compared to the images generated by the ACGAN, and hence is less biased. The highest achieved accuracy for the ReACGAN using the subspace sampling techniques was 98.76%, and 98.04% for the ACGAN. In neither case was this better than randomly sampling synthetic images used for augmenting the training sets.

A similar situation is observed when using GANs trained on 2 200 real samples. Figures 4.11 and 4.12 shows the accuracy of the classifier network when adding a varying number of synthetic images to the training set generated by the ACGAN and the ReACGAN respectively. Similar to the previous experiments, adding only hard cases tended to decrease the accuracy as the number of added synthetic samples increased. However, in some cases, the outlier sampling outperformed the other sampling methods, e.g. when adding 500 samples per class using ReACGAN trained on 11 000 real images. For both the ACGAN and the ReACGAN, using the full set for creating the subspace achieved a higher accuracy than using the small dataset. Because the small dataset in this case consisted of only 2 200 images (compared to 99 660 in the full training set), it can be expected that the subspace will be

4. Results



Figure 4.9: Accuracies from augmented training set using images generated by the ACGAN. 11 000 real images were used.

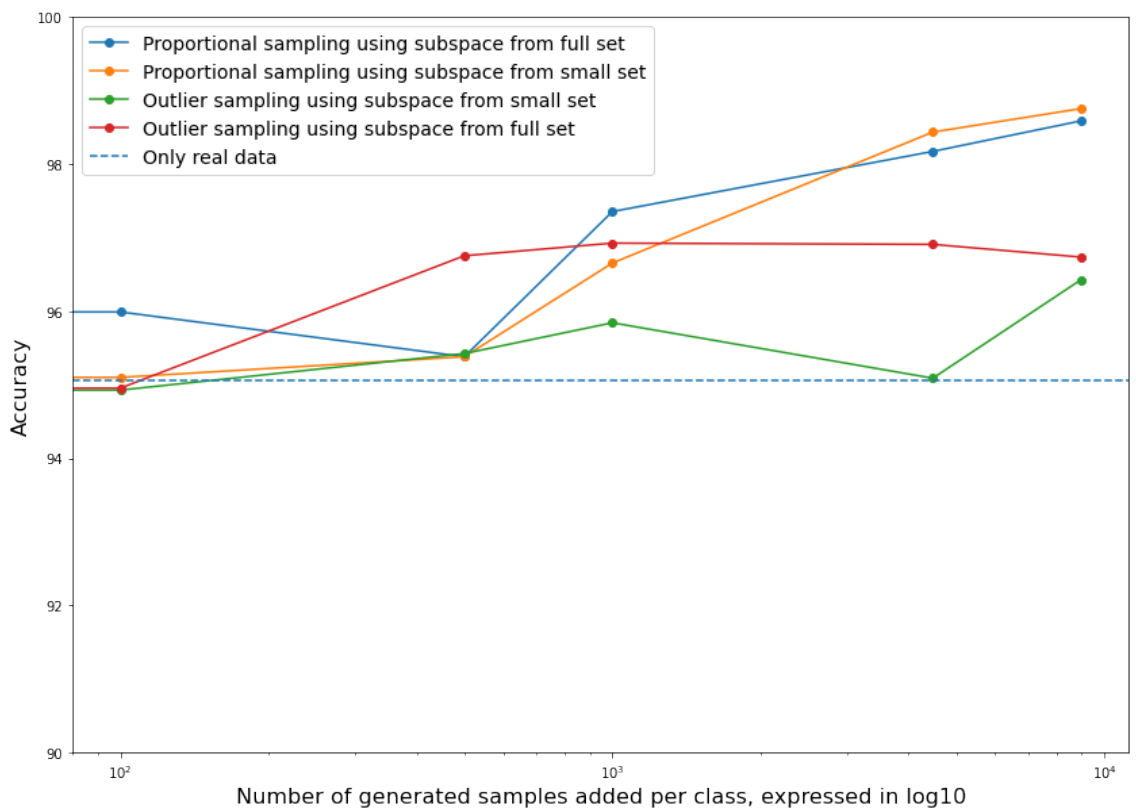


Figure 4.10: Accuracies from augmented training set using images generated by the ReACGAN. 11 000 real images were used.

