Augmenting High-Dimensional Data with Deep Generative Models

MÅRTEN NILSSON
Augmenting High-Dimensional Data with Deep Generative Models

MÅRTEN NILSSON
Abstract

Data augmentation is a technique that can be performed in various ways to improve the training of discriminative models. The recent developments in deep generative models offer new ways of augmenting existing data sets. In this thesis, a framework for augmenting annotated data sets with deep generative models is proposed together with a method for quantitatively evaluating the quality of the generated data sets. Using this framework, two data sets for pupil localization was generated with different generative models, including both well-established models and a novel model proposed for this purpose. The unique model was shown both qualitatively and quantitatively to generate the best data sets. A set of smaller experiments on standard data sets also revealed cases where this generative model could improve the performance of an existing discriminative model. The results indicate that generative models can be used to augment or replace existing data sets when training discriminative models.
**Sammanfattning**

# Contents

1 Introduction ............................................ 1
  1.1 Problem statement .................................... 2
  1.2 Scope and objectives .................................. 2
  1.3 Societal and ethical considerations ................. 2
  1.4 Thesis overview ...................................... 4

2 Background ............................................. 5
  2.1 Concepts and terminology ............................. 5
  2.2 Generative models ..................................... 5
  2.3 Variational Autoencoders ............................. 6
  2.4 Generative Adversarial Networks ....................... 9
    2.4.1 Theoretical properties of GANs ................. 10
    2.4.2 Practical issues ................................ 12
    2.4.3 Different types of GANs ......................... 13
    2.4.4 VAE and GAN hybrid models ..................... 17
    2.4.5 Evaluation measures ............................. 17
    2.4.6 Evaluation issues ............................... 18

3 Methods .................................................. 20
  3.1 Pupil localization .................................... 20
  3.2 Synthetic Data ....................................... 22
  3.3 Real World Data ...................................... 23
  3.4 Network architectures ................................ 24
  3.5 Variational Autoencoders ............................. 26
  3.6 Wasserstein GAN ..................................... 28
  3.7 Progressive GAN ..................................... 29
  3.8 Autoencoding GAN .................................... 29
  3.9 Training setup ...................................... 31
CONTENTS

4 Results
   4.1 Qualitative Results ................. 33
   4.2 Quantitative Results ............... 35
   4.3 Augmenting data for classification .... 38

5 Discussion 40

6 Conclusion
   6.1 Future work ....................... 44

Bibliography 46
Chapter 1

Introduction

Generative models are essential for a wide range of applications within Data Science and Machine Learning. Unlike discriminative models, generative models are capable of reproducing the phenomenon they are modelling, enabling them to be used for synthetic data generation. This has been shown to be successful for low dimensional relational data; as example Patki, Wedge, and Veeramachaneni [29] showed that generative models could be used as a drop in replacement for a relational database. However, achieving similar results on large high dimensional datasets has proved to be more challenging.

Until recently, there has been little progress in developing generative models that can scale beyond small low-dimensional datasets. Therefore, the most refined generative models of today are intractable to apply on large high-dimensional datasets. The past few years have seen major advances in deep learning. These advances allowed a new class of scalable generative models to emerge, which is today called deep generative models. Of these models, a significant amount of attention have been directed at Generative Adversarial Networks (GANs) which are capable of generating highly realistic samples of complicated data distributions.

Generative Adversarial Networks were first introduced by Goodfellow et al. [9] in 2014 as a method of generating realistic samples of a data distribution. The state-of-the-art GANs of today are capable of generating diverse sets of high resolution images [15]. Adversarial methods have also helped improve existing datasets for gaze estima-
A considerable amount of literature has been published on how to train GANs. Most of these studies consider convergence issues related to training GANs and the quality of the generated samples. A search of the literature revealed few studies which attempt to train new networks on the generated samples. The aim of this project is to advance the understanding of these possibilities.

1.1 Problem statement

This thesis intends to determine the extent to which deep generative models can act as a drop-in replacement for existing datasets. The question asked is

Can deep generative models be applied to generate synthetic datasets that can be used to boost the performance of existing discriminative models?

1.2 Scope and objectives

This study is unable to encompass the entire field of deep generative models as well as all possible data sources. Therefore, this study examines the previously stated question through a case study within semantic segmentation for pupil regression. Special emphasis is put on the task of generating synthetic datasets for this task through Generative Adversarial Networks.

1.3 Societal and ethical considerations

In the new global economy, extracting knowledge from data has become a central issue for a wide range of industries. Generative models play an important role in this process as these models are the most generic methods for finding hidden structures and relationships in
large datasets. As more industries move towards a data driven business model, the scalability and applicability of certain generative models in new problem domains are increasingly interesting. The outcomes of this project are especially relevant to industries that utilize large sets of images for supervised learning as it provides valuable insights regarding an approach for learning hidden structures in this type of data.

Another source of interest from the economic perspective emerges from the fact that manually annotated data is expensive and requires large amounts of human labour. Increasing the effectiveness of the existing data might reduce the need for extending the size of the annotated datasets in some cases.

In the case of very large databases, both economical and ecological benefits exist. A generative model requires only a small fraction of the storage space that the full dataset needs. Therefore, replacing databases with generative models has the potential of saving vast amounts of resources and thus being a both cheaper and more ecological alternative to huge data stores when applicable. Though the relevance of this can be discussed since usually storage is not the main ecological problem and the extra computations required to run the generative model could weigh over the benefits of it.

This approach also comes with an ethical issue. When the data consists of personal information, data belonging to a specific individual can easily be removed from a conventional dataset. However, there is no obvious way to delete the impact of that data onto a trained generative model. It could be argued that by training the generative model, the data have already been anonymized. However, the extent to which this is true depends on the specific model. There are many aspects to consider regarding personal integrity and generative modelling, and a full discussion is well beyond the scope of this project. Due to this and the novelty of the methods, they might provide a loophole to circumvent several data protection laws. This risk is assumed to be quite low given the current state-of-the-art, allowing this thesis to be published but any further incremental research within the field should be aware of it.
1.4 Thesis overview

Chapter 2 provides an overview of Generative Adversarial Networks and related methods. The notation used throughout the thesis is established in Section 2.1. In Chapter 3, the methods that have been used to answer the problem statement are presented. The experimental results are presented in Chapter 4 and discussed in Chapter 5. In Chapter 6, the final conclusion is stated and further work is discussed.
Chapter 2

Background

This chapter begins by establishing the notation and the concepts encountered in this chapter. Thereafter, a brief overview of generative models is provided. This is followed by more detailed descriptions of the families of generative models considered in this project. Special emphasis is put on Generative Adversarial Networks and its variants.

2.1 Concepts and terminology

This chapter deals with high dimensional spaces, stochastic variables, sets, vectors and probability distributions. Spaces and sets are written in a calligraphic style (\( \mathcal{Z} \)), stochastic variables (and vectors) are written as uppercase letters (\( \mathbf{Z} \)), vectors are written as lowercase letters (\( z \)) and the letter \( p \) is reserved for probabilities and probability distributions.

2.2 Generative models

In this thesis, the term generative model refers to models that explicitly or implicitly represent a data distribution, thereby allowing sampling from the modeled distribution. This is different from discriminative
Figure 2.1: The directed graphical model representing the dependency between the latent variable $Z$ and the data variable $X$ in GANs. This is the same dependency structure as in GMMs and VAEs. The color coding is consistent with that of Bishop [3], where colored cells represent latent variables and non-colored cells are observable variables. The arrow represents a causal dependency.

models which only model information that can be extracted from the data.

This thesis focuses on parametric generative models which can be formulated as instances of probabilistic graphical models [3]. Examples of these include Gaussian Mixture Models (GMMs), Hidden Markov models (HMMs), Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs). The focus of this work is directed towards models that are applicable on large, high-dimensional and complex datasets. The known models with these properties are GANs and VAEs. Both of these models assume a dependency between a latent variable $Z$ and an observed variable $X$ as shown in Figure 2.1. The rest of this chapter provides a more in-depth description of these models.

