CHALMERS
UNIVERSITY OF TECHNOLOGY

Deep Learning for Fashion Analysis

Generative modelling for semantic segmentation data

Master's thesis in Complex Adaptive Systems

Marie Korneliusson

Department of Physics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Master's thesis 2019:TIFX05

Deep Learning for Fashion Analysis

Generative modelling for semantic segmentation data

Marie Korneliusson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2019

Deep Learning for Fashion Analysis
Generative modelling for semantic segmentation data
Marie Korneliusson

Supervisor: Olof Mogren, RISE AI
Examiner: Mats Granath, Department of Physics

Master's Thesis 2019:TIFX05
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Generated image and corresponding generated semantic segmentation map on top of the generated image.

Typeset in LaTeX
Gothenburg, Sweden 2019

Deep Learning for Fashion Analysis
Generative modelling for semantic segmentation data
Marie Korneliusson
Department of Physics
Chalmers University of Technology

Abstract

Using semantic segmentation algorithms to automatically classify each pixel of an image, could imply great benefits for the fashion industry but also for the use of society. Semantic segmentation algorithms can for example be used within the fashion industry by predicting trends on social media or by robots within health care to help people get dressed. However, performance of semantic segmentation algorithms are dependent on a large amount of annotated training data. The fashion industry in particular, is an area where machine learning algorithms are not as well developed as in many other fields, and the amount of training data is thus limited. Therefore, the purpose of this thesis was to investigate if it is possible to use deep learning for generative modeling to increase the amount of training data in the fashion domain. The results showed that it is possible to use generative adversarial networks (GANs) to generate pairs of images and corresponding pixel wise annotations.

Keywords: Generative Adversarial Networks, Neural Networks, Machine Learning, Deep Machine Learning, Semantic Segmentation, Fashion.

Acknowledgements

First of all I would like to thank my supervisor Olof Mogren, for his support and guidance throughout the project, which made this project possible. I would also like to thank him for having me as a master thesis student. John Martinsson and Edvin Listo Zec thanks for supporting, for chairing ideas, and for including me in the group. A special thanks to John for helping me with testing the generated images on the semantic segmentation task. In addition I want to thank Malin Korneliusson for helping with designing the network architecture figure for the report, and Teodor Norrestad for helping with proofreading.

Marie Korneliusson, Gothenburg, June 2019

Contents

List of Figures

List of Tables

1

Introduction

Deep learning algorithms have received a lot of attention during the last couple of years. Great results in a variety of areas have enhanced the interest and given fuel to further research [22]. The success factor, deep nets, trained on GPUs [21]. However, deep models are complex and use a lot of parameters. Therefore, to make the model generalize and to have capacity to train all the parameters, it is crucial to have a sufficient amount of available training data [46]

One of the areas where great results have been recorded by the utilisation of deep learning algorithms is the task of semantic segmentation [9]. The task of semantic segmentation, is the task of understanding the semantics of an image. In particular, semantic segmentation algorithms aim to classify each pixel of an image into a set of predefined classes [30]. Classifying each pixel of an high dimensional image, implies a certain task complexity. To compensate for the task complexity a large amount of trainable parameters are required. Therefore, to have the ability to train all parameters a large amount of training data is needed. However, the training data used for semantic segmentation includes images and corresponding semantic segmentation maps, i.e a map including a class label for each of the image pixels. Labeling that type of data could both be expensive and time consuming.

The ability to automatically classify each pixel in an image have a great potential to be applied in various of areas. For instance using semantic segmentation algorithms for understanding the semantic of fashion images, could imply great benefits for the fashion industry but also for the use of society. Semantic segmentation algorithms can for example, be used within the fashion industry by predicting trends on social media, or by robots within health care to help people get dressed.

