Synthesizing Training Data for Object Detection Using Generative Adversarial Networks

Jonathan Astermark

Master's thesis
2018:E75

LUND UNIVERSITY

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

CENTRUM SCIENTIARUM MATHEMATICARUM

Lund University

Master's Thesis

Synthesizing Training Data for Object Detection Using Generative Adversarial Networks

Jonathan Astermark

2018-12-30

Supervisors:
Kalle Åström
Jiandan Chen
Martin Ljungqvist

Examiner:
Anders Heyden

Axis Communications AB

Centre of Mathematical Sciences, Faculty of Engineering, Lund University

Abstract

Object detection is an important tool in computer vision and a popular application of machine learning. One of the main challenges in object detection, and machine learning in general, is acquiring sufficient training data. Many types of data can be hard or expensive to collect and label, or be subject to privacy concerns and regulations such as the General Data Protection Regulation (GDPR). This is particularly true for many object detection tasks, such as face detection where the training data consists of images depicting faces. Using synthetic data for training has been attempted before, but no consensus exists on how to best utilize it. This work focuses on using *a priori* trained Generative Adversarial Networks (GANs) to produce synthetic images of faces, and using them to train detectors based on Haar-like features. Experiments were conducted on both replacing real images with synthetic, and introducing synthetic variance by augmenting real images using image-to-image translating GANs. It was found that GAN-generated images can indeed be useful for detector training. Although real images consistently performs better, the amount of data plays a role as well, and a priori trained GANs can easily produce a lot of synthetic data with good variation. If real data is hard to collect, synthetic data produced by a GAN could be a viable option. It was also found that image-to-image translating GANs can be useful for data augmentation, especially when real data is scarce. Future work should focus on the impact of variance and bias in the synthetic data and how it can be controlled for optimal performance.

Keywords: Generative Adversarial Networks, Object Detection, Synthetic Training Data, Data Augmentation, Deep Learning.

Acknowledgements

I would like to extend my thanks to Axis Communications AB, and in particular the team at Core Technologies Analytics, for making this thesis work possible by both providing the topic and the opportunity to do the work at their office in Lund.

I would also like to thank my thesis supervisors, Professor Kalle Åström at the Centre of Mathematical Sciences, Faculty of Engineering at Lund University, and engineers Jiandan Chen and Martin Ljungqvist at Axis Communications, for their help and guidance throughout the work on this thesis.

I would also like to thank my friend Jonas Alfredsson, for being my reliable study mate and unfailing lab partner throughout our time at LTH, and for the countless hours we spent in the Elgkalv computer lab together.

Last but not least, I would like to give a heartfelt thanks my family and friends, for their infinite support and encouragement.

Abbreviations

CelebA	Large-scale CelebFaces Attributes Dataset
cGAN	Conditional Generative Adversarial Network
CNN	Convolutional Neural Network
DAGAN	Data Augmentation Generative Adversarial Network
FDDB	Face Detection Dataset and Benchmark
GAN	Generative Adversarial Network
GDPR	General Data Protection Regulation
WGAN	Wasserstein-GAN
WGAN-GP	Wasserstein-GAN with Gradient Penalty

Contents

1 Introduction **1**
- 1.1 Background and Motivation . . . 1
- 1.2 Purpose and Delimitations . . . 3
- 1.3 Related Work . . . 4
  - 1.3.1 Generation . . . 4
  - 1.3.2 Image Transformation . . . 6
- 1.4 Ethical Considerations . . . 6
- 1.5 Outline . . . 7

2 Theory **9**
- 2.1 Artificial Neural Networks . . . 9
  - 2.1.1 The Artificial Neuron . . . 9
  - 2.1.2 Artificial Neural Networks . . . 10
  - 2.1.3 Training a Neural Network . . . 10
  - 2.1.4 Convolutional Neural Networks . . . 11
- 2.2 Generative Adversarial Networks . . . 12
  - 2.2.1 Convergence and Mode-Collapse Improvements . . . 13
  - 2.2.2 Image-to-Image Translation Using GANs . . . 14
  - 2.2.3 Multiple-Domain Image-to-Image Translation . . . 14
- 2.3 Object Detection . . . 16
  - 2.3.1 Viola-Jones . . . 17

3 Method **19**
- 3.1 Resources . . . 21
  - 3.1.1 Datasets . . . 21
  - 3.1.2 Model Implementations . . . 22
- 3.2 Evaluation Pipeline . . . 23
  - 3.2.1 Training a Viola-Jones Detector . . . 23
  - 3.2.2 Running Detection . . . 24
  - 3.2.3 Calculating a Performance Score . . . 25

4 Experiments **27**
4.1 Image Generation 27
4.2 Image-to-Image Translation 28
4.2.1 In-context Translation 33

5 Discussion **37**
5.1 Image Generation 37
5.2 Image-to-Image Translation 39
5.2.1 In-context Translation 40
5.3 Future Work 40
5.4 Conclusion 41

Chapter 1

Introduction

1.1 Background and Motivation

Object detection is often described as a fundamental problem in computer vision [5, 12, 55, 62], with applications to self-driving vehicles [53], surveillance [56], medical technology [44] and much more. Additionally, it can serve as a first step toward more advanced computer vision tasks, such as image segmentation and object tracking [62]. The development of accurate object detectors is an active area of research that has received much attention in the last couple of decades [11, 15, 54]. A massive performance increase has been seen in recent years, thanks to increasingly advanced algorithms based on machine learning – especially with the emergence of deep learning [21].

A common denominator in machine learning algorithms is the need for large sets of training data to achieve good results. This is particularly true for deep learning, where the number of trainable parameters can be in the hundreds of thousands, or even millions [52]. Usually, acquiring these datasets is not just a matter of collecting data but also producing some sort of labelling. In the context of object detection, this usually means drawing bounding boxes around objects in images and specifying a related class label. This process can be very time-consuming and expensive, especially since it is inherently difficult to automate.

Another difficulty with data collection is the fact that a lot of data may be sensitive to collect with regards to privacy concerns. This concern is especially relevant in the case of face detection, where the training data typically consists of people's faces. Due to the need for diversity in training data, it is often

necessary to collect images of many different individuals. With developments in data protection regulations, such as the recnt introduction of the EU-wide General Data Protection Regulation (GDPR) [47], the awareness and use of ethical and consensual data collection principles has likely increased. Adhering to these principles, while still ensuring sufficiently large and diverse data sets for machine learning applications, remains a challenge.

Another problem with the data collection process is that it might introduce bias if the balance of the collected data is not carefully controlled. This can in fact have severe consequences as machine learning applications end up discriminating based on learned stereotypes present in the data. An example of this was discovered by Zhao et al. [63], who studied gender bias in semantic labeling algorithms trained on a biased dataset. The training data consisted of images depicting an action and an acting agent, labeled as either "man" or "woman". In the training data, the label "woman" occurred more often than the label "man" in scenes related to cooking. What Zhao et al. found was that the trained algorithm was more likely to mislabel people cooking as women, reflecting the bias in the training data. They also found that the trained model in fact *reinforced* the bias from the training data, by mislabeling at a higher percentage than the data was biased. This highlights the importance of having a way to control bias in the training data for machine learning applications.

The need for large and varied sets of training data, not just in the development of machine learning applications and products but also in scientific research, has led to the emergence of large public datasets. For example, the largest such dataset for face detection, at the present time and to the best of our knowledge, consists of almost 400 000 faces across more than 30 000 images [60]. The existence of such datasets reduces the need for collection of new data whenever a new machine learning algorithm needs to be trained. However, due to licensing concerns, many public datasets are not available for commercial use. Additionally, like any dataset they may suffer from some sort of bias. When using such a pre-collected dataset, it is impractical – if not impossible – to identify and control any bias introduced in the data collection process.

This apparently leaves us with only one option – to manually and painstakingly collect as many images as we can, or think we need, while to the best of our ability observing potential biases and trying to minimize them. There is, however, another potential approach: producing synthetic training data using some kind of generative model. There has been much work done on using synthetic training data for machine learning with varied results. However, most work use either explicit models to produce training data or to augment

existing data, or use data driven approaches to produce additional training samples. In this work, we would like to investigate using a generative model trained *a priori* as substitute for data collection.

If we can use a generative model to produce training data, this eliminates or reduces the need for collecting data manually. Additionally, this method may not be subject to privacy concerns and regulations on data collection storage, such as GDPR, since no identity information from the training data can be extracted from the trained model. While the identities are used to train the generative model, it is impossible to extract any information about the individual training images once the model is trained and the original data discarded. This means that even if the generative model needs training data at some point, once the generator is trained we can throw away the real data and only keep the model.

Advances in deep generative models in the last few years suggest that synthetic images generated using these models can be used for training other machine learning models. Notably, Generative Adversarial Networks (GANs) have recently been used to produce high resolution images with stunning realism [25]. GANs can also be used for image-to-image translation to produce variations on input images [8].

Le et al. [31] showed that when training machine learning algorithms on synthetic data, variance in the training data is crucial since it increases tolerance between mismatch in the training data model and test data. In deep generative models such as GANs, variance in the generated data comes from gaussian sampling of latent variables.

1.2 Purpose and Delimitations

The purpose of this thesis is to investigate if GANs can be used for producing synthetic data for the purpose of training object detection algorithms, and how it compares to using real images. We will look at both replacing real data with synthetic data, and augmenting real data using generative models. We will focus on the following questions:

Question 1. Can synthetic data from *a priori* trained GANs replace real data as training data for object detection?

Question 2. Can *a priori* trained image-to-image translation GANs be used for data augmentation, to introduce novel variations in the data?

We will evaluate the synthetic data by using it to train object detectors. We will limit ourselves to one object: faces. We will use a hand-crafted feature based detector.

1.3 Related Work

There exists much work on the topic of producing synthetic training data for machine learning. However, the methods of generation and use of the synthetic data varies greatly. We distinguish between *generation* and *image transformation* processes, where image transformation generates new samples as variations of an input sample, and pure generation processes can generate new samples by sampling from noise.

We further distinguish between model-driven and data-driven processes. The model-driven processes use some prior knowledge that can be explicitly formulated to create new samples. Examples are manual drawing of new samples or explicitly formulated image transformations of the original samples. The data-driven processes learns a model through data, e.g. machine learning based generation or translation processes.

1.3.1 Generation

Yu et al. [61] used model-driven generation of training data for Haar- and HOG-feature based person detectors, by rendering 3D models of pedestrians and pasting them into real images. They concluded that synthetic images was a suitable complement to real data, but could not replace training on real data entirely.

The conclusion that synthetic images could not replace real images for training was shared by Møgelmose et al. [38] who trained a Haar-feature based detector using synthetic images of traffic signs, generated by applying random distortions to a template.

Wang et al. [57] trained a deep-learning based licence plate recognition algorithm using synthetically generated licence plate images which they produced using a computer graphics script. However, unlike previous work, they utilized an image-to-image translation GAN (im2im-GAN) to transform the synthetic images to the space of real images, resulting in more realistic-looking synthetic training samples. They concluded that their synthetic data was always useful for pre-training, especially when the real data was

sparse. They attributed this to the prior knowledge introduced by the scripted generation, and attributed the GAN with transforming this knowledge to be more similar to real data. However, they were not successful in training with just synthetic data.