2.3 Variational Autoencoders

A popular family of generative models are Variational Autoencoders (VAEs). Variational Autoencoders were first introduced by Kingma and Welling [16] as a scalable approach for stochastic variational inference. VAEs are trained using the Auto Encoding Variational Bayes (AEVB) algorithm proposed in the same article.
These models consist of a dataset $S$ with i.i.d samples and a continuous latent variable $Z$ in a latent space $Z$. The dataset is viewed as a set of outcomes of a random variable $S$. The prior $p(Z)$ and likelihood $p(S|Z)$ are assumed to come from parametric families of distributions and the posterior $p(Z|S)$ is assumed to be intractable and is modeled by some parametric distribution. The parameters of the prior and likelihood distribution are commonly denoted as the generative model parameters $\theta$. The probabilities in the context of these models are commonly denoted as $p_\theta(Z)$, $p_\theta(X|Z)$ and $p_\theta(Z|X)$ to explicitly expose the model parameters. Since the posterior $p_\theta(Z|X)$ is assumed intractable it is estimated with a variational approximation, commonly denoted $q_\phi(Z|X)$. From the deep learning perspective, $q_\phi(Z|X)$ is commonly referred to as the encoder, and $p_\theta(X|Z)$ is commonly referred to as the decoder. This is illustrated in figure 2.2.

The AEVB algorithm is applied to learn the generative model parameters and infer the optimal variational parameters. It is a type of approximate variational inference and consists of a gradient based optimization of an estimator of the variational lower bound of the marginal likelihood of the data under the current model. In the original article the authors propose the two estimators based on Monte Carlo esti-
mates of expectations:
\[
\mathcal{L}^A = \frac{1}{L} \sum_{l=1}^{L} \log p_\theta(X = x^{(i)}, Z = z^{(i,l)}) - \log q_\phi(Z = z^{(i,l)}|X = x^{(i)}),
\]
\[
\mathcal{L}^B = -D_{KL}(q_\phi(Z|X = x^{(i)})||p_\theta(Z = z^{(i,l)})) + \frac{1}{L} \sum_{l=1}^{L} \log p_\theta(X = x^{(i)}, Z = z^{(i,l)}).
\]
(2.1)

**Algorithm 1** Example of a training scheme for Variational Autoencoders. This algorithm assumes that the encoder posterior probabilities are modeled as a multivariate gaussian distribution with diagonal covariance and the decoder posterior is an isotropic multivariate laplace distribution with known variance. Enc(x) and Dec(z) correspond to the encoder and decoder functions. The encoder outputs a vector of mean values and standard deviations for the multivariate gaussian, whereas the decoder only outputs the mean values. \( \lambda \) and \( \nu \) are taken as hyperparameters.

1: for number of iterations do
2: Sample minibatch of data points \( \{x_1, \ldots, x_m\} \sim p_S(x) \).
3: Sample minibatch of latent points from a multivariate unit Gaussian \( \{z_1, \ldots, z_m\} \sim N(0, I) \).
4: Assign parameters for the latent posterior: \( \mu_i, \sigma_i \leftarrow Enc(x_i) \)
5: Sample from the posterior and decode it back to the data space:
   \( \hat{x_i} = Dec(\mu_i + z_i \odot \sigma_i) \), where \( \odot \) represents the Hadamard product.
6: Update the model parameters \( \theta \) with the following gradient:
   \[
   \nabla_\theta \left( \frac{1}{m} \sum_{i=1}^{m} \left\{ |\hat{x_i} - x|^1 + \nu (\ln(\sigma_i^2) - \mu_i^2 - \sigma_i^2) \right\} \right).
   \]
7: end for

VAEs can be trained by maximizing either of these estimates using common gradient based methods, an example of this is shown in Algorithm 1. This approach requires sampling from the posterior in the latent space, which is a non-differentiable operation. However, as described in the algorithm, it is possible to sample from a \( N(0, I) \) distribution instead and simply transform this to the posterior using differentiable operations. Kingma and Welling [16] describe the generic
2.4 Generative Adversarial Networks

Generative Adversarial Networks (GANs) are a family of generative models proposed by Goodfellow et al. [9]. The framework for training GANs consists of a dataset $S$ with elements in a domain $\mathcal{X}$, a latent space $Z$ and two neural networks, a generator $G(z)$ and a discriminator $D(x)$ (Figure 2.3). The generator maps elements of $Z$ to $\mathcal{X}$, that is $G : Z \rightarrow \mathcal{X}$. The discriminator is a binary classifier $D : \mathcal{X} \rightarrow [0, 1]$.

The objective of the discriminator is to classify elements in $\mathcal{X}$ as either members or not members of $S$. Members of $S$ are usually referred to as real samples since they are part of the dataset, and non-members are referred to as fake samples. The objective of the generator is to fool the discriminator by mapping elements in $Z$ to the subspace of $\mathcal{X}$ that is classified as real.

Figure 2.3: The main components of GANs

diagram of GANs with $z \sim p_Z \rightarrow G(z) \rightarrow x_f \sim p_G$

$z \sim p_Z$

$G(z)$

$x_f \sim p_G$

$D(x)$

Fake

$x \sim p_S$

Real

The illustrations in [5] are an excellent source of intuition as to why this works.
The generator can be viewed as representing a probability distribution $p_G$ on $\mathcal{X}$. By introducing a latent random variable $Z$ with the probability distribution $p_Z$ on $Z$, the generator probability distribution can be expressed as $p_G(x) = p_Z(G^{-1}(x))$.

In practice $G^{-1}$ is intractable to compute, and therefore explicit probabilities are seldom acquired through GANs. However, this formulation allows sampling from the learned distribution as $G(z), z \sim p_Z$, is straightforward to compute.

The goal of training GANs is to implicitly learn a probability distribution over the data. By viewing the elements of $\mathcal{S}$ as outcomes of a random variable $S$ with probability distribution $p_S$ the goal of the generator is to approximate this distribution with $p_G$, and the goal of the discriminator is to discriminate between these distributions. Using this formulation the objective of the GAN training can be formulated as a minimax game

$$\min_{G} \max_{D} V^D(G, D) \quad (2.2)$$

where $V^D(G, D)$ is the discriminator value function from [8],

$$V^D(G, D) = \frac{1}{2} \mathbb{E}_{x \sim p_S, z \sim p_Z} [\log(D(x)) + \log(1 - D(G(z)))] . \quad (2.3)$$

Since both the generator and discriminator are differentiable functions, (2.3) can be optimized using standard gradient based optimization schemes such as RMSprop [39] or Adam [17]. Typically, the networks are updated in an alternating fashion where the parameters of one of the networks are frozen while updating the other network. The expectations in (2.3) are typically estimated using minibatches as

$$\mathbb{E}_{x \sim p(x)} [f(x)] \approx \frac{1}{m} \sum_{i=1}^{m} f(x_i) \quad (2.4)$$

where $x_i$ is sampled from $p(x)$. This process is described in an algorithmic fashion in Algorithm 2.

### 2.4.1 Theoretical properties of GANs

In the original paper (Goodfellow et al. [9]) a couple of theoretical properties of GANs are formulated and proved. These proofs assume
Algorithm 2 Training scheme for Generative Adversarial Networks using minibatch stochastic gradient descent and $n_D$ discriminator updates per generator update. $\theta_D$ denotes the discriminator parameters and $\theta_G$ denotes the generator parameters. $\lambda$ is the learning rate.

1: for number of iterations do 
2:     for $n_D$ steps do 
3:         Sample minibatch of fake data $\{x_{f1}, ..., x_{fm}\} \sim p_G(x)$. 
4:         Sample minibatch of real data $\{x_1, ..., x_m\} \sim p_S(x)$. 
5:         Update discriminator parameters with the estimated gradient of the value function 
          \[ \theta_D \leftarrow \theta_D + \lambda \nabla \theta_D \left( \frac{1}{m} \sum_{i=1}^{m} \log(D(x_i)) + \log(1 - D(x_{fi})) \right) . \]
6:     end for 
7:     Sample minibatch of fake images $\{x_{f1}, ..., x_{fm}\} \sim p_G(x)$. 
8:     Update generator parameters with the estimated gradient of the negative value function 
          \[ \theta_G \leftarrow \theta_G - \lambda \nabla \theta_G \left( \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(x_{fi})) \right) . \]
9: end for
that both the generator and discriminator have infinite capacity, meaning that they can model all possible mappings between the domains.