more biased, and hence not as precise when sampling hard cases from the synthetic image set. Using the subspace from the full training set for sampling hard cases achieved an accuracy of 94.5% for the ReACGAN and 94.15% for the ACGAN. In the case of the ACGAN, using the subspace from the small set achieved an even higher accuracy of 94.78%. In neither case, this surpassed the accuracy when using random sampling.

All reported accuracies can be seen in Table 4.3. The highest accuracy per GAN and training set sized use is bolded, to show which setup achieved the best performance. In all setups, using random sampling when adding 8960 generated images per class (98 560 in total) achieved the highest accuracy. The ACGAN was performed slightly higher than the ReACGAN in this setup. The highest accuracy achieved overall was 99.17%, when using random sampling of 8960 images per class, generated by the ACGAN.

4. Results

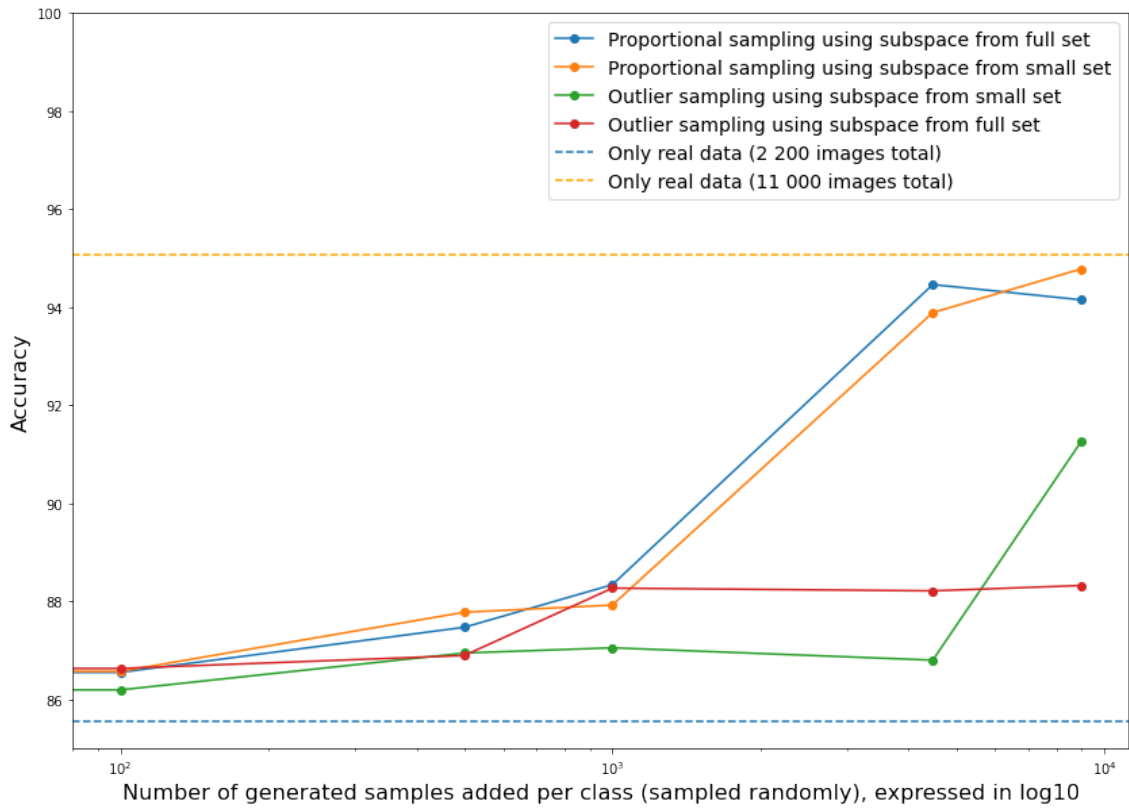


Figure 4.11: Accuracies from augmented training set using images generated by the ACGAN. 2 200 real images were used.