A similar method was emplyoed by Wessman and Andersson [58], who produced synthetic training data for deep-learning based person re-identification tasks. They rendered 3D models of pedestrians using the computer graphics engine Blender, and refined them using an im2im-GAN.

Zheng et al. [64] used a Deep Convolutional GAN [46] to generate images of pedestrians for training person re-identification. They used this as a method of extending the original dataset, i.e. they trained the re-ID on both the generated images and the images used to train the generator, in a semi-supervised manner.

Odena [39] also used a GAN to produce additional training samples for learning classification in a semi-supervised manner, and achieved better classification results compared to just using the real data.

Dozuas and Bacao [10] trained a conditional GAN (cGAN) to oversample the minority class in an unbalanced binary classification problem. They used this data to train a myriad of different classifiers based on logistic regression, support vector machines, nearest neighbours, decision trees, and gradient boosting machine. They found that the synthetically balanced datasets trained better classifiers than using just the real data.

Ouyang et al. [42] trained a GAN to generate new samples of pedestrians in the context of a larger image. They then added these synthetic images to the training data used to train the PS-GAN, and used the extended dataset to train Faster-RCNN [48] for pedestrian detection. Their results showed that adding synthetic images increased the detection score. However, if too many synthetic samples were added, the result was detrimental. They achieved the best result when the ratio of real to fake was between 1:2 and 1:4.

To the best of our knowledge, no extensive survey has been made regarding how well an a priori trained GAN can be utilized as a data generator in a purely data-driven generation of synthetic training data for object detection. This would be useful for privacy concerns and data regulations, assuming the trained model is sufficiently anonymous.

1.3.2 Image Transformation

It is well established that an effective method to create more training samples and to reduce overfitting is *data augmentation* through label-preserving transformations [51, 59]. A common method of data augmentation is geometric transformations in pixel space, such as affine transforms [28].

Khalil et al. [26] instead increased the variance in training data for object detection by simply cropping out the object and pasting them into different contexts. The results varied, with some objects receiving better detection performance, and some worse.

Antoniou et al. [1] took a data-driven approach to data augmentation and trained what they called a Data Augmentation GAN (DAGAN), which learned to create multiple samples from a single image. In contrast to classic data augmentation, the variations introduced in the samples created by DAGAN were not simply geometric pixel transformations. Rather, the DAGAN learned to introduce variations in deep features. For example, for images of faces, DAGAN was observed to vary attributes such as facial hair, skin tone, pose, lighting conditions, and whether the person is wearing eyeglasses. The results showed that classifiers trained with this method of data augmentation performed better than those trained without.

It is noteworthy that DAGAN was trained without attribute labels in the training data. This had the obvious benefit of not requiring these labels, but also meant there was no way of controlling which attributes the DAGAN learned to vary, or that these attributes made sense.

1.4 Ethical Considerations

Using synthetic data to train machine learning algorithms could be a way to reduce the need for personal data in technological developments. In a time where personal data has been coined "the oil of the internet and the new currency of the digital world" [29], synthetic data could be likened with a "green" alternative which carries less ramifications for data ownership and personal integrity. Such an alternative could stimulate a more ethical data economy and combat the lack of transparency present in much of data collection today [30].

Synthetic training data could also potentially help alleviate problems with bias in training and test data, since data that is hard to collect could potentially

be replaced with synthetic data. Reducing such bias is important for equality in algorithmic, data-driven decision making as demonstrated in [9, 63].

In this work, the purpose of the synthetic data is to train object detection algorithms, specifically for face detection. Highly accurate object and face detectors are of great importance to the development of e.g. surveillance systems, autonomous vehicles, and medical imaging. These technologies have the potential to help ensure a safer, better world for everyone. However, like with any technology, it is easily imagined that object and face detection can be used for unethical purposes as well. For example, they could be used as a tool in unethical, integrity violating forms of personal data collection.

In [13] it is argued that engineers are responsible for the primary purpose of the product of their work. If we accept this thought, we are through this work also morally responsible for the development of object and face detectors. It seems like the potential for good justifies both the development of this technology, and the existence of the present work; however, care should always be taken when using this technology to assure it is for a justified end.

1.5 Outline

The next chapter will describe some of the underlying theory which is the basis of this thesis. Chapter 3 will describe the method used, and Chapter 4 the specific experiments and their results. In Chapter 5 the results will be analyzed and a conclusion drawn.

Chapter 2

Theory

This chapter describes the machine learning theory which underlies the thesis. Its focus is mainly on deep learning as that is the basis of Generative Adversarial Networks, which is the main focus of this work. For a more detailed overview of deep learning and Generative Adversarial Networks, please refer to [17].

2.1 Artificial Neural Networks

Artificial neural networks is a mathematical model for information processing used in machine learning, which was originally inspired by the natural process of how biological neurons process information in the brain [36].

2.1.1 The Artificial Neuron

An artificial neuron can be described as in Figure 2.1. In its simplest terms, the neuron takes multiple input values x_k and uses them to produce an output value y. This is done by calculating a weighted sum of the inputs, and then passing this value through an *activation function* φ. More formally, the output takes the form

$$y = \varphi\left(\sum_{k=1}^{m} \omega_k x_k + b\right),$$

where b is a bias term. This is usually replaced with a unit input x_0 with associated weight ω_0, which allows us to write more compactly using vector

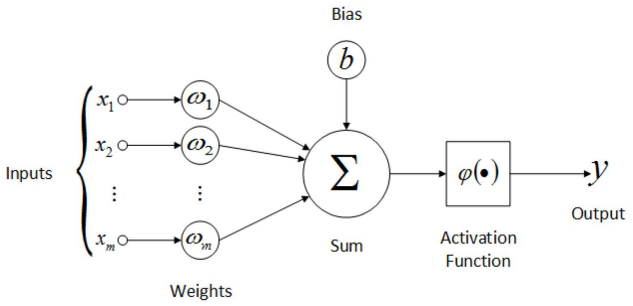


Figure 2.1: Mathematical model of an artificial neuron. Figure from [41].

notation

$$y = \varphi(\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{x}). \tag{2.1}$$

For example, if the activation function is the sign function, (2.1) corresponds to a decision plane in input space [40].

2.1.2 Artificial Neural Networks

By combining multiple artificial neurons, we can form artificial neural networks. The most straightforward is the simple perceptron, in which we have many neurons which take the same input, which leads to multiple outputs

$$y_i = \varphi(\boldsymbol{\omega}_i^{\mathrm{T}}\boldsymbol{x}).$$

Multiple neurons sharing the same inputs are said to form a *layer*. It is also possible to stack layers, so that the outputs of one layer becomes inputs to the next layer. The output of the second layer then takes the form

$$y_j = \varphi_h(\tilde{\boldsymbol{\omega}}_j^{\mathrm{T}}\boldsymbol{h}),$$
$$h_i = \varphi_0(\boldsymbol{\omega}_i^{\mathrm{T}}\boldsymbol{x}),$$

where index and tilde notation is used to indicate that weights and activation functions can differ between layers. The layer which outputs $\boldsymbol{h}$ is called a *hidden* layer. Neural networks of this type are powerful function approximators [3].

2.1.3 Training a Neural Network

The purpose of a machine learning model, such as a neural network, is to create a mapping f from an input space X to an output space Y. For

example, the inputs can be images and the outputs locations of a desired object. The mapping f is defined by the model architecture together with *trainable parameters*. In the case of neural networks, the trainable parameters are the weights $\boldsymbol{\omega}$. Training the model corresponds to setting the parameters so that f becomes the desired mapping.

In order to find suitable parameter values, we can (for example) perform *supervised learning*. The weights $\boldsymbol{\omega}$ are then first initialized to random values. The model is then presented with pairs of points $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}) \in X \times Y$. Using some suitable metric, an error $E(f(\tilde{\boldsymbol{x}}), \tilde{\boldsymbol{y}})$ is then calculated and mathematical optimization methods such as stochastic gradient descent are employed in the space of trainable parameters to minimize E.

However, to learn the exact mapping $f\colon X \to Y$ using this method, one would need to sample every single pair $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})$ contained in $X \times Y$. This is usually impossible; for example, consider the task of training a classifier to decide whether an image depicts a cat not. In that case, X is the space of all possible images which do or do not depict a cat. Obviously, during training, the model can only be presented with a limited subset $X_{\text{Train}} \subset X$ (and its corresponding pair in Y). In order to ensure the model is not overtrained on bias in X_{Train}, a separate test set $X_{\text{Test}} \subset X$ is used to validate the model once training is finished. If samples from X_{Test} does not give rise to much larger prediction errors than samples from X_{Train}, the model is assumed to generalize well to unseen samples.

2.1.4 Convolutional Neural Networks

Artificial neural networks with many hidden layers are called *deep neural networks* and form the backbone of *deep learning* [17]. Deep learning is a very powerful sub-field of machine learning. A particularly successful category of models in deep learning is *Convolutional Neural Networks* (CNNs) [17, 33], which are especially useful when the input data consists of images.

In CNNs, each layer of neurons acts as a kernel operation [17]. This both leads to a significant amount of weight sharing within each layer, which has great impact on the computational burden, and means the model is based on traditional tools in image processing. For these reasons, CNNs have been found to be largely useful and has led the front-line for developments in deep learning [17, 32].

2.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) is a generative model introduced by Goodfellow et al. [18]. In recent years, it has been the focus of much attention [20] and has been used to produce astonishing results in synthetic image generation [25]. The model borrows concepts from game theory; the key idea is that the generator is trained alongside a discriminator, with an adversarial objective to the generator. The generator and discriminator play a two-player zero-sum game where the Nash equilibrium corresponds to the generator producing samples indistinguishable from real data [17, 18, 49].

In a typical implementation the generator and discriminator are both deep neural networks. In that case, the generator is a mapping G from a random noise vector $\boldsymbol{z} \in Z$ to a point in data space $\boldsymbol{x} \in X$. The noise vector $\boldsymbol{z}$ is sampled from a probabilistic noise distribution $p_{\boldsymbol{z}}(\boldsymbol{z})$. As a consequence, the outputs of G will correspond to samples from some other distribution $p_{\mathrm{g}}(\boldsymbol{x})$.

Meanwhile, the discriminator is a mapping $D\colon X \to [0, 1]$, where the value $D(\boldsymbol{x})$ is the probability that $\boldsymbol{x}$ was sampled from the distribution of training data, $p_{\mathrm{data}}(\boldsymbol{x})$, rather than generated by G. Conversely, $1 - D(\boldsymbol{x})$ is the probability that $\boldsymbol{x}$ was *not* sampled from p_{data} but from p_{g}, i.e. $\boldsymbol{x} = G(\boldsymbol{z})$, $\boldsymbol{z} \sim p_{\boldsymbol{z}}$.

An optimal discriminator will maximize the expected output when $\boldsymbol{x} \sim p_{\mathrm{data}}$ and minimize it when $\boldsymbol{x} \sim p_{\mathrm{g}}$. This motivates the definition of the value function [18]

$$V(G, D) = \mathbb{E}_{\boldsymbol{x} \sim p_{\mathrm{data}}} \left[\log D(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}} \left[\log(1 - D(G(\boldsymbol{z})))\right]. \tag{2.2}$$

Maximizing $V(G, D)$ with respect to D corresponds to finding an optimal discriminator. Logarithms have been applied to the probabilities in (2.2) to prevent very large gradients from appearing in the optimization, which could otherwise be caused by V varying by several orders of magnitude.