It is proven that given a fixed generator $G$ the optimal discriminator is

$$D(x) = \frac{p_S(X = x)}{p_S(X = x) + p_G(X = x)}.$$ (2.5)

A straightforward implication of this is that when $G$ is optimal, the optimal $D$ maps all possible inputs onto the value 0.5. In other words

$$p_G \overset{d}{=} p_S \implies D(x) = 0.5 \forall x \sim p_G, p_S$$ (2.6)

if $D$ and $G$ are optimal. It is also proved in the original paper that given an optimal discriminator there exists a global minimum of the discriminator value function and it is reached if and only if $p_G \overset{d}{=} p_S$.

Finally, convergence of the training algorithm (Algorithm 2) is also proven.

This means that one can observe convergence when training GANs by following the development of the discriminator output. When it saturates at 0.5 for all generator samples and real samples, the algorithm has converged. In practice, it is difficult to get to this point with complex sets of high-dimensional data.

### 2.4.2 Practical issues

The theoretical properties indicate that GANs are a sensible method for implicitly learning probability distributions. However, when implementing these models in practice some issues emerge from using models of finite capacity.

The two-player minimax formulation of the game results in a notoriously difficult training scenario. The update of one player may undo progress of the other player instead of moving towards the Nash equilibrium. Therefore, regular gradient descent is not guaranteed to converge but can just as easily get stuck in a stale orbit (Salimans et al. [33]). It is also common that the generator learns to represent a subset of the original data realistically while ignoring the vast majority of the

---

1Nash equilibrium is the point where neither the discriminator nor the generator can improve their performance by only updating their own parameters
data points, thereby collapsing to a specific "mode" of the data. This phenomenon is known as mode collapse.

Another issue encountered when implementing GANs emerges from the limitations of representing a continuous distribution with a finite dataset. Even though the theoretical results guarantee that \( p_G \) converges to \( p_S \), \( p_S \) is not uniquely determined. In fact, \( p_S \) is represented by sampling from a uniform discrete distribution over the dataset \( S \).

Therefore, the optimal generator would theoretically collapse to this discrete distribution. This results in a generator incapable of generalizing to new data points. On the other hand, should this behaviour be encountered in practice with limited capacity, the model would still be usable as a compression of the data for applications that only require sampling from the dataset. An example for this is using the generator as a proxy for the dataset during training or evaluation of another neural network.

### 2.4.3 Different types of GANs

Since the introduction of GANs, several variations and extensions have been proposed to overcome the convergence problems and stability issues of the models. The most relevant variations for this work are outlined in this section.

**Non-saturating GAN**

An issue with training GANs on the original discriminator cost function (2.3) is that the adversarial term \( \log(1 - D(G(Z))) \) saturates if the discriminator is too confident, meaning that it maps fake data to values close to zero. This can prevent the generator from learning in the beginning when the generator output are quite dissimilar to the original data. In the original article it is suggested to train the generator to minimize \( -\log(D(G(Z))) \) instead. This cost is sometimes referred to as the non saturating cost function.

The non saturating cost function preserves the fixed point dynamics of the original GAN formulation while at the same time permitting the generator to learn when the discriminator is confident. Due to this
property, non saturating GANs are the implementation used for the experiments in the original article.

**DCGAN**

In an attempt to bring the success of Convolutional Neural Networks (CNNs) from supervised to unsupervised learning, Radford, Metz, and Chintala [30] presented a set of architectural guidelines for designing GANs for image generation. They showed that their architectural guidelines resulted in improved quality of the generated samples compared to the state of the art at the time. The guidelines they presented are:

- Use strided convolutions instead of pooling in discriminator. Upsample with fractional-strided convolutions in generator.
- Avoid fully connected hidden layers in deeper architectures.
- Use ReLU as an activation function in the generator and LeakyReLU [22] in the discriminator.

Subsequent research have shown alternatives to some of these guidelines. For example, batch normalization results in an unwanted correlation between the generated samples in a minibatch [33]. To circumvent the issues of Batch Normalization, tweaked versions of the original formulations have been proposed such as Reference Batch Normalization and Virtual Batch Normalization. Moreover, it has been increasingly common to apply alternative normalization schemes, such as performing pixelwise normalization over individual feature maps [15], normalizing weight tensors with their largest singular value [25] and decoupling the norm and direction of weight tensors [32, 41].

In recent GAN formulations, it has been increasingly common to apply the LeakyReLU activation function in both networks and not just in the discriminator.
CHAPTER 2. BACKGROUND

Conditional GAN

A limitation of the GAN framework is that it does not allow external information such as image annotations to be modeled. A straightforward extension of the original framework is to let the generator model the conditional distribution $p_S(X|Y)$, where $Y$ represents the additional information. This was proposed in the original article, and was later formally introduced and tested on MNIST\(^2\) by Mirza and Osindero [24]. The conditional distributions are modeled by introducing the additional information as input to both the discriminator and the generator.

Auxiliary Classifier GAN

The Auxiliary Classifier GAN, proposed by Odena, Olah, and Shlens [28], is another approach of modelling conditional distributions with GANs. As in the conditional GAN, the generator receives an outcome from both the latent variable $Z$ and the conditional variable $Y$ and generates a sample from this information. The conditional variable is assumed to be a class label, but the approach is easily extendible. In contrast to the generator, the discriminator only receives the generated sample without any class label. Instead, the discriminator simultaneously attempts to discriminate between real and fake images, as well as producing class probabilities.

In this framework, the generator is trained to simultaneously produce data points that yields confident correct class predictions at the same time as they are indistinguishable from the correct data points. This approach have been shown to generate images with state of the art Inception Score\(^3\) values [28].

Wasserstein GAN

A popular alternative to the original GAN is the Wasserstein GAN. It was proposed in 2017 by Arjovsky, Chintala, and Bottou [2]. The

\(^2\)LeCun, Cortes, and Burges [20]

\(^3\)This concept is explained in the next section
distinguishing feature of the Wasserstein GAN is that instead of mini-
mimizing the Jensen-Shannon divergence it minimizes the Wasserstein-1,
or Earth-Mover EM, distance

\[
W(p_S, p_G) = \inf_{\gamma = \Pi(p_S, p_G)} \mathbb{E}_{(x,y) \sim \gamma} [|x - y|],
\]

(2.7)

where \( \Pi(p_s, p_g) \) is the set of all joint distributions with \( p_S \) and \( p_G \) as marginals. Arjovsky, Chintala, and Bottou [2] illustrated that this is

equivalent to minimizing the cost function

\[
W^{(D)}(D, G) = \mathbb{E}_{x \sim p_S}[D(x)] - \mathbb{E}_{z \sim p_Z}[D(G(z))],
\]

(2.8)

under the constraint that \( D \) is Lipschitz smooth. Arjovsky, Chintala, and Bottou [2] enforced Lipschitz smoothness by clipping the weights of the discriminator. In subsequent works more sophisticated methods

have been proposed such as gradient penalty [10], weight normalization [32] and spectral normalization [25].

**Progressive GAN**

Karras et al. [15] proposed a novel training methodology for GANs
to address the stability and variability issues with these models when
trained to generate images. The proposed methodology concerns the
network architectures during training and can be combined with many
of the other GAN variants seamlessly. Instead of training fixed-size
discriminators and generators, it is proposed to progressively grow
their size during training by gradually increasing the resolution of
the training data. Since training on higher resolution images is much
more difficult than on low resolution images, this method conceptually acts as iteratively reusing a simpler network as a sensible initial-
ization for the next higher-resolution network. It can also be viewed
as a multi-step transfer learning process. This approach demonstrated
high-quality diverse fake samples on a 1028 × 1028 version of the CelebA
dataset.