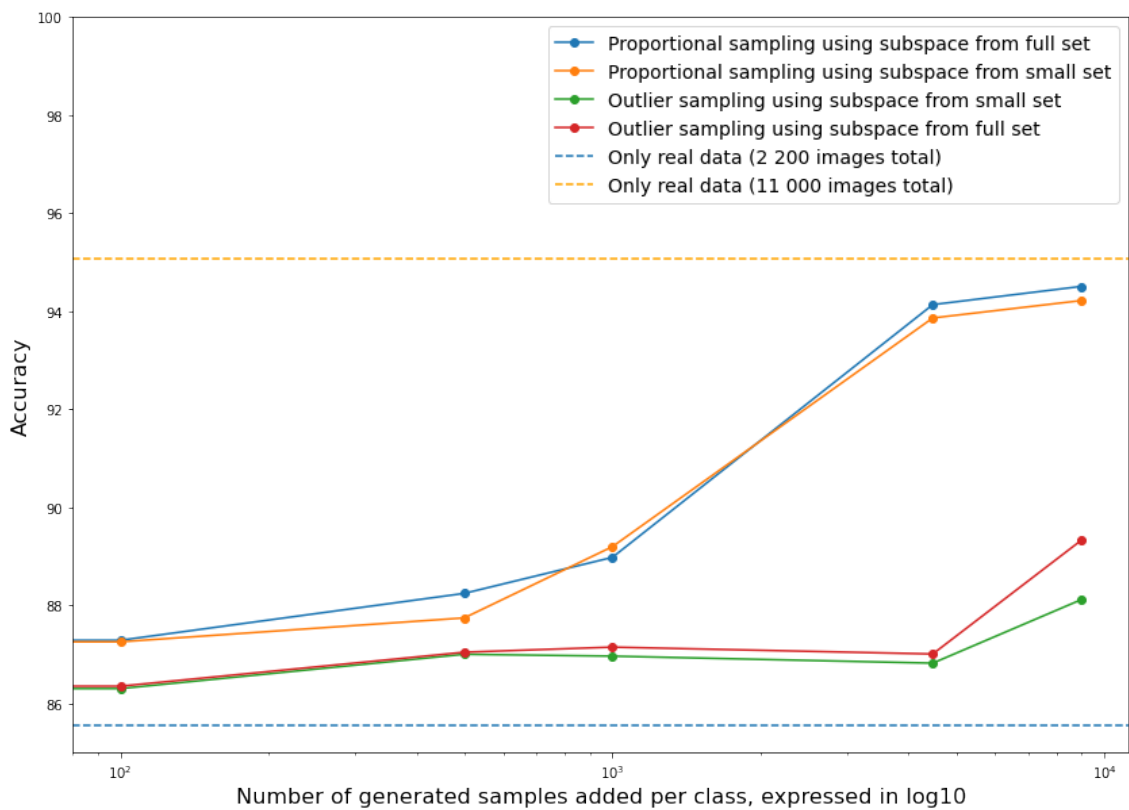


Figure 4.12: Accuracies from augmented training set using images generated by the ReACGAN. 2 200 real images were used.

Table 4.3: Accuracies of classification network for all experiments, expressed in percentages. Each row shows the accuracies when adding the number of synthetic samples to the real images in the training set shown in the second column. Each column shows the accuracies when using the number of real images in the training set shown in the second row. The highest accuracy for each GAN and each number of real images used is bolded.