Unfortunately is, the fashion industry in particular, an area in which deep machine learning have not received as much attention as in many other fields. The amount of training data available for training a semantic segmentation algorithm is thus limited. In this thesis we address this problem, and we aim to investigate what can be done with already available training data. In particular, the goal is to investigate if it is possible to use generative modelling focusing on Generative Adversarial Networks [10] to increase the amount of training data.

1.1 Aim and Research Questions

The aim of this master thesis is to investigate and examine the possibility in generating new training data, that can be used to train a semantic segmentation algorithm on fashion images. In particular, the idea is to examine the possibility to generate training data and corresponding semantic segmentation maps using Generative Adversarial Networks (GANs). To fulfill the aim of the thesis the following research questions will be answered.

(R_1) Is it possible to use GANs to generate realistic pairs of images and semantic segmentation maps?
(R_2) Comparing generated images with images generated together with corresponding semantic segmentation maps, how will the quality of generated images be affected?
(R_3) What will be that variation of generated samples, are we able to generate images that do not already exists in the training set?

1.2 Limitations

Generative adversarial networks is one technique used for implementing generative models. There are other examples of generative models in the literature, that has provided good results, for example deep Boltzmann machines [35] and generative stochastic models [3]. This thesis will be limited to investigate the characteristic of using the GAN framework. Further, the thesis is limited to generating fashion data.

1.3 Thesis outline

The outline of this thesis is as follow. The thesis starts with a background to deep neural networks. The chapter aims to introduce the deep learning related aspects of this thesis. Therefore, multi layer perceptrons, convolutional neural networks, supervised learning and finally the task of semantic segmentation is described. In chapter 3, the theory of generative image modeling using deep leaning is provided. In particular we introduce the theory behind the main technique of this thesis, namely, Generative Adversarial Networks. In chapter 4 the method is described, including the used data set, the model architectures and the evaluation metrics. In chapter 5 the results are presented, where the aim is to answer all the stated research questions. Finally the thesis is concluded with a discussion about the outcomes and future work.

2
Background

The aim of this thesis was to generate fashion images and corresponding segmentation maps using deep neural networks trained under the GAN framework. In this chapter we therefore provide a short background regarding learning in deep neural networks and the task of semantic segmentation.

2.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) were first introduced in [28], where the original idea was to model the animal brain. The model paved the way for a new research area. Namely, designing algorithms where the rules are not set by a human expert but instead computationally learned by the network's interaction with the environment [17]. Today the animal brain interpretation of the ANN model is far from realistic.

The popularity of ANNs has been fluctuating. Short after the introduction it was showed that ANNs were limited by a great amount of computations [29]. It was computationally infeasible to train a large net at the time. However further advances both in algorithms such as back-propagation [34] and in hardware developments have made it possible to train larger nets, which we today call deep neural networks.

2.2 Deep Neural Networks

Deep neural networks (DNNs) are artificial neural networks with multiple layers, whose complex architectures can successfully capture dependencies in high dimensional data [24]. Multi layer perceptons (MLPs) and convolotional neural networks (CNNs) are two of the most well known DNN architectures and are also the two architectures used in this thesis. Below, both of them are described briefly.

2.2.1 Multi Layer Perceptrons

Multi Layer Perceptrons (MLPs) are artificial neural networks with at least one hidden layer (in addition to it's output layer). MLPs are typically illustrated using a graph (see fig. 2.1), with links connecting the nodes, which by the human brain interpretation is a realization of synapses connecting to neurons.