Besides the progressive growing of GANs, Karras et al. [15] also sug-
gest two improvements that can be applied to other variants as well.
They propose weight equalization to address the issue of training deep
networks with weights that have different dynamic range. They pro-
pose using a pixelwise normalization of the feature maps instead of
batch normalization to prevent escalation of signal magnitudes between the networks.

2.4.4 VAE and GAN hybrid models

Both VAEs and GANs implicitly model the data distribution $p_S(x)$ by introducing a latent stochastic variable $Z$. VAEs model the posterior of the latent variable and learn by using the KL divergence on this posterior as well as the reconstruction loss of the data after sampling. GANs on the other hand directly generate samples from the prior of the latent variable and use the discriminator network as an adversarial loss function on the generated samples.

The similarity of these approaches allows different methods of combining the benefits of both approaches. Perhaps the most straightforward approach of training a VAE with an adversarial loss function was proposed by Larsen, Sønderby, and Winther [19]. The resulting model generated sharp realistic-looking samples and allowed meaningful arithmetic to be performed in the latent space.

2.4.5 Evaluation measures

The goal of GANs and VAEs is to model a target distribution and they are often trained on high-dimensional datasets of high complexity. Due to the nature of this setup and the observation that explicit probabilities are unobtainable for both the true data distribution and the modeled distribution, these models are difficult to evaluate. Two popular quantitative measures for evaluating the performance of GANs are the Inception Score (IS) [33] and the Fréchet Inception Distance (FID) scores [13].

The Inception Score metric was defined as an alternative to having human annotators assess the quality of visual samples. It is defined as

$$\exp \left( E \left[ D_{KL} \left( p(Y|X) || p(Y) \right) \right] \right)$$

(2.9)

where $p(Y|X)$ is the class conditional probability distribution obtained through a pretrained Inception network [38], $p(Y)$ is obtained through marginalization and $D_{KL}$ is the Kullback-Leibler Divergence.
The Fréchet Inception Distance metric is a proposed improvement over the Inception Score. Unlike the Inception Score, the Fréchet Inception Distance metric measures the distance between the learned data distribution and the true distribution. It does so by computing the Frechet distance between Inception encodings of the true data distribution and the fake data distribution.

These measures have been shown to correspond to increased quality and diversity of generated samples, but they still suffer from the limitations imposed by the absence of explicit probability distributions. For example, Shu, Bui, and Ermon [36] showed that the ACGAN generator is biased towards learning to generate images that can easily be classified. This means that the training doesn’t necessarily converge towards $p_G \overset{d}{=} p_S$. This bias was both theoretically established and experimentally verified by training an ACGAN on MNIST and observing that the generator completely avoids generating ones with serifs despite the high frequency of serifed ones in the dataset. This skew in data distribution is not quantitatively apparent as the model was shown to still result in high IS values. In fact the ACGAN model obtained higher scores than the original dataset. Similar examples for the FID have not been discovered during the course of this project.

In specific applications, the quality of the generated samples can be evaluated indirectly based on the application of the data. Shrivastava et al. [35] used this approach to compare the quality of a generated dataset against the real data alone for the task of gaze estimation. This approach does not guarantee correspondence between visually pleasing samples and generated samples and does not allow the results to be compared with similar GANs in a different domain. However, it is appealing in the sense that it brings the quantitative evaluation of the models closer to the targeted application.

### 2.4.6 Evaluation issues

Besides the difficulties in finding quantitative evaluation measures, GANs are very sensitive to hyperparameter settings and it is difficult to assess if improved results are caused by better hyperparameters or improved methods. This was demonstrated in a recent work by Lucic et al. [21], where a large scale evaluation of some of the state-of-the-art
variants was performed. It was observed that similar results could be obtained for all the models as a result of enough random restarts and hyperparameter optimization.

In parallel with the empirical comparisons of different GANs theoretical arguments regarding the objective of these models are taking place. Arjovsky and Bottou [1] introduced the divergence that the NS-GAN minimizes, and argues why this causes mode collapse and convergence issues. The view on the training of GANs as minimization of a divergence have subsequently been criticized as overly restrictive, and empirical counterexamples have been presented to challenge this view [7].
Chapter 3

Methods

In this chapter, the methods employed to answer the proposed question are presented. This chapter begins by establishing the task as well as the datasets and data processing utilized in this project. Thereafter, the experimental setup is described and relevant neural network architectures are outlined. Finally, the approaches for training the generative models are presented. Three approaches of this kind have been employed in this project: standard Variational Autoencoders (VAEs), Progressive Generative Adversarial Networks (GANs) and Autoencoding Generative Adversarial Networks (GANs). They are presented in their respective section.

The level of detail presented here should suffice for re-implementing the algorithms and repeat the experiments of this project. There certainly exist questions which need more details to answer than what can possibly be captured in a report like this. For further details that allows replicating the experiments, the original implementation is available online.

3.1 Pupil localization

To investigate the extent to which deep generative models can act as a drop-in replacement for real datasets, a case study of pupil localization was performed. Pupil localization is an instance of the object

1 The code is available at: www.github.com/netrome/DeepGeneration
Figure 3.1: Pupil localization is the problem of finding the position and shape of the pupil in an image of an eye, such as the one to the left in this figure. This problem can be represented as a semantic segmentation problem, where the goal is to learn a mapping between the image and a segmentation map corresponding to the pupil in the image, positioned right of the image of the eye.

localization problem. The objective is to localize the pupil in an image of an eye. This problem can be formulated as a semantic segmentation problem as illustrated in figure 3.1. In normal cases, pupil localization is not achieved with deep neural networks. There already exist more computationally efficient methods for this task, such as the method proposed in [23]. On the other hand, deep neural networks for semantic segmentation are capable of predicting high quality segmentation maps on diverse complicated datasets [4] and should therefore perform extraordinary well on the relatively simple task of pupil localization.

By comparing pupil localization performance when the localizer is trained on either data from a deep generative model or data from the original dataset and tested on data from either of the domains, the usability of the generated data for model training or evaluation can be assessed. This becomes especially interesting in the case of training on generated data and testing on the original data. Assuming that the localizer learns the optimal mapping for the data domain it is trained on, the performance of this localizer can then be used as a measure of the quality and usability of the generated data.
Figure 3.2: Example of how a $256 \times 256$ image can be concatenated together with a segmentation map to form a single 2-channel image. The segmentation map in this case corresponds to the pupil in the image.

The benefit of this approach is that the relative magnitudes of the errors obtained when training on one dataset and testing on another indicate the difficulties of adapting models between these domains. This does not only give a binary answer to the original question of this project but also shows how close the tested methods are to learning the data distribution.

### 3.2 Synthetic Data

The initial experiments in this project are performed on synthetic data obtained through the rendering of a 3D head model in a data generation framework. This framework is based on the work of Świrski and Dodgson [37] and utilizes the same head rig. Two datasets with resolution $256 \times 256$ were generated, a training set consisting of 3000 images and a test set of 300 images. For reproducibility, these datasets are publicly available\(^2\).

The synthetic datasets are fully annotated with automatically generated ground truth labels. The relevant labels for the experiments are converted to heatmaps when loaded for training and evaluation as in most cases of semantic segmentation [11]. These heatmaps are there-
Figure 3.3: Example of a GAN where the generator generates a 2-channel image and the discriminator distinguishes fake 2-channel images from real ones. The training procedure is the same as that of a normal GAN.

Figure 3.4: Example of a VAE where the encoder generates a probability distribution in the latent space conditioned on a 2-channel image and the decoder maps a latent vector to a conditional probability distribution over 2-channel images.

after concatenated with the real images, forming feature maps with the dimensions $2 \times 256 \times 256$. An example of this procedure is illustrated in figure 3.2. By viewing the annotations as a part of the data, unsupervised models that learn the data distribution implicitly learn the relationships between the annotations and the images. Figure 3.3 and 3.4 show how this type of data is processed in a VAE and a GAN.