		ReACGAN		ACGAN	
		1000	200	1000	200
Only real samples		95.06	85.57	95.06	85.57
Random sampling	100	94.95	86.86	95.11	86.27
	500	95.19	87.74	95.74	87.02
	1000	97.39	88.75	97.33	88.48
	4480	98.43	94.26	98.34	94.61
	8960	98.94	95.50	99.17	95.62
PCA proportional	100	95.10	87.26	95.18	86.57
	500	95.38	87.74	95.56	87.78
	1000	96.66	89.20	97.09	87.92
	4480	98.44	93.86	97.31	93.89
	8960	98.76	94.22	98.85	94.78
PCA outliers	100	94.93	86.30	95.15	86.19
	500	95.42	87.00	95.63	86.95
	1000	95.84	86.96	96.27	87.05
	4480	95.09	86.82	95.66	86.80
	8960	96.43	88.11	97.42	91.25
PCA proportional (full set)	100	95.99	87.29	95.12	86.55
	500	95.38	88.24	95.70	87.47
	1000	97.36	88.98	97.03	88.34
	4480	98.17	94.13	98.04	94.46
	8960	98.59	94.51	98.86	94.15
PCA outliers (full set)	100	94.95	86.35	96.26	86.19
	500	96.75	87.04	96.26	86.95
	1000	96.93	87.15	95.66	87.05
	4480	96.91	87.01	95.86	86.80
	8960	96.74	89.32	97.42	91.25

5

Conclusion

Automating quality control using deep learning, as a part of smart manufacturing, is a method used for reducing both production costs and heavy manual labor. This thesis aimed to improve an object detector, and more specifically the image classification network (the *control network*), used for quality assertion. This is a task that demands high accuracy, since mistakes in the classification task can lead to expensive errors. Data collection and annotation is expensive, however, and the method to approach the general problem in this thesis was to use GANs for generating synthetic data for the object detection network to be trained on. Using synthetic data is a way of reducing the need for real data, but this requires the generated data to be very similar to the real data.

In this project, two GANs were selected to generate synthetic data to be used for training a CNN used for image classification. The real dataset consists of images of digits and letters, cropped from photographs of serial numbers on machine parts on a production line. This data was used for training the GANs, from which a set of synthetic images were generated. The synthetic data was mixed with the real data, and the mixed data was used for training the CNN. The CNN was then evaluated on a test set of real images, and the accuracy was used as the measure of performance. A research question in this thesis was how the size of the real dataset affects the performance of the control network. Given that data is expensive to collect, it is of interest to keep this dataset as small as possible while still maintaining a good performance. To approach this question, two smaller training sets of real images were sampled from the full dataset, in order to simulate a situation where data is limited. The sizes of these subsets were a total of 11 000 images (1 000 per class), and 2 200 images (200 per class) respectively.

Another research question was whether specific sampling techniques could improve the performance of the control network. We argued that this dataset could be represented by a low-dimensional subspace. To use this property for specifically sampling synthetic images that might be hard for the control network to classify, a sampling technique based on subspaces were suggested. By computing principal components using PCA, subspaces were found that spanned these principal components. The reconstruction error can then be used to find data points that are the farthest away in the n -dimensional space from the subspace, which intuitively should correspond to hard cases. In this thesis, three different sampling techniques were compared to see which achieved the highest accuracy: random sampling, proportional subspace sampling, and outlier sampling. Random sampling means that synthetic images were randomly sampled from each class. Proportional subspace sampling means sampling partly from hard cases found using subspaces, and partly from non-hard cases.

Finally, outlier sampling refers to only sampling hard cases found by subspaces. In a series of experiments, we varied the sampling technique as well as the number of synthetic images that were added to the mixed dataset. The results show that adding synthetic data to the dataset has a positive effect on performance. The increase of performance was highest when the GANs were trained on the smallest training set of 2 200 real images. When adding a total of 98 560 synthetic images (8 960 per class), the highest achieved accuracy was 95.62%, which can be compared to 85.57%, which is the accuracy achieved when no synthetic images were added to the training set. The best performance overall was achieved when adding 98 560 synthetic images from the ACGAN trained on 11 000 real images, with an achieved accuracy of 99.17%. When trained on this set of real images with no added synthetic data, the accuracy was 95.06%. These results suggest that using synthetic data can improve the performance of an image classification network even when using a very limited dataset of real images, which is desirable given that real data is expensive and sometimes difficult to collect and annotate. When using a slightly larger dataset of real images, an accuracy over 99% can be achieved, which is also desirable since the demand of performance is high in manufacturing processes.