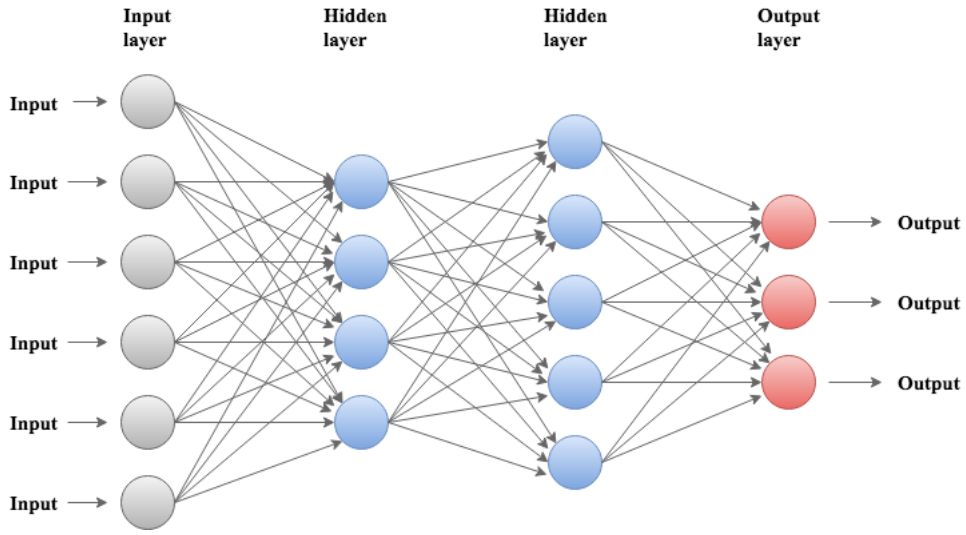

Figure 2.1: Graph illustrating a multi layer perceptron, with one input layer, two hidden layers and one output layer.

Mathematically the model is defined by the following. Let $\mathbf{y}^{(l+1)}$ denote the activation or output of each neuron in layer $l+1$. Further let $\mathbf{y}^{(0)} = \mathbf{x}$, where $\mathbf{x}$ is the input vector, then for each layer the activations are given by:

$$\mathbf{y}^{(l+1)} = g(\ \mathbf{W}\mathbf{y}^{(l)} + \mathbf{b}\ ), \tag{2.1}$$

where $\mathbf{W}$ denotes the connection strengths of layer l and $l+1$ also called weights, $\mathbf{b}$ denotes the bias vector for the particular layer and g denotes the activation function.

2.2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) is a DNN architecture that was introduced in [23] with the aim of improving the current state of image recognition algorithms. CNNs differs from MLPs by the convolutional operator that they employ, which enable the network to capture spatial local dependencies in images.

The convolutional operator is illustrated in fig. 2.2 and is defined by first specifying three hyper-parameters, namely, the kernel size k, the stride s, and the number of feature maps f. Then given an input $\mathbf{x}$ with dimension $(W \times H \times D)$, where W, H and D denotes the input width, height and depth or number of channels respectively, we can now define the convolutional operator by eq. (2.2), for $s = 1$.

$$y^f_{i,j} = \sum_{m=1,n=1}^{k} x_{i+m,j+n} w^f_{m,n}\ \forall i, j, f, \tag{2.2}$$

where $\mathbf{w}^f \in (k \times k)$ denotes the weights in each feature map and $\mathbf{y} \in (W \times H \times f)$ denotes the output from the convolutional operator. We hence note that the number of feature maps determines the dimension of the output in a convolutional layer.

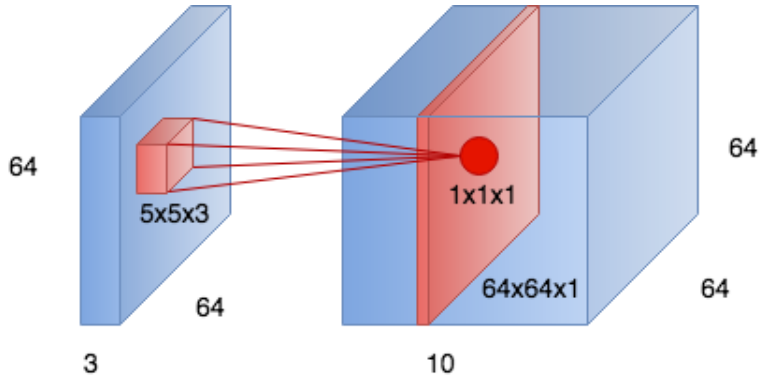

Figure 2.2: Illustrating a convolutional operator with kernel size 5, stride 1 and 10 feature maps acting on an input image of dimension (64×64×3)