### 3.3 Real World Data

A proprietary dataset was used to further evaluate the methods. The dataset consists of 9605 different manually annotated recordings of human subjects looking at a screen, split into a training set of 8605 recordings and a test set of 1000 recordings. Each recording consists of a set of Region Of Interest (ROI) images of the eye region of the subjects.
To adapt this data to the Pupil Regression framework, the images were cropped to $256 \times 256$ patches around the eyes. An example crop together with its corresponding ROI image is shown in figure 3.5. The cropping was random and ensured that the images always contained a pupil. In practical cases it would be sensible to center the pupil in all images. For this purpose however, such a cropping would cause the networks to learn undesired patterns such as always producing trivial predictions in the center of the image. Therefore the center of the pupil was positioned randomly in the image during the cropping.

During data sampling, the probabilities of the individual images were weighted in such a way that the separate recordings were equally probable to appear. This prevents recordings with more images to be over-represented during training, which otherwise could cause the models to become skewed to the conditions of these recordings.

### 3.4 Network architectures

The deep generative models as well as the Pupil localizer are based on deep convolutional neural networks. There are four different types of networks that are utilized in the different models in this work: generators, discriminators, encoders and transformers. These types describe the transformations the networks represent.

The default generator architecture is outlined in table 3.1. The default discriminator architecture is outlined in table 3.2. The main structure of these networks follows the architectures used by Karras et al. [15], but the feature maps are smaller in this project and instead of using non-parametric upsampling followed by an extra convolution, the up-
Table 3.1: Generator architecture used in the experiments. Feature normalization and leaky ReLUs were applied after each convolution except the last one. Note that input is not technically an operation but it can be viewed as an identity operation and is included here to illustrate the input shape in a concise manner.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Output shape</th>
<th>Stride</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>128 × 1 × 1</td>
<td>1</td>
</tr>
<tr>
<td>Conv 4x4</td>
<td>128 × 4 × 4</td>
<td>1</td>
</tr>
<tr>
<td>Conv 3x3</td>
<td>128 × 4 × 4</td>
<td>1</td>
</tr>
<tr>
<td>Transpose Conv 2x2</td>
<td>128 × 8 × 8</td>
<td>2</td>
</tr>
<tr>
<td>Conv 3x3</td>
<td>112 × 8 × 8</td>
<td>1</td>
</tr>
<tr>
<td>Transpose Conv 2x2</td>
<td>112 × 16 × 16</td>
<td>2</td>
</tr>
<tr>
<td>Conv 3x3</td>
<td>96 × 16 × 16</td>
<td>1</td>
</tr>
<tr>
<td>Transpose Conv 2x2</td>
<td>96 × 32 × 32</td>
<td>2</td>
</tr>
<tr>
<td>Conv 3x3</td>
<td>80 × 32 × 32</td>
<td>1</td>
</tr>
<tr>
<td>Transpose Conv 2x2</td>
<td>80 × 64 × 64</td>
<td>2</td>
</tr>
<tr>
<td>Conv 3x3</td>
<td>64 × 64 × 64</td>
<td>1</td>
</tr>
<tr>
<td>Transpose Conv 2x2</td>
<td>64 × 128 × 128</td>
<td>2</td>
</tr>
<tr>
<td>Conv 3x3</td>
<td>32 × 128 × 128</td>
<td>1</td>
</tr>
<tr>
<td>Transpose Conv 2x2</td>
<td>32 × 256 × 256</td>
<td>2</td>
</tr>
<tr>
<td>Conv 3x3</td>
<td>16 × 256 × 256</td>
<td>1</td>
</tr>
<tr>
<td>Conv 1x1</td>
<td>2 × 256 × 256</td>
<td>1</td>
</tr>
</tbody>
</table>

Sampling is performed with transposed convolutions. These changes are motivated by the limited time-frame of this project as they enable faster training of the networks. Furthermore, the datasets of this project are believed to be less complex than those of Karras et al. [15], whereby less parameters should be necessary for the networks to capture the essence of the datasets.

The encoder follows the structure of the discriminator but does not use minibatch discrimination, and the output shape is adjusted to the size of the latent space instead of producing a scalar output.

The transformer is only used for pupil localization and is an image-to-image network. The structure of the transformer is shown in Table 3.3. The main structure resembles that of Ronneberger, Fischer, and Brox [31], but many details are different.
Table 3.2: Discriminator architecture, Leaky ReLUs were applied after each convolution except the last one.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Output shape</th>
<th>Stride</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>$2 \times 256 \times 256$</td>
<td>1</td>
</tr>
<tr>
<td>Conv 1x1</td>
<td>$16 \times 256 \times 256$</td>
<td>1</td>
</tr>
<tr>
<td>Conv 3x3</td>
<td>$32 \times 256 \times 256$</td>
<td>1</td>
</tr>
<tr>
<td>Conv 2x2</td>
<td>$32 \times 128 \times 128$</td>
<td>2</td>
</tr>
<tr>
<td>Conv 3x3</td>
<td>$64 \times 128 \times 128$</td>
<td>1</td>
</tr>
<tr>
<td>Conv 2x2</td>
<td>$64 \times 64 \times 64$</td>
<td>2</td>
</tr>
<tr>
<td>Conv 3x3</td>
<td>$80 \times 64 \times 64$</td>
<td>1</td>
</tr>
<tr>
<td>Conv 2x2</td>
<td>$80 \times 32 \times 32$</td>
<td>2</td>
</tr>
<tr>
<td>Conv 3x3</td>
<td>$96 \times 32 \times 32$</td>
<td>1</td>
</tr>
<tr>
<td>Conv 2x2</td>
<td>$96 \times 16 \times 16$</td>
<td>2</td>
</tr>
<tr>
<td>Conv 3x3</td>
<td>$112 \times 16 \times 16$</td>
<td>1</td>
</tr>
<tr>
<td>Conv 2x2</td>
<td>$112 \times 8 \times 8$</td>
<td>2</td>
</tr>
<tr>
<td>Conv 3x3</td>
<td>$128 \times 8 \times 8$</td>
<td>1</td>
</tr>
<tr>
<td>Conv 2x2</td>
<td>$128 \times 4 \times 4$</td>
<td>2</td>
</tr>
<tr>
<td>Minibatch stddev Conv 3x3</td>
<td>$128 \times 4 \times 4$</td>
<td>1</td>
</tr>
<tr>
<td>Conv 4x4</td>
<td>$128 \times 1 \times 1$</td>
<td>1</td>
</tr>
<tr>
<td>Fully connected</td>
<td>1x1</td>
<td></td>
</tr>
</tbody>
</table>

### 3.5 Variational Autoencoders

A Variational Autoencoder was constructed and trained on the synthetic data as a baseline for further experiments. The advantages of using VAEs for data generation in contrast to GANs are that they are easier to train and do not suffer from vanishing modes of data.