The results from the experiments also show that while adding more synthetic data has a positive effect, this has diminishing returns. For instance, the relative increase of accuracy was lower when increasing the number of synthetic images from 4 480 per class to 8 960, compared to the relative increase of accuracy when increasing the synthetic sample size from 1 000 per class to 4 480. While this was beyond the scope of the thesis, it seems plausible that there is an upper limit to how much synthetic data can contribute with. Further, synthetic data did not seem to be a perfect substitute for real data. Using 200 real images + 8 960 synthetic images per class achieved higher accuracy than 1 000 real images + 0 synthetic images, but did not surpass the accuracy of using 1 000 real + 1 000 synthetic images.

The random sampling technique was in general better than both the subspace sampling techniques, in the sense that it achieved the highest accuracy in all experiment setups. The difference in performance was higher when adding more synthetic samples, and also when using the small subset of real data compared to the larger. It is, however, worth looking at when this method was successful. In Table 4.3, it can be observed that proportional subspace sampling performer better than random sampling when the number of added synthetic samples is low. This would suggest that using subspace sampling has a positive effect, but this effect disappears when the number of added synthetic samples increases. Hence, there is indeed a potential for using subspace based sampling methods when the data can be well-represented by a low-dimensional subspace. Since most data in smart manufacturing likely shares this property, this could be a beneficial method to investigate further.

5.1 Future work

This study has shown that using GANs for augmenting a dataset, characterized as being well-represented by a low-dimensional subspace, can increase the performance of an image classification network. It has also been shown that under some conditions, using subspaces for sampling hard cases can further increase the performance.

However, it is yet to be shown how well this method can be generalized, i.e., if being well-represented by a low-dimensional subspace is a necessary condition for this to work. A recommendation for future studies is therefore to further investigate the properties of the data, and its relation to different sampling techniques of the synthetic data.

The behavior of the proportional sampling indicates that sampling conditional on the subspace might indeed have a positive effect on performance, but it is beyond the scope of this study to draw further conclusions from this. A suggestion for future studies is to further investigate the relationship between the size of the real dataset used, how many synthetic images are added, and the effect of the subspace sampling on this. A hypothesis is that smaller datasets results in subspaces with more bias, which has a negative effect if the sampling is based on these subspaces.

In this thesis, linear PCA was used to find subspaces. It is not guaranteed that using linear subspaces is the best method to sample hard cases, since non-linear methods might find subspaces that provide better representations of the data.