Conversely to the discriminator, the objective of the generator is to fool the discriminator as often as possible. This means that the generator should instead try to minimize the value function. This conflict defines the min-max game

$$G^* = \arg\min_G \max_D V(G, D),$$

where G^* is an optimal generator. It can be shown that the global optimum occurs when $p_{\mathrm{g}} = p_{\mathrm{data}}$ [18].

In practice, G and D are implemented using neural networks and can be written $G(\boldsymbol{z}; \boldsymbol{\omega}_G)$ and $D(\boldsymbol{x}; \boldsymbol{\omega}_D)$, respectively, where $\boldsymbol{\omega}_G$ and $\boldsymbol{\omega}_D$ are the weights of the respective networks. The distribution p_g is thus implicitly defined by the network weights $\boldsymbol{\omega}_G$. The networks are updated alternatingly to minimize or maximize $V(G, D)$, respectively, and typically an update step for G is run for every n_critic update steps for D. For more details on the training algorithm, we refer to [18].

For future reference, we re-formulate the value function (2.2) as a *loss function*

$$\mathcal{L}_\mathrm{GAN} = \mathbb{E}_{\boldsymbol{x} \sim p_\mathrm{data}} \left[\log D(\boldsymbol{x})\right] - \mathbb{E}_{\tilde{\boldsymbol{x}} \sim p_\mathrm{g}} \left[\log(D(\tilde{\boldsymbol{x}}))\right]. \tag{2.3}$$

During training, G is trained to minimize the loss $\mathcal{L}_\mathrm{GAN}$ while D is trained to maximize it. It is also possible to have different loss functions for G and D, respectively. Actually, for convergence reasons, in [18] D is trained to minimize $\log(1 - D(\tilde{\boldsymbol{x}}))$ rather than maximize $\log(D(\tilde{\boldsymbol{x}}))$.

2.2.1 Convergence and Mode-Collapse Improvements

A problem with the GAN model defined in [18] is that it is not guaranteed to converge in practice [16] – or it might converge to a single point, usually referred to as "mode collapse" [49]. These problems are addessed by Wasserstein-GAN (WGAN) [2] which minimizes the so-called Earth-mover distance between the probabilities p_data and p_g. This is done by replacing the loss function in (2.3) with

$$\mathcal{L}_\mathrm{WGAN} = \mathbb{E}_{\boldsymbol{x} \sim p_\mathrm{data}} \left[D(\boldsymbol{x})\right] - \mathbb{E}_{\tilde{\boldsymbol{x}} \sim p_\mathrm{g}} \left[D(\tilde{\boldsymbol{x}})\right]$$

and introducing the additional constraint that D is 1-Lipschitz [2, 19]. In WGAN, this constraint is enforced by weight clipping at each update step to ensure that each individual weight $\omega_i \in [-c, c] \forall i$ for some predefined $c > 0$. The Lipschitz constraint leads to better behaved gradients [2], so the logarithms from (2.2) are no longer needed.

Gulrajani et al. [19] improved the enforcement of the Lipschitz constraint in WGAN by replacing the weight clipping with a gradient penalty. They used the loss function

$$\begin{aligned} \mathcal{L}_\text{WGAN-GP} = \mathbb{E}_{\tilde{\boldsymbol{x}} \sim p_\mathrm{g}} \left[D(\tilde{\boldsymbol{x}})\right] - \mathbb{E}_{\boldsymbol{x} \sim p_\mathrm{data}} \left[D(\boldsymbol{x})\right] \\ + \lambda_\mathrm{GP} \mathbb{E}_{\hat{\boldsymbol{x}}} \left[\left(\left\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\right\|_2 - 1\right)^2\right], \end{aligned} \tag{2.4}$$

where λ_GP is a hyperparameter controlling the impact of the gradient penalty term and $\hat{\boldsymbol{x}} = t\tilde{\boldsymbol{x}}' + (1 - t)\boldsymbol{x}'$ is formed by sampling $\tilde{\boldsymbol{x}}' \sim p_\mathrm{g}$, $\boldsymbol{x}' \sim p_\mathrm{data}$ and $t \sim \mathcal{U}(0, 1)$.

Wasserstein-GAN with Gradient Penalty (WGAN-GP) has been successfully used to produce synthetic images with a high level of realism [25]. The training algorithm is described in detail in [19] and uses the Adam optimization method [27] rather than SGD.

2.2.2 Image-to-Image Translation Using GANs

Mirza and Osindero [37] modified the GAN model to create what they called a Conditional GAN (cGAN). Here, labels on the training data are utilized to control the generation by specifying a target subset of data space $X_c \subseteq X$ at generation time. This is done by conditioning both the generator and discriminator on a class label $\boldsymbol{c}$ associated with that specific subset. In practice this is done by concatenating the input to G and D with the label $\boldsymbol{c}$, so that $G = G(\boldsymbol{z}, \boldsymbol{c})$ and $D = D(\boldsymbol{x}, \boldsymbol{c})$.

Isola et al. [22] further modified the cGAN to perform image-to-image translation (im2im-GAN). They translated images from one domain, X, to another, Y, by using the image as condition in the cGAN, i.e. they used $G = G(\boldsymbol{z}, \boldsymbol{x})$ and $D = D(\boldsymbol{y}, \boldsymbol{x})$ where $\boldsymbol{x} \in X$ and $\boldsymbol{y} \in Y$. Using this model, it is possible to translate e.g. black-and-white images to color, daylight images to night, and sketches to photographs.

A major drawback of the im2im-GAN by Isola et al. is the need to perform supervised learning using paired training samples. This was addressed by Zhu et al. [65], by training a second generator $F\colon Y \to X$ alongside $G\colon X \to Y$. They then required the two generators to be *cycle consistent*, i.e. that $F \circ G$ and $G \circ F$ maps samples back onto themselves. This was enforced by minimizing the 1-norm between samples and their "cycled" image, leading to the cycle-consistency loss

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[\|F(G(\boldsymbol{x})) - \boldsymbol{x}\|_1\right] + \mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}}\left[\|G(F(\boldsymbol{y})) - \boldsymbol{y}\|_1\right]. \quad (2.5)$$

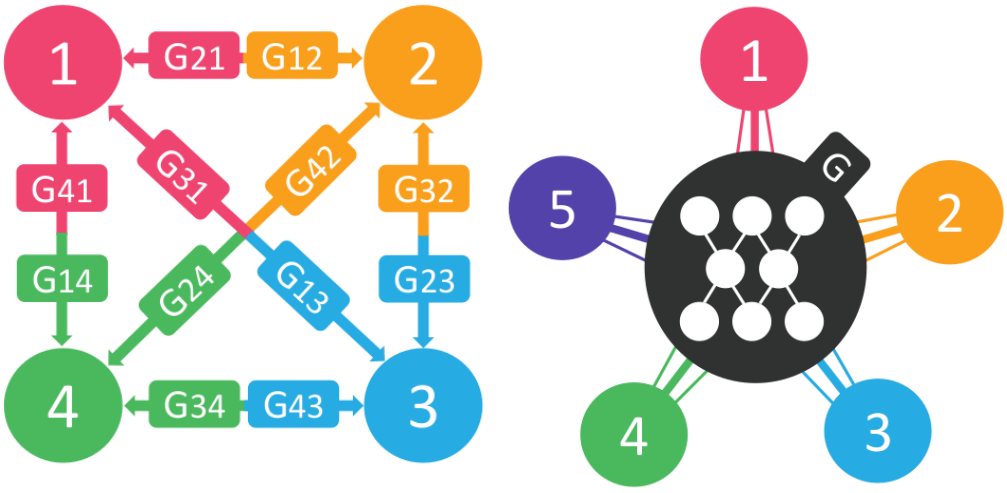
By adding this term to the loss function in (2.3), Zhu et al. were able to perform impressive image translation without any paired training samples; for example they could translate paintings by Monet to photographs, and vice versa.

2.2.3 Multiple-Domain Image-to-Image Translation

While Zhu et al. achieved impressive results on mapping from one domain to another, we are more interested in translating a single image to multiple

domains. This is possible using the method in [65], of course, by training many different generators. However, this method does not scale well with the number of domains.

This scalability problem was addressed by Choi et al. [8] who trained a single generator (and discriminator) to generate conditionally from multiple domains. The model is called Star-GAN, after the star-like topology in which the generator can take an input from any domain and produce an output in any domain, see Figure 2.2.



(a) Multiple generators *(b) Star-GAN*

Figure 2.2: Illustration of the difference in topology between (a) using many single-domain translators such as [65] and (b) Star-GAN. Image is from [8].

In order to accomplish this, StarGAN uses the trick from cGAN of conditioning the generator on a target domain label, so that $G\colon \{\boldsymbol{x}, \boldsymbol{c}\} \mapsto \boldsymbol{y}$. They also modified the discriminator to not only produce a probability distribution over the two sample sources D_{src}, but also a probability distribution over the domains $D_{\text{cls}}(\boldsymbol{x})$, so that $D\colon \boldsymbol{x} \mapsto \{D_{\text{src}}(\boldsymbol{x}), D_{\text{cls}}(\boldsymbol{x})\}$.

The domain classification loss $D_{\text{cls}}(\boldsymbol{x})$ was introduced to ensure that images generated with the condition label $\boldsymbol{c}$ will be classified as being in that domain. This is enforced by the classification loss

$$\mathcal{L}_{\text{cls}}^{\texttt{fake}} = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}, \boldsymbol{c}\in C}\left[-\log D_{\text{cls}}(\boldsymbol{c}|G(\boldsymbol{x}, \boldsymbol{c}))\right].$$

The classifier needs to learn this classification from real samples. This is

enforced by the following loss:

$$\mathcal{L}_{\text{cls}}^{\texttt{real}} = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}},\boldsymbol{c}\in C}\left[-\log D_{\text{cls}}(\boldsymbol{c}|\boldsymbol{x})\right].$$

In order to enforce preservation of the image content not related to the domain transfer, a cycle consistent reconstruction loss was used similarly to that in [65] to minimize the L_1 norm of an image passed through the generator twice (cf. (2.5))

$$\mathcal{L}_{\text{rec}} = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}},\boldsymbol{c}\in C}\left[\|\boldsymbol{x} - G(G(\boldsymbol{x},\boldsymbol{c}),\boldsymbol{c}')\|_1\right],$$

where $\boldsymbol{c}'$ is the label associated with the image's original domain.

The adversarial generator-discriminator interaction was controlled by the WGAN-GP loss (cf. (2.4)):

$$\begin{aligned}\mathcal{L}_{\text{adv}} = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[D_{\text{src}}(\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}},\boldsymbol{c}\in C}\left[D_{\text{src}}(G(\boldsymbol{x},\boldsymbol{c}))\right] \\ - \lambda_{\text{GP}}\mathbb{E}_{\hat{\boldsymbol{x}}}\left[(\|\nabla_{\hat{\boldsymbol{x}}} D_{\text{src}}(\hat{\boldsymbol{x}})\|_2 - 1)^2\right].\end{aligned}$$

The full loss functions used to update G and D are:

$$\begin{aligned}\mathcal{L}_G &= \mathcal{L}_{\text{adv}} + \lambda_{\text{cls}}\mathcal{L}_{\text{cls}}^{\text{fake}} + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}}, \\ \mathcal{L}_D &= -\mathcal{L}_{\text{adv}} + \lambda_{\text{cls}}\mathcal{L}_{\text{cls}}^{\text{real}},\end{aligned}$$

where λ_{cls} and λ_{rec} are hyperparameters controlling the relative impact of classification and reconstruction loss, respectively.