The Variational Autoencoder was designed using the default generator (the one from Table 3.1) network as a decoder and the default encoder network with an output shape of $128 \times 2$. In this framework, the encoder output is interpreted as the mean vector and the diagonal covariance matrix of a 128-dimensional Gaussian distribution. Using $Enc$ to denote the encoder function, this can be formulated as $P(Z|X = x) \sim \mathcal{N}(\mu, \sigma)$, where $(\mu, \sigma) = Enc(x)$. Furthermore, the posterior $P(X|Z = z)$ is modeled as a Laplace distribution where the mean value is the generator output as $P(X|Z = z) \sim \text{Laplace}(G(z), b)$, where $b$ is taken to be a hyperparameter. From a deep learning per-
Table 3.3: Transformer architecture used in the experiments. Skip layers were introduced as indicated in the table. Leaky ReLUs were applied after each convolution except the last one. In the presence of skip connections the addition of the feature maps was performed before the Leaky ReLUs were applied.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Output shape</th>
<th>Stride</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>$1 \times 256 \times 256$</td>
<td>1</td>
</tr>
<tr>
<td>Conv 1x1 (1)</td>
<td>$16 \times 256 \times 256$</td>
<td>1</td>
</tr>
<tr>
<td>Conv 3x3 (2)</td>
<td>$32 \times 128 \times 128$</td>
<td>2</td>
</tr>
<tr>
<td>Conv 3x3 (3)</td>
<td>$64 \times 64 \times 64$</td>
<td>2</td>
</tr>
<tr>
<td>Conv 3x3 (4)</td>
<td>$128 \times 32 \times 32$</td>
<td>2</td>
</tr>
<tr>
<td>Conv 3x3 (5)</td>
<td>$256 \times 16 \times 16$</td>
<td>2</td>
</tr>
<tr>
<td>Conv 3x3 (6)</td>
<td>$256 \times 8 \times 8$</td>
<td>2</td>
</tr>
<tr>
<td>Conv 3x3 (6)</td>
<td>$256 \times 4 \times 4$</td>
<td>2</td>
</tr>
<tr>
<td>Transpose Conv 4x4 + (6)</td>
<td>$256 \times 8 \times 8$</td>
<td>2</td>
</tr>
<tr>
<td>Transpose Conv 4x4 + (5)</td>
<td>$256 \times 16 \times 16$</td>
<td>2</td>
</tr>
<tr>
<td>Transpose Conv 4x4 + (4)</td>
<td>$128 \times 32 \times 32$</td>
<td>2</td>
</tr>
<tr>
<td>Transpose Conv 4x4 + (3)</td>
<td>$64 \times 64 \times 64$</td>
<td>2</td>
</tr>
<tr>
<td>Transpose Conv 4x4 + (2)</td>
<td>$32 \times 128 \times 128$</td>
<td>2</td>
</tr>
<tr>
<td>Transpose Conv 4x4 + (1)</td>
<td>$16 \times 256 \times 256$</td>
<td>2</td>
</tr>
<tr>
<td>Conv 1x1</td>
<td>$1 \times 256 \times 256$</td>
<td>1</td>
</tr>
</tbody>
</table>

The training objective can then be formulated as

$$L_{VAE}(\theta) = \lambda D_{KL}(P(Z|X=x)||\mathcal{N}(0,1)) + |\hat{X} - X|_1, \quad (3.1)$$

where $\hat{X}$ is the reconstructed image, $D_{KL}$ is the Kullback-Leibler divergence [18], $\theta$ is the parameters used for modelling $P(Z|X)$ and $P(X|Z)$ and $\lambda$ is a scale parameter that controls the trade-off between the KL term and the reconstruction loss. Scaling the KL loss has the same relative effect between the two loss terms as adjusting the $b$ parameter in the Laplacian distribution, and fixing the weight of the $L_1$ loss instead of the KL-loss means that a sensible learning rate in this setting should be similar to that of a normal autoencoder.
Figure 3.6: Structure of the convolutions in the generator network for different stages of training in the progressive GAN.

### 3.6 Wasserstein GAN

The Wasserstein GAN (WGAN) was adopted as a second baseline to compare against more complex methods. It was chosen because it currently is one of the most prominent GAN variations. Furthermore the progressive GAN is based on the WGAN causing the WGAN to be both a natural baseline for the evaluation and a special case of the progressive GAN.

The WGAN was constructed using the default generator and discriminator. To enforce the discriminator to be 1-Lipschitz \(^3\), a gradient penalty was used as described in [10].

\(^3\)The relevant theory of Lipschitz analysis is presented in [12], including the constraint on a function to be 1-Lipschitz. In short this constraint requires the function to have a bounded gradient.
CHAPTER 3. METHODS

3.7 Progressive GAN

The most advanced method tested in this project was the progressive GAN, proposed by Karras et al. [15]. The proposed advantages of this approach over other existing GAN variations is two-fold. Firstly, there is reason to believe that the quality and diversity of the generated samples are improved by the progressive training. Secondly, the training stability is believed to increase. However due to the novelty of the approach and the lack of extensive evaluations of different GANs, it is difficult to know for sure if this is the case.

The progressive GAN was adopted because of the proposed training stability that arises from progressively increasing the complexity of the learned task. It was implemented using the default generator and discriminator. The training was divided in six stages, each stage producing images with four times the resolution of the previous stage. As in the original article, each stage contained an extra 1x1 convolution to produce 2-channel images. The illustrated networks are therefore examples of stage-6 networks. To obtain a stage-5 network one should simply reshape the 1x1 convolutions and remove the two last (non 1x1) convolutions of the generator and the two first (non 1x1) convolutions of the generator. Figure 3.6 shows the structure of the generator network for different stages.

As in the original article, feature normalization, minibatch discrimination and smooth fade-in of new layers was implemented as closely as possible to the original formulation.

3.8 Autoencoding GAN

The final method tested in this project is based on a novel combination of autoencoders and GANs, named the Autoencoding GAN (AE-GAN). The proposed method consists of simultaneously training a denoising autoencoder and a GAN, using the same network as generator and decoder. This is illustrated in figure 3.7. The idea is conceptually similar to the VAEGAN framework [19]. There are two major differences between this method and the VAEGAN framework. First and foremost the autoencoder trained in this method is deterministic
and trained with the mean squared error loss function. Secondly, the adversarial loss is not used directly to train the autoencoder, but instead a normal GAN is trained separately from the autoencoder but with the decoder network as the generator.

Given enough capacity in the generator and encoder, the global minimum for the autoencoder training in this setting corresponds to the autoencoder learning the identity mapping. This causes the generator to generate realistic images on the points in the latent space the autoencoder maps the original data onto. The optimal generator in the GAN framework maps the entire latent space to realistic images. From this point of view the two objectives should lead to the same solution, however GANs tend to suffer from mode collapse and autoencoders often produce poor samples in terms of visual quality. Therefore if this method converges, it should force the decoded images to be sharp and the generator to capture the entire training data distribution.

\[ x \sim p_X \]

\[ z \sim N(0, I) \]

\[ \text{Encoder} \]

\[ \text{Decoder} \]

\[ \text{Reconstruction Loss} \]

\[ \text{Generator} \]

\[ \text{Discriminator} \]

\[ \text{Adversarial Loss} \]
3.9 Training setup

For the experiments, all models were trained using the Adamax optimizer [17] with 8 samples per batch, a learning rate of 0.0001, $\beta_1 = 0.5$ and $\beta_2 = 0.99$. The non-progressive models were trained for around 440k iterations, however convergence of the methods might be either much faster or much slower than this depending on the model. The progressive training consisted of 100k iterations per stage, with 10k iterations of fade-in between the stages.
Chapter 4

Results

A set of exploratory experiments were conducted for the different proposed methods to find settings where the methods succeed at producing visually pleasing data. Due to the unstable nature of GANs and the limited available resources in this project, most of the experiments based on GANs failed to do this. The few successful models also turned out to be highly sensitive to changes in network architecture, hyperparameters or training setup leaving a lot of room for improvement and further investigations.

Of the tested methods, only the VAE and AEGAN were capable of producing data with some visual resemblance to the original data. The Wasserstein GAN suffered from severe mode collapse and training instability thereby resulting in only failures.

The progressive GAN produced promising results on the initial stages with highly downsampled data. However the proposed fade-in procedure failed, forcing the model to re-learn everything from scratch at each stage of training. At the earlier stages the model was capable of recovering after the fade-in but not at the full resolution stage, resulting in behaviour similar to the normal WGAN.

To circumvent the fade-in issues of the progressive GAN an alternative freeze-in method of introducing the layers was tested. This approach views the introduction of the new layers as a transfer learning problem and freezes all but the newly introduced weights for a number of iterations. The method resembles gradual tuning for transfer learning
Figure 4.1: Discriminator output on real and fake images during transition between $16 \times 16$ and $32 \times 32$ resolution images for the different transition strategies on the synthetic data set when training the progressive GAN. It can be seen that the difference between the predictions increase rapidly during the end of the fade-in period whereas they remain much more stable during the weight freezing.