Bibliography

- [1] Heiner Lasi et al. “Industry 4.0”. In: *Business & information systems engineering* 6.4 (2014), pp. 239–242.
- [2] Alberto Diez-Olivan et al. “Data fusion and machine learning for industrial prognosis: Trends and perspectives towards Industry 4.0”. In: *Information Fusion* 50 (2019), pp. 92–111.
- [3] Angelos Angelopoulos et al. “Tackling faults in the industry 4.0 era—a survey of machine-learning solutions and key aspects”. In: *Sensors* 20.1 (2019), p. 109.
- [4] Jinjiang Wang et al. “Deep learning for smart manufacturing: Methods and applications”. In: *Journal of Manufacturing Systems* 48 (2018). Special Issue on Smart Manufacturing, pp. 144–156. ISSN: 0278-6125. DOI: <https://doi.org/10.1016/j.jmsy.2018.01.003>. URL: <https://www.sciencedirect.com/science/article/pii/S0278612518300037>.
- [5] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems* 27 (2014).
- [6] Antreas Antoniou, Amos Storkey, and Harrison Edwards. “Data augmentation generative adversarial networks”. In: *arXiv preprint arXiv:1711.04340* (2017).
- [7] Esther Robb et al. “Few-shot adaptation of generative adversarial networks”. In: *arXiv preprint arXiv:2010.11943* (2020).
- [8] Bo Dai et al. “Towards diverse and natural image descriptions via a conditional gan”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2970–2979.
- [9] Minhyeok Lee and Junhee Seok. “Controllable generative adversarial network”. In: *Ieee Access* 7 (2019), pp. 28158–28169.
- [10] Ross Girshick et al. “Region-based convolutional networks for accurate object detection and segmentation”. In: *IEEE transactions on pattern analysis and machine intelligence* 38.1 (2015), pp. 142–158.
- [11] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.
- [12] Te Han et al. “A novel adversarial learning framework in deep convolutional neural network for intelligent diagnosis of mechanical faults”. In: *Knowledge-based systems* 165 (2019), pp. 474–487.
- [13] Haodong Lu et al. “GAN-based data augmentation strategy for sensor anomaly detection in industrial robots”. In: *IEEE Sensors Journal* (2021).
- [14] Xin Gao, Fang Deng, and Xianghu Yue. “Data augmentation in fault diagnosis based on the Wasserstein generative adversarial network with gradient penalty”. In: *Neurocomputing* 396 (2020), pp. 487–494.

- [15] Patxi Ortego et al. “Data augmentation for industrial prognosis using generative adversarial networks”. In: *International Conference on Intelligent Data Engineering and Automated Learning*. Springer. 2020, pp. 113–122.
- [16] J Hernavs et al. “Deep learning in industry 4.0—brief overview”. In: *J Prod Eng* 21.2 (2018), pp. 1–5.
- [17] Xianghua Xie. “A review of recent advances in surface defect detection using texture analysis techniques”. In: *ELCVIA: electronic letters on computer vision and image analysis* (2008), pp. 1–22.
- [18] Arnaz Malhi, Ruqiang Yan, and Robert X Gao. “Prognosis of defect propagation based on recurrent neural networks”. In: *IEEE Transactions on Instrumentation and Measurement* 60.3 (2011), pp. 703–711.
- [19] Yuting Wu et al. “Remaining useful life estimation of engineered systems using vanilla LSTM neural networks”. In: *Neurocomputing* 275 (2018), pp. 167–179.
- [20] Charbel El Hachem et al. “Automation of quality control in the automotive industry using deep learning algorithms”. In: *2021 International Conference on Computer, Control and Robotics (ICCCR)*. IEEE. 2021, pp. 123–127.
- [21] Yunfeng Lin, Jiangbei Li, and Hanjing Wang. “Dcnn-gan: Reconstructing realistic image from fmri”. In: *2019 16th International Conference on Machine Vision Applications (MVA)*. IEEE. 2019, pp. 1–6.
- [22] Veit Sandfort et al. “Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks”. In: *Scientific reports* 9.1 (2019), pp. 1–9.
- [23] Abdul Waheed et al. “Covidgan: data augmentation using auxiliary classifier gan for improved covid-19 detection”. In: *Ieee Access* 8 (2020), pp. 91916–91923.
- [24] Siyu Shao, Pu Wang, and Ruqiang Yan. “Generative adversarial networks for data augmentation in machine fault diagnosis”. In: *Computers in Industry* 106 (2019), pp. 85–93.
- [25] Shuanlong Niu et al. “Defect image sample generation with GAN for improving defect recognition”. In: *IEEE Transactions on Automation Science and Engineering* 17.3 (2020), pp. 1611–1622.
- [26] Thomas Pinetz, Johannes Ruisz, and Daniel Soukup. “Actual Impact of GAN Augmentation on CNN Classification Performance.” In: *ICPRAM*. 2019, pp. 15–23.
- [27] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [28] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.
- [29] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).
- [30] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
- [31] Mehdi Mirza and Simon Osindero. “Conditional generative adversarial nets”. In: *arXiv preprint arXiv:1411.1784* (2014).