2.3 Object Detection

Object detection is the task of determining the location of a specific object in an image. In many use-cases, there can be zero, one, or multiple different instances of the object in a single image. It can also entail detection of many different kinds of objects simultaneously, and distinguishing them from each other. In that case, the detector needs to be able to distinguish between all those objects, or multiple single-object detectors needs to be run on each image.

Object detection can be compared to another common computer vision application of machine learning: classification. Since object detection is basically classification with the added step of localization, object detection can be said to be strictly harder than classification. A straightforward implementation of object detection is to simply run a classifier on every

feasible sub-region of an image, at some level of granularity. This approach is the basis for *sliding window* based detectors [48], and some of the earliest efficient object detectors were based on this method [54]. A sliding window based detector is trained like a classifier, with positive and negative examples.

More modern object detectors, typically based on deep learning, do not use a sliding window approach but instead includes a region proposal component. This means that the detector learns, along with the classification, to propose the windows on which classification should be run [48]. This type of detector requires training images where the object occurs "in-context" in the same way they will when the actual detection task is performed. This also means that the training images needs to have object locations labeled.

2.3.1 Viola-Jones

Although sliding window based detectors are basically classifiers, it does not work well to simply train any classifier and run it as a detector. The reason for this is that it needs to be run very many times for each image, typically in the millions [54], where most windows are negatives.

The first efficient object detection algorithm was developed by Viola and Jones [54] and is based on the sliding window approach. It is quick to discard negatives because it runs multiple classifiers in succession, with increasing complexity. The first layer of classifiers are simple and fast, but can still discard many negative samples at the cost of a high rate of false positive detections. Subsequent layers will thus operate on much fewer samples and can afford harder scrutinization. Ideally, each layer will eliminate some false positives from the previous layer, while keeping all of the true positive detections. This idea is illustrated in Figure 2.3.

Each classifier in the cascade work by detecting a few simple features, reminiscent of Haar basis functions [43]. Viola and Jones suggested using two- three- and four-rectangular features, illustrated in Figure 2.4. The set of all possible such features for a given image is naturally very large, and extensively evaluating them all would be prohibitively expensive. The solution to this is to learn a classification function which selects a small set of features to evaluate [54].

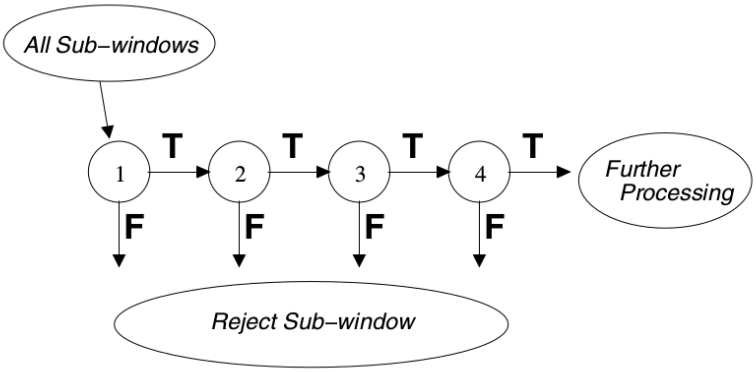


Figure 2.3: Schematic over the cascade of classifiers used in the Viola-Jones detector. Source: [54].

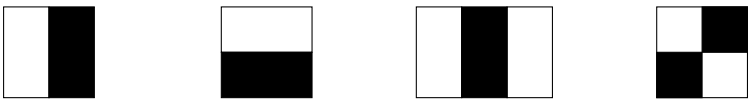


Figure 2.4: The features used in the Viola-Jones detector can be divided into three categories: edge features (a and b), line features (c), and four-rectangle features (d).

Chapter 3

Method

The goal of the present work is to investigate if synthetic data generated with GANs can be used as a substitute for real data for training object detection algorithms. In Section 1.2 we formulated two questions to which we shall direct our focus. For instructive purposes, they are repeated here:

Question 1. Can synthetic data from *a priori* trained GANs replace real data as training data for object detection?

Question 2. Can *a priori* trained image-to-image translation GANs be used for data augmentation, to introduce novel variations in the data?

To investigate these questions, we need to establish a framework for comparing different datasets' usefulness as training data. This is done by training an object detector with the different datasets and running them through a detection benchmark to establish a score for the detectors' performance. The process is illustrated in Figure 3.1 and 3.2 for the tasks related to Question 1 and 2, respectively. As seen there, the evaluation pipeline is the same for both tasks.

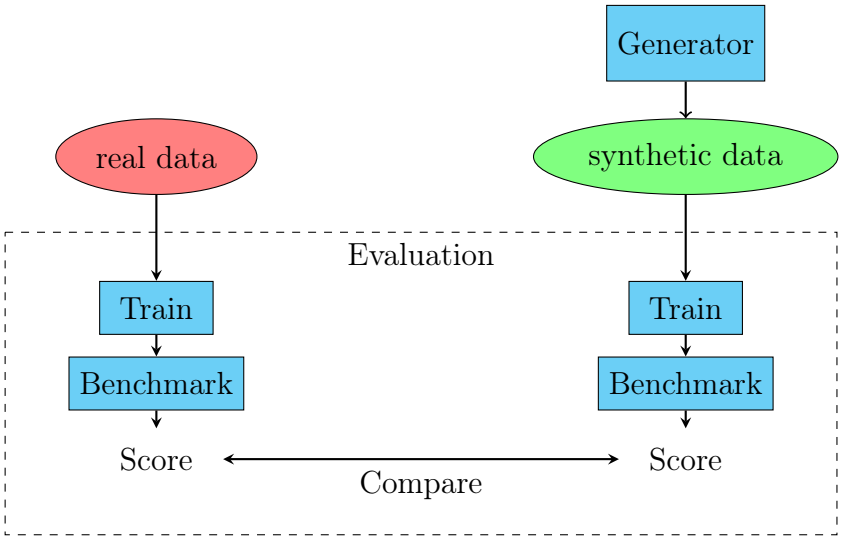


Figure 3.1: Pipeline for comparing using synthetic training data vs. using real data.

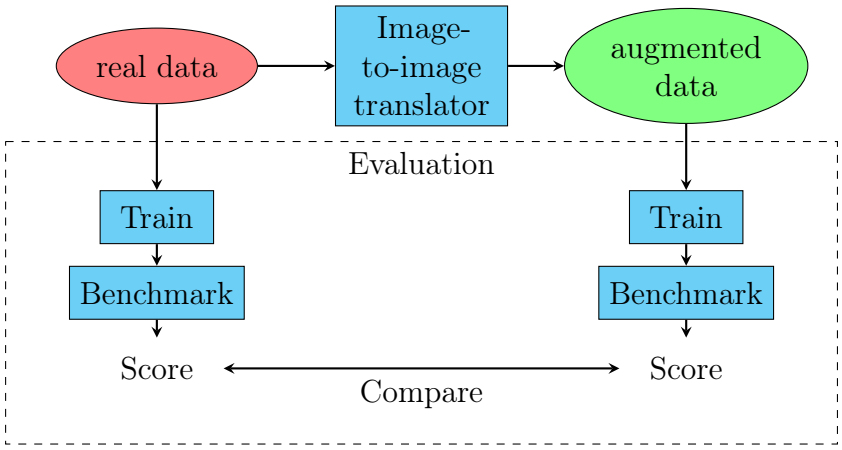


Figure 3.2: Pipeline for evaluating image-to-image translation GAN for data augmentation.

3.1 Resources

3.1.1 Datasets

Training data

For training data, we used the publicly available *Large-scale CelebFaces Attributes Dataset* (CelebA) [35], which consists of 202 599 head-shot style images of celebrities with 10 177 different identities. These images are also labelled with 40 binary attributes describing facial features, hair color and accessories, which will be utilized for training image translation.

Evaluation data

For evaluation data, we used the publicly available *Face Detection Dataset and Benchmark* (FDDB) [23], which has 5171 faces annotated over 2845 images. This dataset was specificly created to be an unconstrained benchmark for face detection, with large variations in pose and facial attributes [23].

To also get an idea of the detection performance on faces with different attributes, evaluation was also run on a few highly biased datasets. These were available internally at Axis Communications AB and had been manually collected from Flickr at an earlier time. The biased datasets are summarized in Table 3.1.

Table 3.1: Overview of biased datasets used for evaluation.

Dataset	Images	Faces	Bias
Children	260	277	Infants and small children.
Elderly	176	187	People with aged faces; wrinkles.
Glasses	178	184	People with eyeglasses and sunglasses.
Women	219	225	Images of women.

3.1.2 Model Implementations

Synthetic Image Generation

For synthetic image generation, we used a GAN based on the WGAN-GP loss implemented in Python 3 by Karras et al. [25], using TensorFlow 1.6, which was available as open source [24]. This model used several tricks on top of WGAN-GP described in [25] and had been trained to produce high-resolution images of faces and other objects. Trained models produced by Karras et al. were available for download along with the code. We used a model trained on CelebA which could produce head-shot images of imaginary celebrities at a 1024×1024 resolution.

Image-to-Image Translation

For image-to-image translation, we used a Star-GAN model implemented in Python 3 by Choi et al. [8], using PyTorch 0.4.0, available as open source [7]. We did not use any pre-trained models, but instead trained our own from scratch. Relevant training parameters used are summarized in Table 3.2. The Adam parameters α, β_1 and β_2 are described in [27], while the rest are described in Section 2.2.

Table 3.2: Training parameters used for Star-GAN.

Training parameter	Value
λ_{GP}	10
λ_{cls}	1
λ_{rec}	10
n_{critic}	5
α	0.0001
β_1	0.5
β_2	0.999

Object Detection

For object detector in the evaluation pipeline, we used the Viola-Jones detector implemented in the open source computer vision library OpenCV 2.4.9.1 [4]. Training was done using the tool `opencv_traincascade` included in OpenCV.

The training parameters passed as arguments to `opencv_traincascade` are summarized in Table 3.3. Inference using the trained model was done in Python 2 using the package `cv2`.

Table 3.3: Arguments passed to `opencv_traincasacade` when running training.

Argument	Value
`numStages`	20
`minHitRate`	0.999
`maxFalseAlarmRate`	0.5
`w`	20
`h`	20
`mode`	BASIC
`featureType`	HAAR
`acceptanceRatioBreakValue`	10^{-5}

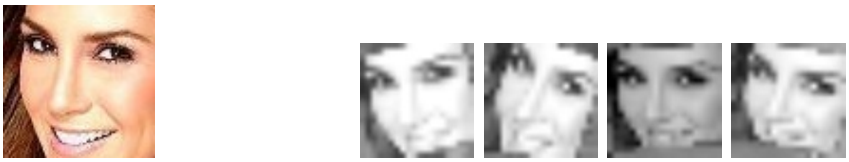
3.2 Evaluation Pipeline

The evaluation pipeline consisted of three steps: training a detector, running detection on a benchmarking dataset, and calculating a performance score.

3.2.1 Training a Viola-Jones Detector

In order to train a new detector, we need to provide sets of both positive and negative images. The positive images are examples of the object we want to detect, while the negative images are examples of anything else. For negative images, we consistently used an Axis internal dataset consisting of 3019 images depicting a wide variety of scenes and objects.