The discriminator output of the progressive GAN during fade-in and freeze-in is illustrated in figure 4.1. In this figure one can see that the freeze-in strategy results in a more stable transition between the stages. However this strategy requires the adversarial game between the last layers of the generator and the first layers of the discriminator to be balanced for all transitions which was observed not to be the case, which is why the progressive GAN is not featured in any further evaluation.

4.1 Qualitative Results

Figure 4.2 shows a batch of images, together with the same batch after being encoded and decoded by both a VAE and an AEGAN. From this figure it is clear that both of these methods have managed to learn to capture the main aspects of the full data distribution. Both methods display varying illuminance levels, pupil dilations, different reflections and eyelid openness. However it is common for both methods to fail to capture the original information in some of these such as pupil dilation. Even though the methods display a clear variance in pupil dilation, the reconstructed dilation does not always correspond
to the original pupil size.

The single most striking observation to emerge from comparing these images is the level of details captured by the AEGAN. Unlike the VAE, the AEGAN learned to produce glints and sharp eyelashes. It is also interesting to note that the glints are sometimes correctly positioned which due to the structure of deep convolutional neural networks is not easily learned. However in many cases it can be seen that the glints are poorly placed.

Decoding images from a learned posterior is not the same as generating new images from the prior distribution. Examples of generated images from a $\mathcal{N}(0,1)$ distribution are shown in figure 4.3. In this case the WGAN is also featured as it allows sampling new data.

What stands out in this figure after observing the decoded images is the VAE images, which sometimes look decent as before but in some cases look more artistic than realistic. These images lack many high-frequency details as previously, but in some cases even the low-frequency structure of the image is unrealistic. Except for this observation, it is also clear from the figure that the WGAN suffered from severe mode collapse whereas the AEGAN managed to capture both a good variance and visually pleasing quality of some of the samples even though some artefacts easily can be spotted in most images. Some examples
CHAPTER 4. RESULTS

(a) Generated with VAE.

(b) Generated with WGAN

(c) Generated with AEGAN

Figure 4.3: Examples of a batch of generated images by the different models.

of failure modes for the VAE and the AEGAN can be seen in figure 4.4.

4.2 Quantitative Results

For the quantitative evaluation the default transformer was trained to generate segmentation maps from images of eyes. This transformer will henceforth be referred to as the pupil regressor. The images in this context can be sampled directly from one of the data sets as well as from a generative model trained on one of these data sets. In the first case the images are denoted as "real" images, and in the second case they are denoted as "fake". A visualization of predicted segmentation maps when the pupil regressor was trained on fake data is shown in figure 4.5.

The difference between the predicted segmentation map and the original was defined as the Jaccard Distance, defined as

\[ D_{\text{Jaccard}}(X, Y) = 1 - \frac{|X \cap Y|}{|X \cup Y|}, \]  

(4.1)

where the sets \(X\) and \(Y\) are the sets of activated pixels in the respective segmentation map, which correspond to the set of 1s for the real
Figure 4.4: Some examples of different types of failure for the AEGAN (first row) and VAE (second row) models. The leftmost column shows the problems of generating realistic pupil occlusions. The middle column shows examples of multiple pupils in a single image. The rightmost column illustrates plain failures where it is difficult to see any similarities with an eye.

Figure 4.5: Segmentation map predictions for the pupil regressor when trained on data generated by the generative models. Left to right: Input image, WGAN, VAE, AEGAN.

annotations. Any segmentation map generated by a neural network may include any floating point value for each pixel. These values were therefore rounded to produce a binary segmentation map. In this evaluation there exist images without pupils in some images, this results in computing the Jaccard distance between the ground truth segmentation map and the empty set. To solve this all sets implicitly contain an extra element common to all sets. This can easily be computed as it only requires adding 1 to the denominator and numerator in the fraction in (4.1).
Table 4.1: Jaccard distance between pupil regressor output and annotations for different sources of training data using synthesized data as the original data source. The leftmost columns indicate which data set the transformer was trained on. The test data used in 4.1a is the original test set, whereas in 4.1b designated test sets were generated by the respective generative models.

<table>
<thead>
<tr>
<th>Training data</th>
<th>WGAN</th>
<th>VAE</th>
<th>AEGAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fake</td>
<td>0.813</td>
<td>0.159</td>
<td><strong>0.148</strong></td>
</tr>
<tr>
<td>Fake + real</td>
<td>0.122</td>
<td><strong>0.0944</strong></td>
<td>0.103</td>
</tr>
<tr>
<td>Real</td>
<td></td>
<td>0.0961</td>
<td></td>
</tr>
</tbody>
</table>

(a) Tested on the real test data.

(b) Test data generated by the generative models.

Table 4.2: Jaccard distance between pupil regressor output and annotations for different sources of training data using models trained on real world data as original data. The structure and notation follows table 4.1.

<table>
<thead>
<tr>
<th>Training data</th>
<th>WGAN</th>
<th>VAE</th>
<th>AEGAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fake</td>
<td><strong>4.82 \times 10^{-5}</strong></td>
<td>0.0133</td>
<td>0.0491</td>
</tr>
<tr>
<td>Real</td>
<td>0.186</td>
<td><strong>0.0264</strong></td>
<td>0.123</td>
</tr>
</tbody>
</table>

(a) Tested on the real test data.

(b) Test data generated by the generative models.

The Jaccard distances for segmentation maps produced by transformers trained and tested on different data sets are shown in table 4.1 for the models trained on synthetic data and in table 4.2 for the models...
trained on the real world data. The most important result in both of these tables is the first row which roughly corresponds to the direct usability of the generated data. In both cases, the AEGAN performed best for this measure indicating that the AEGAN produced the best data sets, as already observed in the qualitative evaluation. Another interesting finding is that training on both generated data and real world data did not manage to outperform training on real data alone on any case except the VAE on synthetic data, in which case the difference is so small that it is difficult to claim any improvement.

4.3 Augmenting data for classification

To test the usability of the deep generative models for data augmentation in a wider scope than semantic segmentation for pupil localization, two small toy experiments were performed. For the toy experiments the problem of discrete classification was considered instead of semantic segmentation. The experiments were performed on the iris and MNIST datasets.

In the case of discrete classification there is no obvious method for concatenating data with the annotations. The iris dataset is low-dimensional and each data point consists of a four-dimensional feature vector and the class label. The entire data set consists of 150 data points. To make the dataset less predictable only the first two features were used. The features were concatenated directly with the labels by adding the one-hot encoding of the label as extra dimensions of the feature vector. The train/test split was performed by splitting the dataset in half. When the discriminative model was trained with augmented data, 200 extra data points were added from the generative model.

For the MNIST dataset it is sensible to assume that a small perturbation of an image does not alter the class label. With this assumption the autoencoder-based methods can still be used for data augmentation\(^1\). Instead of sampling the latent points for the new data from the prior distribution \(P(Z)\), they are sampled from the posterior \(P(Z|X = x)\)

\(^1\)For the AEGAN, the posterior was asserted as an isotropic Gaussian with known variance, and the encoder output was interpreted as the mean. This is analogous to the VAE case.
yielding new data points. The data augmentation was performed by perturbing each minibatch with this method and training on both the perturbed and the original minibatch. The results of the MNIST and iris experiments are illustrated in Table 4.3.

Table 4.3: Toy experiments on data augmentation.

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>WGAN</th>
<th>VAE</th>
<th>AEGAN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Std</td>
<td>Mean</td>
<td>Std</td>
</tr>
<tr>
<td>Error rate (%)</td>
<td>32.0</td>
<td>3.08</td>
<td>33.3</td>
<td>3.40</td>
</tr>
</tbody>
</table>
| (a) Augmenting the iris data set. The classifier network was trained and tested five times on data augmented by each generative model. The sample mean and unbiased sample standard deviation is reported as mean and std respectively.