-
- [32] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks”. In: *arXiv preprint arXiv:1511.06434* (2015).
- [33] Jie Gui et al. “A review on generative adversarial networks: Algorithms, theory, and applications”. In: *IEEE Transactions on Knowledge and Data Engineering* (2021).
- [34] Zhengwei Wang, Qi She, and Tomas E Ward. “Generative adversarial networks in computer vision: A survey and taxonomy”. In: *ACM Computing Surveys (CSUR)* 54.2 (2021), pp. 1–38.
- [35] Sung-Wook Park et al. “Review on generative adversarial networks: Focusing on computer vision and its applications”. In: *Electronics* 10.10 (2021), p. 1216.
- [36] Martin Arjovsky and Léon Bottou. “Towards principled methods for training generative adversarial networks”. In: *arXiv preprint arXiv:1701.04862* (2017).
- [37] Martin Arjovsky, Soumith Chintala, and Léon Bottou. “Wasserstein generative adversarial networks”. In: *International conference on machine learning*. PMLR. 2017, pp. 214–223.
- [38] Ishaan Gulrajani et al. “Improved training of wasserstein gans”. In: *Advances in neural information processing systems* 30 (2017).
- [39] Takeru Miyato et al. “Spectral normalization for generative adversarial networks”. In: *arXiv preprint arXiv:1802.05957* (2018).
- [40] Augustus Odena, Christopher Olah, and Jonathon Shlens. “Conditional image synthesis with auxiliary classifier gans”. In: *International conference on machine learning*. PMLR. 2017, pp. 2642–2651.
- [41] Mingming Gong et al. “Twin auxiliary classifiers gan”. In: *Advances in neural information processing systems* 32 (2019).
- [42] Liang Hou et al. “Conditional GANs with Auxiliary Discriminative Classifier”. In: *arXiv preprint arXiv:2107.10060* (2021).
- [43] Takeru Miyato and Masanori Koyama. “cGANs with projection discriminator”. In: *arXiv preprint arXiv:1802.05637* (2018).
- [44] Han Zhang et al. “Self-attention generative adversarial networks”. In: *International conference on machine learning*. PMLR. 2019, pp. 7354–7363.
- [45] Andrew Brock, Jeff Donahue, and Karen Simonyan. “Large scale GAN training for high fidelity natural image synthesis”. In: *arXiv preprint arXiv:1809.11096* (2018).
- [46] Minguk Kang et al. “Rebooting ACGAN: Auxiliary Classifier GANs with Stable Training”. In: *Advances in Neural Information Processing Systems* 34 (2021).
- [47] Minguk Kang and Jaesik Park. “Contragan: Contrastive learning for conditional image generation”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 21357–21369.
- [48] Ali Borji. “Pros and cons of gan evaluation measures”. In: *Computer Vision and Image Understanding* 179 (2019), pp. 41–65.
- [49] Tim Salimans et al. “Improved techniques for training gans”. In: *Advances in neural information processing systems* 29 (2016).

- [50] Martin Heusel et al. “Gans trained by a two time-scale update rule converge to a local nash equilibrium”. In: *Advances in neural information processing systems* 30 (2017).
- [51] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. “How good is my GAN?” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 213–229.
- [52] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- [53] *PyTorch-StudioGAN*. <https://github.com/POSTECH-CVLab/PyTorch-StudioGAN/>. 2022-05-06.
- [54] Giovanni Mariani et al. “Bagan: Data augmentation with balancing gan”. In: *arXiv preprint arXiv:1803.09655* (2018).
- [55] Adamu Ali-Gombe and Eyad Elyan. “MFC-GAN: class-imbalanced dataset classification using multiple fake class generative adversarial network”. In: *Neurocomputing* 361 (2019), pp. 212–221.
- [56] Shengyu Zhao et al. “Differentiable Augmentation for Data-Efficient GAN Training”. In: *CoRR* abs/2006.10738 (2020). arXiv: 2006.10738. URL: <https://arxiv.org/abs/2006.10738>.