Using the OpenCV utility `opencv_createsamples`, we could apply the common data augmentation method of applying geometric transformations to create multiple positive samples from each positive image. The transformations used in this work were rotation, shearing, and variation of pixel intensity, see Figure 3.3. Each formed sample was pasted onto a random background from the negative images. The samples were also resized to a specified width and height, and converted to greyscale. This was done because the Viola-Jones detector only works with pixel intensities and not color channels.



(a) Original *(b) Samples*

Figure 3.3: Multiple samples were created from the original by applying geometric transformations and pasting it onto random negative images. The samples were also rescaled and converted to greyscale.

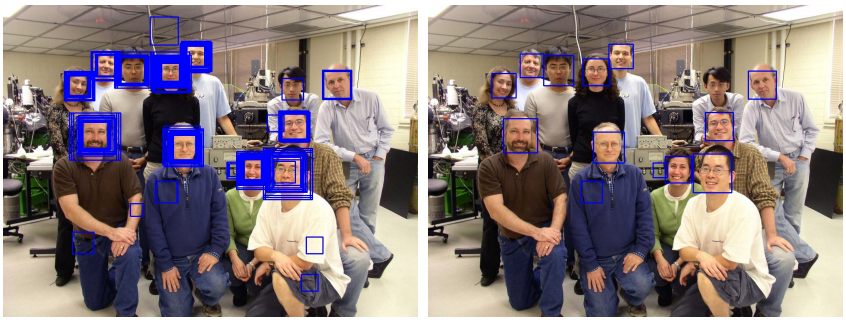
By using this method, a dataset of eg. 100 images of faces could be turned into 1000 positive samples by sampling each image 10 times. In this work, the rotational transformation was sampled uniformly from the angular interval ± 0.3 rad. The shears were actually performed as rotations around the images' x- and y-axis, followed by projection onto the view plane. These rotations were sampled uniformly from the angular intrerval ± 0.6 rad. The arguments passed to `opencv_createsamples` are summarized in Table 3.4.

Table 3.4: Arguments passed to `opencv_createsamples` when performing data augmentation.

Argument	Value
`maxxangle`	0.6
`maxyangle`	0.6
`maxzangle`	0.3
`maxidev`	40
`w`	20
`h`	20

3.2.2 Running Detection

The inference step of the Viola-Jones algorithm is done by sliding a window across the image at multiple scales. Each window is then evaluated by passing it through the cascade of classifiers. This was done using the OpenCV-function `detectMultiScale` with the `scaleFactor` parameter set to 1.3. Since variation in window position and scale is small, it is likely that each positive classification is accompanied by many other neighbouring and overlapping positive classifications. This leads to the detector often



(a) (b)

Figure 3.4: Example of the effect of the `minNeighbors` *parameter. Effect of balancing between reducing false-positives while retaining true positives. In a) minNeighbors is 0. In b)* `minNeighbors` *is set to 1, so overlapping boxes have been averaged. The number of false positives is reduced, but this happened at the cost of reducing the number of true positives as well.*

outputting multiple detection boxes at the same location, as can be seen in Figure 3.4a.

To handle this, the inference step takes an integer parameter, `minNeighbors`. When this is set to $n > 0$, overlapping boxes will be averaged into a single box. Additionally, all boxes which do not overlap with at least n other boxes will be turned into negative detections. This enables the possibility to reduce the number of false positives by removing the least certain detections. However, there is also a risk of removing true positives this way. Tuning of the parameter `minNeighbors` is thus a balancing act between reducing false positives and retaining true positives, see Figure 3.4.

3.2.3 Calculating a Performance Score

Since our evaluation datasets are annotated with ground truths, once detection has been run we can determine if each detection is a true or false positive. This is done by calculating the pixel overlap ratio called intersection over union (IOU) [34], which is the area of the intersection between a detection box and ground truth, divided by the area of the union. If this ratio is larger than a certain threshold, usually set to 0.5, the detection is determined to be a *true positive*; otherwise it is regarded as a *false positive*. A ground truth which is not detected is called a *false negative* [23]. In our experiments, the IOU threshold value was set to 0.4 to make the score less sensitive to differences in labelling convention between different datasets.

When we know the number of true positives (TP), false positives (FP) and false negatives (FN) for an entire dataset, we can calculate two metrics called *precision* and *recall*. Precision is the ratio of detections which were true, and recall is the ratio of how many of the objects were detected [11]:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

By calculating their harmonic mean, we get a measure called F1-score [6, 50]:

$$\text{F1-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

When benchmarking a detector to evaluate its training data, we do not want the method of setting `minNeighbors` to affect the result. For this reason, we run the detector for all values of the parameter between 1 and 10, and calculate the corresponding F1-score. We then report the highest of these scores.

Bounding Boxes for FDDB

Calculating IOU is complicated for the FDDB benchmark by the fact that FDDB uses elliptical ground trouth annotations, while detectors typically output rectangular detection boxes (as seen in Figure 3.4). When calculating the IOU for detections on FDDB, we thus chose to approximate the elliptical ground truths with a rectangle. We did this by horizontal projection of the major axis, and vertical projection of the minor axis, as illustrated in Figure 3.5. While this practice means that IOU was not calculated using the exact ground truth, the impact on the results are likely negligible – especially considering the lowered IOU threshold.

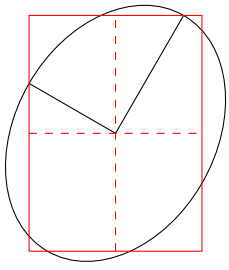


Figure 3.5: Approximation of elliptical bounding box with a rectangle, using projection of the major and minor axes.

Chapter 4

Experiments

We conducted two sets of experiments, to answer the two different questions posed under the scope of replacing data collection with a priori trained GANs. First we compared training on data generated by a GAN with training on real data. Then we tried using an image-to-image translation GAN for data augmentation.

4.1 Image Generation

To test the possibility of using synthetic images instead of real images for training, we used the pretrained GAN model by Karras et al. [25] trained on CelebA to generate 5000 synthetic "fake-CelebA" images. This was done by sampling a 512-dimensional vector $\boldsymbol{z} \sim \mathcal{N}(0, I)$ and generating $G(\boldsymbol{z})$. Samples of synthetic images can be seen in Figure 4.1 alongside real samples from CelebA. Note that only the face area is presented. We then trained Viola-Jones detectors using subsets of these, from 10 images up to all 5000. To compare with real training data, we also trained detectors using subsets from the real CelebA, from 10 images to 5000. The number of negative samples used was set constant to 3000. No data augmentation was used in this experiment.

Each trained detector was then run on the five evaluation datasets from Section 3.1.1, and the F1-score was calculated. The results can be seen in Figure 4.2. We see that using synthetic data is consistently worse than using the same amount of real data. But we also see a trend that using more data is better than using less data. This trade-off means that using a lot of

(a) *(b)*

Figure 4.1: Samples of (a) real images from CelebA and (b) synthetic images generated by a GAN trained on CelebA. Each synthetic image is the generator output $G(\boldsymbol{z})$ from a noise sample $\boldsymbol{z} \sim \mathcal{N}(0, I)$.

synthetic data can be better than using a little real data. For example, using 5000 synthetic images gives a better score than 100 real images for all tests. In Table 4.1, the scores for real images and 5000 synthetic images are also presented to highlight the cutoff where 5000 synthetic data becomes better to use than real images. We see that this cutoff happens around 600-800 real images.

In Figure 4.2, we also see that performance varies across the datasets with different bias, with the Glasses and Elderly datasets being the hardest to achieve a high score on.

In order to investigate the limits of achievable performance using the synthetic data generated here, we finally trained a detector on 5000 synthetic images while applying the data augmentation technique described in Section 3.2.1. All training parameters were retained, except the `acceptanceRatiobreakValue` parameter from Table 3.3, which was not used in this experiment. For comparison, we also trained a detector on 5000 real images using the same method. The results of evaluating on the five test datasets from Section 3.1.1 are presented in Table 4.2. We see that real data consistently gave a better performance on all test datasets, except for the Women dataset, where the detectors trained on real and synthetic data, respectively, performed equally.

4.2 Image-to-Image Translation

The second question posed concerned using image-to-image translating GANs for data augmentation. This idea is similar in spirit to the DAGAN in [1].

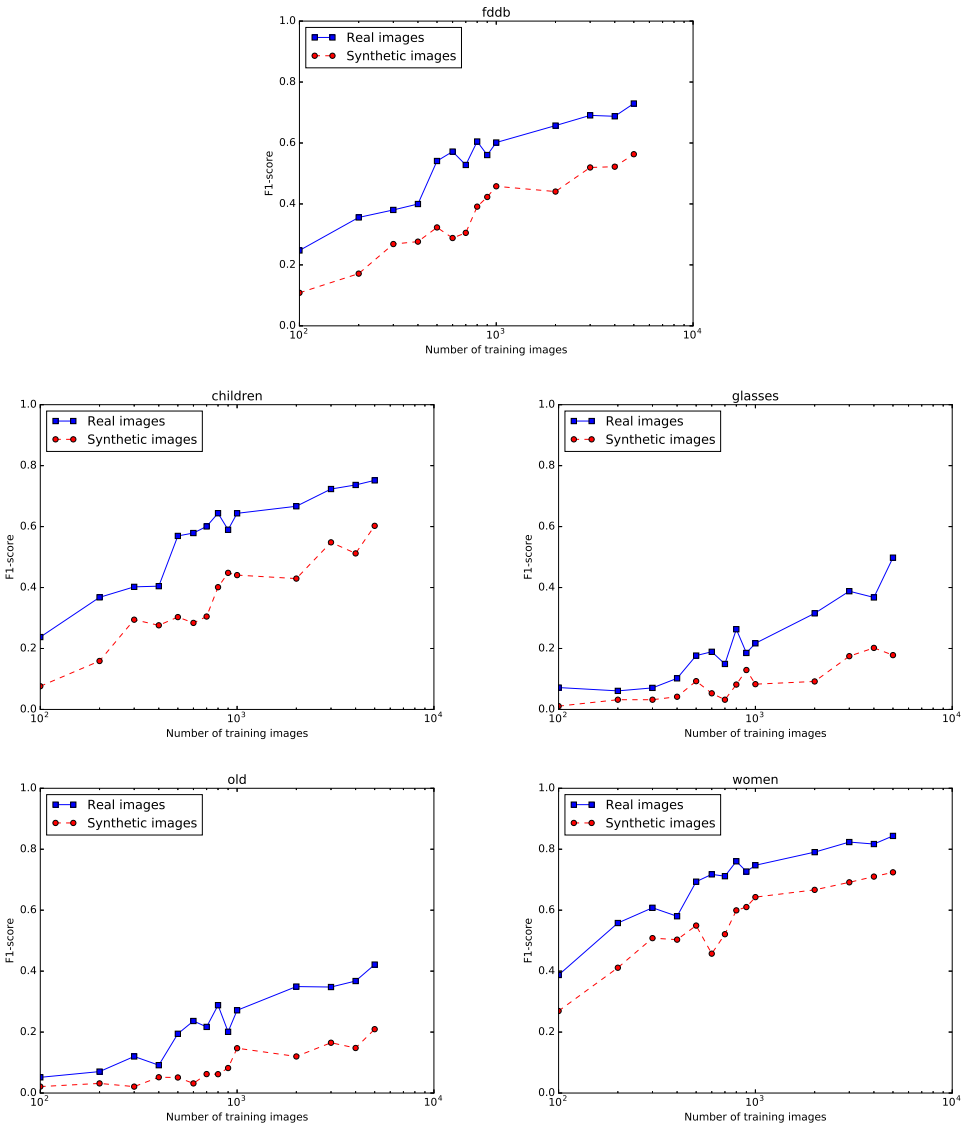


Figure 4.2: F1-score of Viola-Jones detectors trained for face detection, as a function of the number of positive training images. Training was done either on real images (blue squares) or GAN-generated, synthetic images (red circles). The detectors were evaluated on five different datasets, corresponding to the five different plots.