<table>
<thead>
<tr>
<th></th>
<th>Baseline 100</th>
<th>Baseline 200</th>
<th>Baseline 400</th>
<th>VAE</th>
<th>AEGAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error rate (%)</td>
<td>0.900</td>
<td>1.02</td>
<td>0.870</td>
<td>1.33</td>
<td><strong>0.860</strong></td>
</tr>
</tbody>
</table>
| (b) Augmenting the MNIST data set. The classifier was trained for 200 epochs when augmented with the generative models. Three different baselines, denoted "Baseline x", were trained with different number of training epochs to account for the increase in training iterations per epoch during the data augmentation. The number in the name corresponds to the number of training epochs.
This study set out with the aim of assessing the possibilities of using deep generative models for data augmentation when training deep neural networks on complex data sets. The results of the study confirm that GANs are associated with a highly unstable training procedure, whereas VAEs are more stable and capable of producing more predictable image reconstructions. The most important finding was that GANs can be improved by simultaneous training of an autoencoder. Unlike previous methods combining GANs and autoencoders [19, 27, 6, 40], no explicit coupling between the autoencoder and GAN exists. Instead sharing generator parameters and the latent space was shown to be sufficient for generating better data sets than any of the baseline methods.

One unanticipated finding in this project was that the fade in approach of Karras et al. [15] failed to preserve the learned information between the stages of the progressive GAN. A possible explanation of this is that the generator learns to mute the residual connection and compensates by scaling the original signal during the majority of the fade in period. When the fade-in almost is finished and the signal moves towards only using the new connections, the new parameters remain untrained rendering the whole process useless. The fact that this behavior is not reported in [15] suggests that it is data-dependent. Since the segmentation maps are binary, fake maps can easily be distinguished if they are scaled which indicate that this phenomenon might be triggered by this property of the data. However further investigations are
necessary before this can be stated for certain.

The problem of mode collapse is one of the main research frontiers of GANs. The extremely low fake-fake score\(^1\) for the Wasserstein GAN suggests this model suffered from severe mode collapse. By inspecting the generated images it is clear that this is the case. This outcome is contrary to the original claim by Arjovsky, Chintala, and Bottou [2] that these models should be liberated from mode collapse. However, this claim assumes the discriminator to be trained to optimality between each generator iteration which in practice is infeasible.

While the WGAN suffered from severe mode collapse, both the VAE and the AEGAN seem to have learned to capture most of the variance in the original distribution. For the VAE this result is quite expected as mode collapse generally is not a problem for these models. Furthermore, since it mostly learns low-frequency features of the data, there is less variation left to be learned making the accomplishment less impressive. The AEGAN on the other hand is partially a normal GAN and should be very susceptible to mode collapse. The fact that it manages to capture the levels of variation in pose, illumination, eyelid openness etc. suggests that the simultaneous training of an autoencoder improved the variation of the generated samples as desired.

The present study was designed to determine the effect of training neural networks on data generated by deep generative models. It is reasonable to believe that generated datasets that are qualitatively similar to the original data in terms of visual quality and variation results in better neural networks when training on the generated data. The results of this study indicate that this is the case.

It is somewhat surprising how well the numbers of the first row in the first subtable in the tables 4.1 and 4.2 correspond to the previously discussed visual quality of the data sets. The high Jaccard distance for the WGAN can be explained by the severe mode collapse from which the model suffered, making it very difficult to learn anything usable from this model for the semantic segmentation network. Moreover the loss of high-frequency details in the VAE seem to have had a similar effect, making it more difficult to generalize to the real data given only

\(^1\)The score when the discriminative model was both trained and tested on the generated data.
this data. However in this case the generated data seem to be somewhat usable, giving quite reasonable Jaccard distances. The AEGAN produced both a good variance and quite realistic images with high-frequency details, although they contained some artifacts and could without too much effort be distinguished from the real data. This resulted in the best Jaccard distances when training solely on the generated data in both cases.

Considering the data augmentation test in the first subtable of tables 4.1 and 4.2 instead of the stand alone case for the generative models, it can be observed that the best stand-alone generative model is not the same as the best generative model for data augmentation. Instead the VAE resulted in the lowest Jaccard distance for both the synthetic data and the real world data sets. It seems plausible that these results are due to the artefacts produced by the GAN training. When observing the results from training on real data and testing on synthetic one can see that the only case where the Jaccard distance is lower than the baseline is when tested on the VAE images. An interpretation of this is that the absence of high-frequency details causes this data to be easier than the original data in terms of predicting the pupil positions. This means that introducing these images during the training process of a normal network gives relatively small gradients and does not introduce false patterns that the learning process may adjust to. In other words, the VAE does not cause as much harm to the training as the other models may do.

On the other hand, using the same argument it is reasonable to suspect that the VAE is unlikely to provide the type of data that a network could benefit from during training in settings where the need for data augmentation is higher than in the pupil regression experiments, where the existing data volume is more than sufficient for the purpose. This suspicion is further confirmed by the toy experiments presented in table 4.3.

In the iris experiment there is no clear improvement using the VAE over no augmentation at all, even though it is sensible to assume that some level of improvement is obtained by reducing over-fitting in the model. However in the MNIST example, the error rate is clearly much higher for the VAE than all the baselines. It is important to keep in mind that the MNIST error rates are all very low, so the difference is with regard to some small set of difficult examples for which it would
be very surprising if the VAE augmentation would help. It is also reasonable that augmenting with the VAE data dilutes the corner cases of the training data causing the model to be more confident in the easy cases but having more problems with the edge cases, explaining why this augmentation results in worse performance.

These results are not very encouraging, although quite expected. However both of these toy experiments revealed performance improvements when the data sets were augmented by generated data from the AEGAN. Although it is important to note that these improvements might as well be due to random chance, they show that the AEGAN did not inhibit the learning process of the classifier as in the pupil regression case. Also, in both cases there are reasons to believe that the AEGAN augmentation could improve the results. For the iris data set, simply extending the size of the data set by perturbing the data points slightly should decrease the risk of over-fitting and therefore improve performance of the models. This could of course easily be performed just by adding a small Gaussian noise to the data points due to the low-dimensional structure of the data. An advantage of using deep generative models on the other hand is that the approach scales to more complex data sets. Already when considering the MNIST data set, gaussian perturbations is not going to cover the difficult corner cases. Instead when perturbing the data points with the AEGAN the shapes are modified, which forces the classifier to be less sensitive to that kind of changes. This type of perturbations could very well improve the generalization of the models which would explain why the AEGAN managed to get a lower error rate than any of the baselines in this case too.
Chapter 6

Conclusion

This study was set out to investigate if deep generative models can be applied to generate synthetic data sets that can be used to boost the performance of existing discriminative models. The experiments revealed some settings where this likely is the case. It was also illustrated that with a simple task and a large data set, a discriminative model can be used to evaluate the quality of the generated data sets. Although this study focuses on data augmentation, this finding may well have a bearing on how to benchmark different GANs in general.

The scope of this study was limited in terms of time and computational resources, and therefore several questions still remain to be answered. Notwithstanding these limitations, this study strengthens the idea that synthetic data sets can be used as drop-in replacements for existing data sets and it lays the groundwork for future research into generating usable data with deep generative models.

6.1 Future work

Since this project was of an exploratory nature, more emphasis was put on testing different models on different data sets than exploring the performance of a single model under specific circumstances. More research using controlled experiments is needed to further evaluate how much it is possible to improve the performance on MNIST and other datasets using this type of data augmentation. More broadly,
research is also needed to determine which deep generative models are most suitable for data augmentation due to the fact that many models remain untested and the current experiments are not sufficient to claim that one model is better than another in general.

Further research might also explore to what extent the choice of discriminative model affects the evaluation of the generative models. This could shed some light on how to improve the evaluation in order to find the best generative models for a given task. A natural progression on this is to test whether the best generative models for one task perform the best when the task is changed, for example test if the best generated dataset for training a pupil localizer also is the best dataset when training a gaze estimator.
Bibliography