2.3 Supervised Learning in Deep Neural Networks

From section 2.2 we note that DNNs can be seen as a function f_θ that maps an input $\mathbf{x}$ to an output $\mathbf{y}$, where θ denotes the parameters of the function such as weights and biases. In deep learning we aim to learn the parameters θ such that f_θ is the best function for solving the given task. There are different types of learning that can be used for finding the best values of the parameters in a DNN [17]. In this thesis we will focus on the learning type called supervised learning.

In supervised learning the parameters or weights are learned by the aim of minimizing a task specific error $e(\mathbf{y}, \mathbf{t})$, which defines a measure of the distance between the networks predictions $\mathbf{y}$ and the given target values $\mathbf{t}$. Hence, supervised learning requires a data set that consists of pairs of inputs $\mathbf{x}$ and targets $\mathbf{t}$.

After defining the error measure $e(\mathbf{y}, \mathbf{t})$ the learning procedure is as follows, we feed the inputs $\mathbf{x}$ to through the DNN f_θ which yields the networks prediction $\mathbf{y}$. If the network prediction $\mathbf{y}$ and the target $\mathbf{t}$ are equal, no updates are needed. If they are not equal the parameters are updated in the direction such that the error is minimized. This is typically done by using the back-propagation algorithm [34], which relies on updating each parameter using gradient decent. Hence for a given parameter $w_{i,j}$ the update is determined by,

$$w_{i,j} = w_{i,j} - \alpha \frac{\partial e(\mathbf{y}, \mathbf{t})}{\partial w_{i,j}}, \tag{2.3}$$

where α is called the learning rate, a parameter whose optimal value is dependent on the particular task and and has been shown to have great influence on convergence [37].

2.4 Semantic Segmentation using Deep Neural Networks

Semantic segmentation is the task of understanding the semantics of an image. Classical recognition tasks include image classification and object detection. Semantic segmentation is related to both the image classification and object detection and aims to classify each pixel of the image. The transition from classifying the entire image to classify each pixel comes with an increase of problem complexity. However, implementing convolutional neural networks (CNNs) has been shown beneficial for the task [9]. In particular, the results have rapidly increased after the introduction of the fully convolutional network (FCN) [25], which employs only convolutional layers to well known image classification models such as AlexNet [21], VGG net [39] and GoogLeNet [40]. The FCN model has been used as powerful baseline system which has allowed for further advances on the task[12].

The task of semantic segmentation in the supervised setting can be formulated by the following: Semantic segmentation aims to find the parameters θ of the function f_θ that maps an image $\mathbf{x} \in \mathbb{R}^{(W \times H \times 3)}$ to a semantic segmentation map $\mathbf{y} \in \mathbb{R}^{(W \times H \times C)}$ such that cross entropy over each pixel eq. (2.4) is minimized,

$$\min_{\theta} \sum_{n \in N} \mathbf{t}^n log(\mathbf{y}^n), \tag{2.4}$$

where $W, H, 3$ denotes the image width, height and colour depth i.e red, green and blue in pixels, C denotes the number of predefined classes, N is the number of images in the data set and $\mathbf{t} \in \mathbb{R}^{(W \times H \times C)}$ denotes the target or ground truth semantic segmentation map, i.e the pixel-wise annotation of the image.

From eq. (2.4) we note that solving semantic segmentation in a supervised setting requires a data set that consists of images and corresponding ground truth semantic segmentation maps. Annotating the semantic segmentation map of an image is expensive which hence limits the available amount of training data. In fig. 2.3, an example of the required training data is illustrated.

Figure 2.3: Image illustrating semantic segmentation. The left figure shows an input image. The middle figure shows the semantic segmentation map highlighting the dress, pants and footwear classes. The right figure shows the semantic segmentation map on top of the image.