Table 4.1: F1-score for training on real data (blue squares in Figure 4.2). The first score to reach the same or better performance than 5000 synthetic images is marked with bold.

Training data	# images	FDDB	children	glasses	elderly	women
Real	100	0.25	0.24	0.07	0.05	0.39
Real	200	0.36	0.37	0.06	0.07	0.56
Real	300	0.38	0.40	0.07	0.12	0.61
Real	400	0.40	0.40	0.10	0.09	0.58
Real	500	0.54	0.57	0.18	0.19	0.69
Real	600	**0.57**	0.58	0.19	**0.24**	**0.72**
Real	700	0.53	**0.60**	0.15	0.22	0.71
Real	800	0.60	0.64	**0.26**	0.29	0.76
Real	900	0.56	0.59	0.19	0.20	0.73
Real	1000	0.60	0.64	0.22	0.27	0.75
Real	2000	0.66	0.67	0.32	0.35	0.79
Real	3000	0.69	0.72	0.39	0.35	0.82
Real	4000	0.69	0.74	0.37	0.37	0.82
Real	5000	0.73	0.75	0.50	0.42	0.84
Synthetic	5000	0.56	0.60	0.20	0.21	0.72

Table 4.2: F1-score of Viola-Jones detectors trained on 5000 real or synthetic positive training data, respectively. Classical data augmentation was used, as described in Section 3.2.1. The best score achieved for each dataset is marked with bold.

Evaluation Dataset	Score (Real Data)	Score (Synthetic Data)
FDDB	**0.77**	0.73
Children	**0.84**	0.78
Elderly	**0.57**	0.45
Glasses	**0.52**	0.43
Women	**0.87**	**0.87**

Their idea was that instead of performing a small number of hand-picked transformations, they trained a neural network to decide how to make the transformations instead. This led to more complex variations being introduced than using standard augmentation methods. One advantage of using image-to-image translation, however, is the possibility to control the sampling of the augmentations to attribute labels.

For this purpose, we trained a Star-GAN model, using all images in CelebA except 2000 randomly selected for testing. We trained the model using four salient labeled attributes: "wearing eyeglasses", "mouth slightly open", "mustache", and "smiling". Training was done for 200 000 iterations with a batch size of 16. At that point, no further improvement in visual quality seemed to happen with more iterations, so training was stopped. An example of images transformed with the trained model can be seen in Figure 4.3.

Figure 4.3: Example of images transformed using Star-GAN. The leftmost images are the originals from CelebA, and the following images are translations which added the attributes (from left to right): "wearing eyeglasses", "mouth slightly open", "mustache", and "smiling".

To test how this method of data augmentation impacted training, a number of real images were selected from the CelebA dataset and used to train a Viola-Jones face detector. Two experiments were made, where 50 or 250 images were hand-picked to ensure they did not contain the feature "wearing eyeglasses". In the 50-images experiment, it was also ensured they did not contain the feature "mustache", and that the images were not in profile. Training was done using the parameters from Table 3.3, except `acceptanceRatioBreakValue` which was not used. The images were cropped so that training was only done on the face area, as illustrated in Figure 4.4. Regular data augmentation, described in Section 3.2.1, was used to create a total of 5000 positive training samples both in the 50- and 250-images experiments.

Figure 4.4: The training images were cropped to only contain faces.

The images were then transformed using the trained Star-GAN model, which produced 200 or 1000 transformed images in the two experiments, respectively. The transformed images were added to the two sets of originals, resulting in one augmented dataset with 250 images (50 real, plus 200 StarGAN-transformed) and one with 1250 images (250 real, plus 1000 StarGAN-transformed). These were used to train new Viola-Jones detectors, using the same training procedure as before. Regular data augmentation was still applied to create 5000 training samples from the 250 or 1250 images, respectively. The F1-scores of the resulting detectors, tested on the evaluation datasets from Section 3.1.1, are seen in Table 4.3 and 4.4 for the two experiments, respectively.

From the tables, we see that the detectors' achieved F1-score increased across all the test data in both experiments when Star-GAN augmentation was employed. The benefit of Star-GAN augmentation was the greatest for the small example, and especially for the general FDDB dataset. For the larger experiment, the Elderly dataset benefited the most from Star-GAN augmentation, but in the smaller experiment it benefited the least.

Table 4.3: F1-score of a Viola-Jones detector evaluated on the five evaluation datasets from Section 3.1.1, after training on 50 real images (Baseline) or 50 real plus 200 StarGAN-transformed images (StarGAN-augmented). Regular data augmentation was used to create 5000 training samples in both cases. The training images were hand-picked and did not contain faces with glasses or mustaches, and were not in profile. Δ denotes the difference in F1-score between "Baseline" and "StarGAN-augmented" trainings.

Dataset	Baseline	StarGAN-augmented	Δ
FDDB	0.24	0.51	+0.27
children	0.43	0.57	+0.14
elderly	0.04	0.15	+0.11
glasses	0.02	0.19	+0.17
women	0.54	0.68	+0.14

Table 4.4: F1-score of a Viola-Jones detector evaluated on the five evaluation datasets from Section 3.1.1, after training on 250 real images (Baseline) or 250 real plus 1000 StarGAN-transformed images (StarGAN-augmented). Regular data augmentation was used to create 5000 training samples in both cases. The training images were hand-picked and did not contain faces with glasses. Δ denotes the difference in F1-score between "Baseline" and "StarGAN-augmented" trainings.

Dataset	Baseline	StarGAN-augmented	Δ
FDDB	0.68	0.73	+0.05
children	0.74	0.79	+0.05
elderly	0.30	0.47	+0.17
glasses	0.34	0.41	+0.07
women	0.84	0.87	+0.03

4.2.1 In-context Translation

In the sections above, focus have been on generating or translating cropped images which only contain the object of interest. While this suffices for sliding-window based detection algorithms such as Viola-Jones, many promising object detection algorithms also learn region proposal and require training samples where the object appears in a context [48]. For this reason, it could be useful to also modify object features when the object appears in a context.

In order to demonstrate this, we trained Star-GAN on CelebA with the same settings as above. We then employed a similar (automated) procedure as [45]:

1. Given an image containing faces in a context, detect the faces using a face detector. For this purpose, we used a deep learning based face detector from an open source face recognition library for Python 3 [14].
2. Pad the detected face areas to include approximately as much of the surroundings as CelebA.
3. Cut each head area from the image and re-scale it to fit the StarGAN input requirements.
4. Apply StarGAN to the head-images, to create the transformed faces.
5. Scale back the transformed head-images to the original's in-context dimensions, and paste it back into the original image.

The procedure above is illustrated in Figure 4.5. An example result of running

an image through the pipeline can be seen in Figure 4.6, where the attribute "wearing eyeglasses" has been imposed on all detected faces.

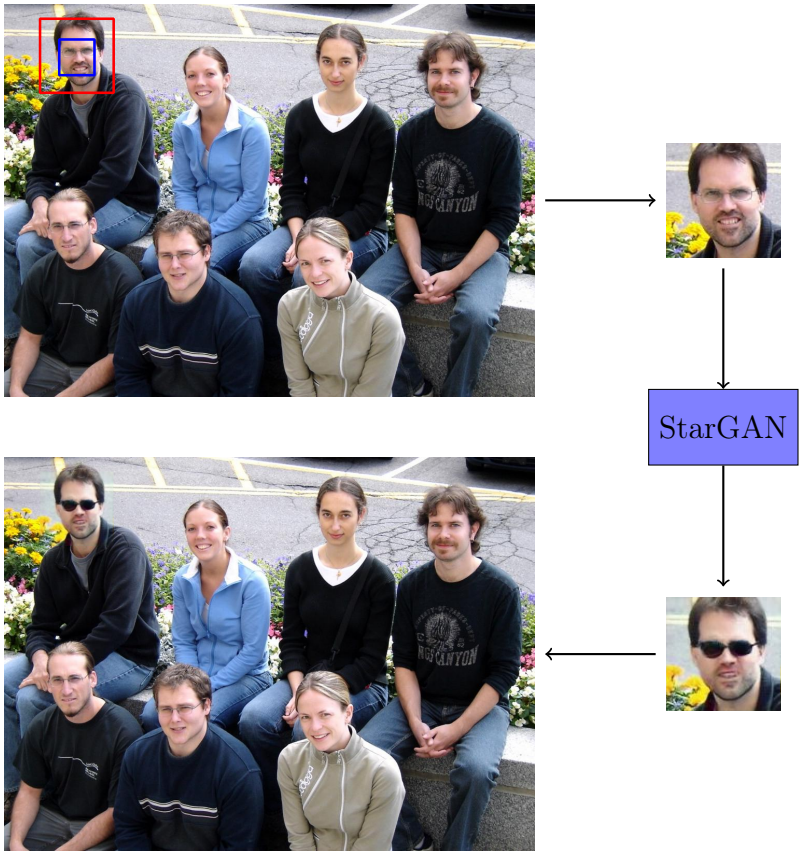
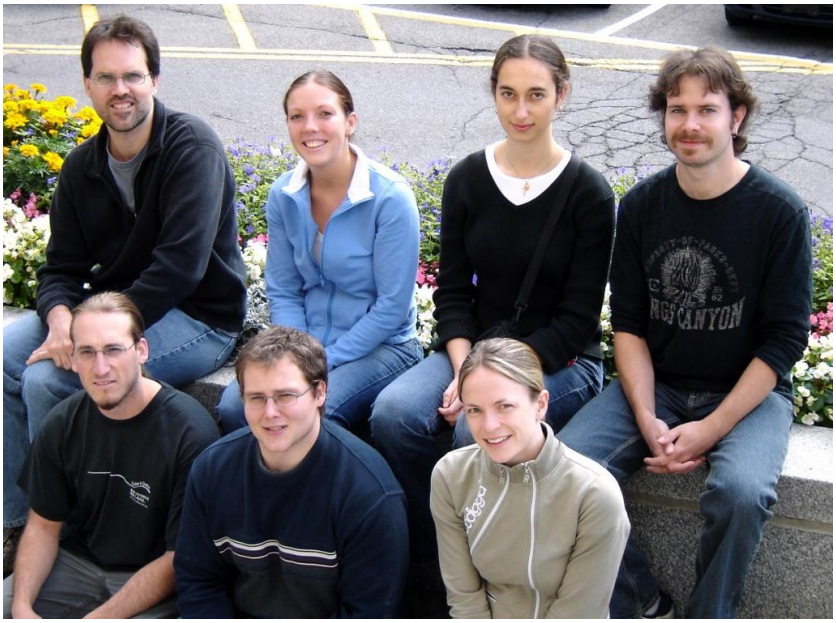
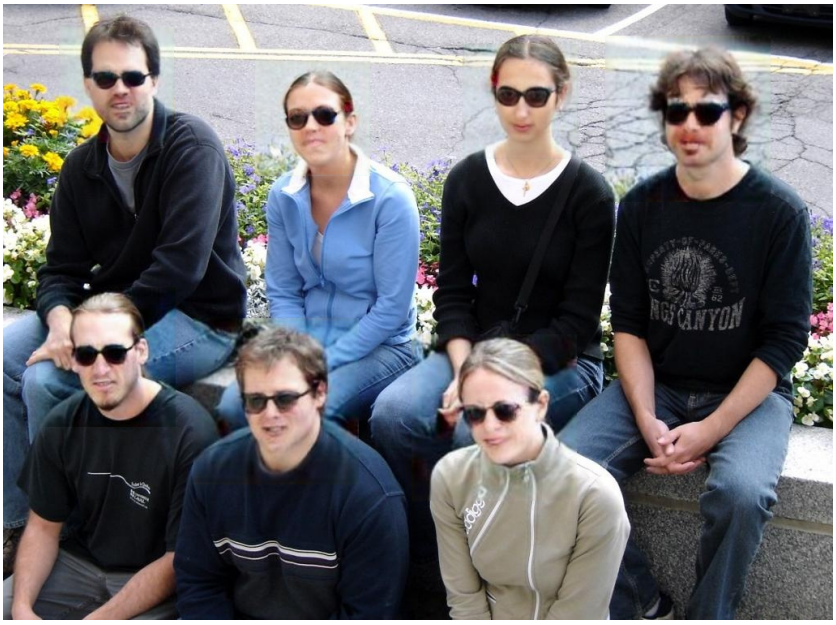


Figure 4.5: Illustration of the procedure employed to perform In-context translation using Star-GAN. First, the face was detected using a face detector and cropped out. Then, Star-GAN was applied to transform the face. Finally, the transformed image was pasted back in its context.



(a) Original image, taken from the WIDER FACE dataset [60].



(b) Added eyeglasses in context using the procedure described in Section 4.2.1.

Figure 4.6: Demonstration of in-context image-to-image translation using Star-GAN.

Chapter 5

Discussion

5.1 Image Generation

Visual inspection of the synthetic images in Section 4.1 reflects the results in [25] of high realism in synthetic images, and does indeed suggest that synthetic images might be useful for training. Figure 4.2 shows that training on real data consistently gives better results than training on the same amount of synthetic data. However, this is actually not surprising since the synthetic data is modeled after the real data. If the model is perfect, it will exactly replicate the distribution of the real data. If it is not perfect, which is the more likely scenario, at least some of the generated samples will not quite resemble the real images. Such samples might be detrimental for the detector training as they will most likely not resemble any real samples.

It is also not surprising that training on more data is generally better than training on less data, as this is a well established phenomenon in machine learning. This leads to a trade-off situation where using a large amount of synthetic data can be better than using a small amount of real data. In Figure 4.2, we see that in our specific setting we needed around 10 times as many synthetic images to get comparable performance to training on real images.

However, provided we have a trained generative model, it is typically just as easy to generate e.g. 5000 images as it is to generate 5. Acquiring more real data, on the other hand, does not become easier. It therefore makes more sense to compare the best scores for synthetic data with different amounts of real data. We can then see there is a cutoff, where it becomes better to

collect n real images than acquire a generative model. From Table 4.1 we see that this cutoff happens around $n = 600$ to 800 images for our specific test case, assuming we could not train a better detector using more than 5000 synthetic images. The reason for this limit is that it becomes impractical to train the Viola-Jones detector with much more than 5000 positive samples.

Another possibility would be to use real data in combination with the generative model to get superior results to using either real or synthetic data. This is beyond the scope of this thesis, but previous work has shown the potential of adding synthetic data to real training data [39, 42, 57, 61, 64] although it has also been shown it can sometimes be detrimental [42, 58].

It should be noted that any numbers presented are highly specific to the GAN model, training data, choice of detector, choice of score metric, and other parameters specific to our experiment. It is hard to infer general behaviour from our experiments, but it seems reasonable to hypothesize that for any reasonable generative model there is a lowest number of real images that performs better than using the generative model to produce a lot of synthetic samples.

In Figure 4.2, we see that retraining the detector using more data does not always result in higher performance score, which leads to the "spikeyness" of the plots. For example, the detector trained on 900 real images gets a lower score on all test datasets than the detector trained on 800 real images. This counter-intuitive result is most likely a product of stochasticity in the detector training. In fact, two separate training runs with the exact same training parameters are not guaranteed to produce the exact same results. This makes it difficult to draw conclusions based on exact numbers. If time allowed, it would be preferable to average each point over many training runs.

We also see that performance differs greatly on the different datasets, with Glasses and Elderly seemingly being the hardest benchmarks, both for real and synthetic images. This indicates a larger mismatch between these test sets and the training data, which can be expected for such specific categories.

When data augmentation is employed, the performance of the detectors trained on synthetic data closes in on the detectors trained on real data, as seen in Table 4.2 – and on the Women dataset, the detector trained on synthetic data reached the same score as the detector trained on real data. This result indicates that synthetic data can indeed be used to train reasonably performing detectors, and might not be much worse compared to real data.

5.2 Image-to-Image Translation

When using Star-GAN for data augmentation, the F1-score of the resulting detectors increased on all evaluation datasets, as seen in Tables 4.3 and 4.4. This might be expected since, as discussed in Section 5.1, increasing the number of training images typically leads to higher performance. However, data augmentation is not quite comparable to simply collecting more data. This can be readily seen in our experiments, where 50 real plus 200 Star-GAN generated training images gives worse results compared to using 250 real images.

Instead, data augmentation should be seen as a way of increasing variance in the data to combat *a priori* biases, as discussed in [51, 59]. Compared to using affine transforms, or other pixel-space transformations, our method should be superior since it introduces variance in deep features, which are higher-level features compared to pixel values. Additionally, unlike DAGAN it is based on labelled features and thus likely more sensible than the features learned by DAGAN [1].

The results indicate that Star-GAN successfully introduced variance in the training data. We also see that the benefits of this method is the largest when data is very scarce, which can be somewhat expected. In our case, the increased variance comes from the features the Star-GAN learned by seeing all of CelebA. Thus, we are indirectly using information from all of CelebA even though we only need access to 50 or 250 real images at the time of training the detector – the rest is encoded in the trained Star-GAN model. Of course, there is a potential for using this augmentation for training samples not within CelebA, but we leave that for future work.

In Table 4.3 we see that in the low-data regime, the performance increase was greatest on the FDDB dataset. This is interesting, because it means that even if we selected a few specific features for augmentation, the benefits were greatest for a very general test case. For the same reason, it is noteworthy that performance was increased on the datasets Children and Women, even though one of the augmented features, "mustache", should have a negative impact on those. Similarly, the performance increase on the Glasses dataset did not stand out considerably, even though that *was* one of the augmented features.

However, in the 250-images example in Table 4.4, the performance increase was greatest on the Elderly dataset, which could be expected to contain a larger proportion of the features "wearing eyeglasses" and "mustache".

It could be expected that augmenting for a specific feature would increase detection on samples with that feature. In that case, this technique could be used to shift a dataset toward a specific bias. More experiments would have to be done, however.

5.2.1 In-context Translation

The in-context image-to-image translation results presented in this report are preliminary and mostly included for demonstration purposes. The results suggest that the image-to-image translation in this report could potentially also be used for detectors which learns region proposal. However, the method used here employing Star-GAN in context might not be optimal; for example, bounding box artifacts are left around the transformed object as can be seen in Figure 4.6b. In [45] they used masking of the object for a smoother transition between the transformed area and the surrounding pixels.

Generation of entirely new samples of an object in a context is a different challenge. In [42], a GAN was trained to generate pedestrians at a specific location of an image by training one discriminator to enforce realism of the generated pedestrian, and another discriminator to make sure the generated pedestrian fit in to the context of the image. A similar procedure would probably be more complicated when the object has specific environmental constraints, such as how a face is usually located at the top of a body. One possibility then could be to have another generator generate the body.

5.3 Future Work

For future work, it would be interesting to further investigate exactly how good performance can be reached when training on only the GAN-generated synthetic data. For example, it could be investigated if there is a better way to sample $\boldsymbol{z}$ for the purpose of generating training data than simply drawing from a Gaussian distribution. Perhaps consciously controlling the bias of generated samples, for example using the method in [46], could be used to mitigate a priori bias in the data used to train the GAN. Another way could be combining the generated images with Star-GAN augmentation.

It could also be of some interest to see how a combination of the generated images with real images would fare. This could be investigated for both the

case when the real images are sampled from the training data of the GAN, and when the real images are from an entirely different dataset.

Future work could also investigate how to use the synthetic data for training deep learning based detectors with region proposal, since this type of detector is likely most relevant for object detection applications in the future. In this work, we began looking at in-context image-to-image translation, but no training using this data was performed. Further developments of this idea are left for future work. Generation of new samples in a context could also be useful to investigate.

5.4 Conclusion

The image quality attainable with Generative Adversarial Networks has increased a lot in recent years, and the results in this thesis suggest that synthetic data generated by GANs can be used to train object detectors with decent results. While it seems like replacing real data with an equal amount of synthetic data always leads to worse performance, one of the advantages of a generative model is that it can be used to easily produce large amounts of synthetic data. In this work we have seen that, using an a priori trained GAN, it is possible to train reasonable detectors using only synthetic data.

We have also seen that image-to-image translation using GANs can be useful for data augmentation. Unlike both traditional data augmentation methods and more recent approaches like DAGAN, this introduces variance in labeled features learned in a supervised manner. For that reason, it should both introduce realistic variance and enable new ways of controlling dataset bias.

The main benefit of using synthetic over real training data is to reduce the need for handling real personal data, which is subject to both regulations such as GDPR and ethical concerns. The methods discussed in this thesis could make it easier to train object detectors and other machine learning algorithms while adhering to these regulations and ethical considerations. Another benefit of the methods discussed is that they could enable more control over dataset bias, which can help reduce the risk of algorithmic discrimination and produce more reliable applications of machine learning.

Bibliography

[1] A. Antoniou, A. Storkey, and H. Edwards. "Data Augmentation Generative Adversarial Networks". In: *arXiv:1711.04340 [cs, stat]* (Nov. 12, 2017). arXiv: 1711.04340.

[2] M. Arjovsky, S. Chintala, and L. Bottou. "Wasserstein GAN". In: *arXiv:1701.07875 [cs, stat]* (Jan. 26, 2017). arXiv: 1701.07875.

[3] C. M. Bishop. *Pattern Recognition and Machine Learning*. New York, USA: Springer Science + Business Media, LLC, 2009.

[4] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[5] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. Van Gool. "Domain Adaptive Faster R-CNN for Object Detection in the Wild". In: *Computer Vision and Pattern Recognition (CVPR)*. 2018.

[6] N. Chinchor. "MUC-4 Evaluation Metrics". In: *Proceedings of the 4th Conference on Message Understanding*. MUC4 '92. Stroudsburg, PA, USA: Association for Computational Linguistics, 1992, pp. 22–29.

[7] Y. Choi. *Official Implementation of StarGAN - CVPR 2018*. original-date: 2017-11-27T01:43:01Z. Nov. 13, 2018. URL: https://github.com/yunjey/stargan (visited on 11/13/2018).