3

Generative Image Modeling

In this chapter we give an overview of the theory regarding generative image modelling using deep learning, staring with a general discussion of generative modelling using deep learning. Then continuing with discussing the main technique used in this work namely Generative Adversarial Networks.

3.1 Generative Modeling Using Deep Learning

Generative modeling refers to modeling that aims to learn the distribution that is very likely to have generated the observed data. For example, consider a data set that consists of a feature space X with corresponding targets T. For that example, a generative model would aim to learn the joint distribution $P(X, T)$ that is most likely to have generated the observed pairs (X, T). The counter part of a generative model is a discriminative model. The aim of a discriminative model is to learn the conditional distribution over the data set. Considering the same example as for the generative model the discriminative model would aim to learn $P(T|X = x)$. Both a discriminative and a generative model can learn to model targets given input data, however in the case of a generative model the information learned will be richer, we will not only learn the conditional distribution, instead we will learn the whole distribution of the data set.

The traditional approach of learning the distribution most likely to have generated the observed data, includes, defining a parametric family of densities $(P_\theta)_{\theta\in\mathbb{R}}$ and then finding the parameter θ that maximizes the log-likelihood [1] :

$$\max_\theta \frac{1}{n}\sum_{i=1}^{n} logP_\theta(x^i), \tag{3.1}$$

where $(x)_i^n$ are observed samples. In the limit of infinite observations x, the log-likelihood eq. (3.1) tends to the negative Kullback-Leibler divergence eq. (3.2) [38]. Implying, asymptotically finding the maximum log-likelihood corresponds to minimizing the Kullback-Leibler divergence between the real data distribution P_r and the parameterized distribution P_θ, i.e minimizing $KL(P_r|P_\theta)$:

$$\int_\chi P_r(x)log(\frac{P_r(x)}{P_\theta(x)})dx \tag{3.2}$$

Minimizing the KL divergence is a well defined problem under two assumptions. Assumption one, the model distribution P_θ exists, and two, the real data distribution

P_r and the model distribution P_θ have a non-negligible intersection [1]. However, according to [27], it is very unlikely that model distribution and the real data distribution has non-negligible intersection. Implying that the KL divergence is infinite. Classical machine learning techniques for generative modelling deals with the problem by adding a noise term to the observed data, typically a Gaussian [1].

The idea of adding a noise term to the observed data from the real distribution leads us to the main topic of this thesis, namely modeling the distribution over segmented images. According to [44], the optimal noise term to add to an image normalized between $[0, 1]$, is a noise term with a standard deviation 0.1. Adding a noise term with standard deviation 0.1 to an image ranging between $[0, 1]$ results in blurry images, implying a drastic decrease of sample quality. Hence classical machine learning techniques can not solve the problem of generating sharp images. Luckily, there are other techniques for generating images, and in the next section we will introduce one technique that has successfully been used for image generation [4, 18, 19, 47], and also is the fundamental technique of this thesis, namely Generative Adversarial Networks.

3.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) is a framework used to train a generative model. The algorithm was first introduced by Ian Goodfellow in 2014 [10]. The main idea is to train a generative model in supervision of a discriminative model, during a two player min max game. The role of the generator is to fool the discriminator that the generated samples come from the real data distribution rather than from the generators. The aim of the discriminator is to discriminate between real and generated samples. Often the game is explained by considering the analogue of the generative model being a team of counterfeiters aiming to produce fake currency and use it without detection, whereas the discriminative model plays the role of the police, aiming to detect the fake currency.