[8] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo. "StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation". In: *arXiv:1711.09020 [cs]* (Nov. 24, 2017). arXiv: 1711.09020.

[9] A. Datta, M. C. Tschantz, and A. Datta. "Automated Experiments on Ad Privacy Settings: A Tale of Opacity, Choice, and Discrimination". In: *arXiv:1408.6491 [cs]* (Aug. 27, 2014). arXiv: 1408.6491.

[10] G. Douzas and F. Bacao. "Effective data generation for imbalanced learning using conditional generative adversarial networks". In: *Expert Systems with Applications* 91 (Jan. 2018), pp. 464–471.

[11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. "The Pascal Visual Object Classes (VOC) Challenge". In: *International Journal of Computer Vision* 88.2 (June 2010), pp. 303–338.

[12] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. "Object Detection with Discriminatively Trained Part-Based Models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.9 (Sept. 2010), pp. 1627–1645.

[13] J. Forge. "The Morality of Weapons Research". In: *Science & Engineering Ethics* 10.3 (July 2004), pp. 531–542.

[14] A. Geitgey. *The world's simplest facial recognition api for Python and the command line*. original-date: 2017-03-03T21:52:39Z. Nov. 13, 2018. URL: `https://github.com/ageitgey/face_recognition` (visited on 11/13/2018).

[15] R. Girshick. "Fast R-CNN". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015 IEEE International Conference on Computer Vision (ICCV). Dec. 2015, pp. 1440–1448.

[16] I. J. Goodfellow. "On distinguishability criteria for estimating generative models". In: *arXiv:1412.6515 [stat]* (Dec. 19, 2014). arXiv: `1412.6515`.

[17] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., 2014, pp. 2672–2680.

[19] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. "Improved Training of Wasserstein GANs". In: *arXiv:1704.00028 [cs, stat]* (Mar. 31, 2017). arXiv: `1704.00028`.

[20] A. Hindupur. *A list of all named GANs!* original-date: 2017-04-14T16:45:24Z. Nov. 26, 2018. URL: `https://github.com/hindupuravinash/the-gan-zoo` (visited on 11/26/2018).

[21] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. "Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). July 2017, pp. 3296–3297.

[22] P. Isola, J. Y. Zhu, T. Zhou, and A. A. Efros. "Image-to-Image Translation with Conditional Adversarial Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). July 2017, pp. 5967–5976.

[23] V. Jain and E. Learned-Miller. *FDDB: A Benchmark for Face Detection in Unconstrained Settings*. UM-CS-2010-009. University of Massachusetts, Amherst, 2010.

[24] T. Karras. *Progressive Growing of GANs for Improved Quality, Stability, and Variation*. original-date: 2017-10-27T09:04:34Z. Oct. 31, 2018. URL: `https://github.com/tkarras/progressive_growing_of_gans` (visited on 10/31/2018).

[25] T. Karras, T. Aila, S. Laine, and J. Lehtinen. "Progressive Growing of GANs for Improved Quality, Stability, and Variation". In: *arXiv:1710.10196 [cs, stat]* (Oct. 27, 2017). arXiv: `1710.10196`.

[26] O. Khalil, M. E. Fathy, D. K. E. Kholy, M. E. Saban, P. Kohli, J. Shotton, and Y. Badr. "Synthetic training in object detection". In: *2013 IEEE International Conference on Image Processing*. 2013 IEEE International Conference on Image Processing. Sept. 2013, pp. 3113–3117.

[27] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv:1412.6980 [cs]* (Dec. 22, 2014). arXiv: `1412.6980`.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (May 24, 2017), pp. 84–90.

[29] M. Kuneva. "Roundtable on Online Data Collection, Targeting and Profiling". Brussels, Mar. 31, 2009.

[30] S. Larsson. "Algorithmic governance and the need for consumer empowerment in data-driven markets". In: *Internet Policy Review* 7.2 (2018), pp. 1–12.

[31] T. A. Le, A. G. Baydin, R. Zinkov, and F. Wood. "Using synthetic data to train neural networks is model-based reasoning". In: *2017 International Joint Conference on Neural Networks (IJCNN)*. 2017 International Joint Conference on Neural Networks (IJCNN). May 2017, pp. 3514–3521.

[32] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (Nov. 1998), pp. 2278–2324.

[33] Y. LeCun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hubbard. "Handwritten digit recognition: applications of neural network chips and automatic learning". In: *IEEE Communications Magazine* 27.11 (Nov. 1989), pp. 41–46.

[34] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. "Microsoft COCO: Common Objects in Context". In: *arXiv:1405.0312 [cs]* (May 1, 2014). arXiv: 1405.0312.

[35] Z. Liu, P. Luo, X. Wang, and X. Tang. "Deep Learning Face Attributes in the Wild". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015 IEEE International Conference on Computer Vision (ICCV). Dec. 2015, pp. 3730–3738.

[36] W. S. McCulloch and W. Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (Dec. 1, 1943), pp. 115–133.

[37] M. Mirza and S. Osindero. "Conditional Generative Adversarial Nets". In: *arXiv:1411.1784 [cs, stat]* (Nov. 6, 2014). arXiv: 1411.1784.

[38] A. Møgelmose, M. M. Trivedi, and T. B. Moeslund. "Learning to detect traffic signs: Comparative evaluation of synthetic and real-world datasets". In: *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012). Nov. 2012, pp. 3452–3455.

[39] A. Odena. "Semi-Supervised Learning with Generative Adversarial Networks". In: *arXiv:1606.01583 [cs, stat]* (June 5, 2016). arXiv: 1606.01583.

[40] M. Ohlsson. *Lecture Notes on Introduction to Artificial Neural Networks and Deep Learning (FYTN14/EXTQ40)*. Lund: Computational Biology, Biological Physics, Department of Astronomy, and Theoretical Physics, Lund University [Unpublished], 2017.

[41] R. M. S. d. Oliveira, R. C. F. Araújo, F. J. B. Barros, A. P. Segundo, R. F. Zampolo, W. Fonseca, V. Dmitriev, and F. S. Brasil. "A System Based on Artificial Neural Networks for Automatic Classification of Hydro-generator Stator Windings Partial Discharges". In: *Journal of Microwaves, Optoelectronics and Electromagnetic Applications* 16.3 (Sept. 2017), pp. 628–645.

[42] X. Ouyang, Y. Cheng, Y. Jiang, C.-L. Li, and P. Zhou. "Pedestrian-Synthesis-GAN: Generating Pedestrian Data in Real Scene and Beyond". In: *arXiv:1804.02047 [cs, stat]* (Apr. 5, 2018). arXiv: 1804.02047.

[43] C. P. Papageorgiou, M. Oren, and T. Poggio. "A general framework for object detection". In: *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*. Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271). Jan. 1998, pp. 555–562.

[44] A. R. Pathak, M. Pandey, and S. Rautaray. "Application of Deep Learning for Object Detection". In: *Procedia Computer Science* 132 (2018), pp. 1706–1717.

[45] A. Pumarola, A. Agudo, A. M. Martinez, A. Sanfeliu, and F. Moreno-Noguer. "GANimation: Anatomically-aware Facial Animation from a Single Image". In: *arXiv:1807.09251 [cs]* (July 24, 2018). arXiv: 1807.09251.

[46] A. Radford, L. Metz, and S. Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *arXiv:1511.06434 [cs]* (Nov. 19, 2015). arXiv: 1511.06434.

[47] *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance)*. May 4, 2016.

[48] S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *arXiv:1506.01497 [cs]* (June 4, 2015). arXiv: 1506.01497.

[49] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. "Improved Techniques for Training GANs". In: *arXiv:1606.03498 [cs]* (June 10, 2016). arXiv: 1606.03498.

[50] Y. Sasaki. "The truth of the F-measure". In: *Teach Tutor Mater* (2007), p. 6.

[51] P. Simard, D. Steinkraus, and J. Platt. "Best practices for convolutional neural networks applied to visual document analysis". In: *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.* Seventh International Conference on Document Analysis and Recognition. Vol. 1. Edinburgh, UK: IEEE Comput. Soc, 2003, pp. 958–963.

[52] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going deeper with convolutions". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). June 2015, pp. 1–9.

[53] Y. H. Tseng and S. S. Jan. "Combination of computer vision detection and segmentation for autonomous driving". In: *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. 2018 IEEE/ION Position, Location and Navigation Symposium (PLANS). Apr. 2018, pp. 1047–1052.

[54] P. Viola and M. Jones. "Rapid object detection using a boosted cascade of simple features". In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001. Vol. 1. 2001, I–511–I–518 vol.1.

[55] H. Wang, Q. Wang, M. Gao, P. Li, and W. Zuo. "Multi-scale Location-aware Kernel Representation for Object Detection". In: *arXiv:1804.00428 [cs]* (Apr. 2, 2018). arXiv: 1804.00428.

[56] X. Wang, M. Wang, and W. Li. "Scene-Specific Pedestrian Detection for Static Video Surveillance". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.2 (Feb. 2014), pp. 361–374.

[57] X. Wang, Z. Man, M. You, and C. Shen. "Adversarial Generation of Training Examples: Applications to Moving Vehicle License Plate Recognition". In: *arXiv:1707.03124 [cs]* (July 11, 2017). arXiv: 1707.03124.

[58] D. Wessman and P. Andersson. "Generation of Artificial Training Data for Deep Learning". In: *Master's Theses in Mathematical Sciences* (2018).

[59] L. S. Yaeger, R. F. Lyon, and B. J. Webb. "Effective Training of a Neural Network Character Classifier for Word Recognition". In: *Advances in Neural Information Processing Systems 9*. Ed. by M. C. Mozer, M. I. Jordan, and T. Petsche. MIT Press, 1997, pp. 807–816.

[60] S. Yang, P. Luo, C. C. Loy, and X. Tang. "WIDER FACE: A Face Detection Benchmark". In: *arXiv:1511.06523 [cs]* (Nov. 20, 2015). arXiv: 1511.06523.

[61] J. Yu, D. Farin, C. Krüger, and B. Schiele. "Improving person detection using synthetic training data". In: *2010 IEEE International Conference on Image Processing*. 2010 IEEE International Conference on Image Processing. Sept. 2010, pp. 3477–3480.

[62] Y. Zhang, Y. Bai, M. Ding, Y. Li, and B. Ghanem. "W2F: A Weakly-Supervised to Fully-Supervised Framework for Object Detection". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[63] J. Zhao, T. Wang, M. Yatskar, V. Ordonez, and K.-W. Chang. "Men Also Like Shopping: Reducing Gender Bias Amplification using Corpus-level Constraints". In: *arXiv:1707.09457 [cs, stat]* (July 28, 2017). arXiv: 1707.09457.

[64] Z. Zheng, L. Zheng, and Y. Yang. "Unlabeled Samples Generated by GAN Improve the Person Re-identification Baseline in vitro". In: *arXiv:1701.07717 [cs]* (Jan. 26, 2017). arXiv: 1701.07717.

[65] J. Y. Zhu, T. Park, P. Isola, and A. A. Efros. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017 IEEE International Conference on Computer Vision (ICCV). Oct. 2017, pp. 2242–2251.

Master's Theses in Mathematical Sciences 2018:E75
ISSN 1404-6342

LUTFMA-3370-2018

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/