Mathematically the GAN game can be described by the following, let $G(\mathbf{z}, \theta_g)$ be a generative model with parameters θ_g that maps the input $\mathbf{z}$ drawn from a prior distribution $p_{\mathbf{z}}(z)$ to a space following the distribution $p_{\mathbf{g}}$. Let $D(\mathbf{x}, \theta_d)$ denote a discriminative model with parameters θ_d, that outputs a scalar value representing the probability that an input $\mathbf{x}$ comes from the real data space p_{data} rather than the generated $p_{\mathbf{g}}$. D and G will be considered as deep neural networks throughout this work. Then the GAN game can be described as finding the min max of the value function $V(G, D)$:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{data}}[log(D(x))] + \mathbb{E}_{\mathbf{z}\sim p_z}[log(1 - D(G(z)))] \tag{3.3}$$

In [10] it is shown that the theoretical min max optimum of the function $V(G, D)$ eq. (3.3) is obtained when $p_{data} = p_{\mathbf{g}}$. They derive this result using two main steps. Step one yields the expression for the optimal discriminator D^* for any fixed generator G. Step two shows that given the expression for D^* eq. (3.3) simplifies to

optimizing the Jensen-Shannon divergence between the real data distribution p_{data} and the generated data distribution $p_{\mathbf{g}}$,

$$JSD(p_{data}|p_{\mathbf{g}}) = KL(p_{data}|p_{\mathbf{a}}) + KL(p_{\mathbf{g}}|p_{\mathbf{a}}) \tag{3.4}$$

where KL refers to the Kullback-Leibler divergence eq. (3.2) and $p_{\mathbf{a}} = \frac{p_{data}+p_{\mathbf{g}}}{2}$. This concludes the proof since the Jensen-Shannon divergence has the nice property of being strictly positive and zero iff $p_{data} = p_{\mathbf{g}}$. Hence the optimal solution of eq. (3.3) is obtained when $p_{data} = p_{\mathbf{g}}$, the state when the discriminator D can not discriminate between generated samples $\hat{x} = G(z)$ and samples x drawn from the true distribution.

The parameters θ_d and θ_g that correspond to the min max optimum of $V(G, D)$ are typically found using back-propagation based on gradient descent, where the parameters of G and D are updated in the direction of the gradients of $V(G, D)$ with respect to each parameter. A formal explanation of the procedure of finding the parameters of G and D is presented in Algorithm 1, and in [10] it is shown that Algorithm 1 optimizes eq. (3.3).

Algorithm 1 GAN training algorithm.

Initialize D and G with random weights θ_d and θ_g.
for number of training steps **do**
Sample minibatch of m noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from noise prior $p_g(z)$.
Sample minibatch of m samples $\{x^{(1)}, ..., x^{(m)}\}$ from data distribution $p_{data}(x)$.

Update the discriminator parameters by going gradient ascent using:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\ logD(x^{(i)}) + log(1 - D(G(z^{(i)})))\ ]$$

Update the generator parameters by going gradient descent using:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} log(1 - D(G(z^{(i)})))$$

end for

3.2.1 Training Instability

Recent work has shown that a generator trained using the GAN framework can successfully capture a given data distribution, and generate excellent photo realistic samples [18, 19, 4]. However, training a generator using the GAN framework is remarkably difficult [27]. Training instability, non convergence and slow learning are examples of well reported problems [27, 36, 11]. How to solve these problems are still open research questions. This section will give a short summary of commonly observed problems and suggested solutions.

The training instability and the non converging behaviour of GANs are due to several factors, and most of these factors are still not known [27]. Anyhow, as remarked in [10] one of the factors leading to training instability, is that the GAN objective eq. (3.3) may fail to give the generator sufficient gradients. Early in training the generator is weak, implying that the discriminator finds it easy to distinguish between real and fake data samples, i.e we have a "perfect" discriminator. Intuitively it seems desirable to have a perfect discriminator, since intuitively one may argue that a perfect discriminator would yield a perfect signal to the generator. Further, it has been shown that a perfect discriminator in theory should result in finding the desired min-max optimum [10, 27]. Nevertheless, in practice it has been observed that as the discriminator gets better the generator gets remarkably worse [27].

One reason to the observed behaviour was pointed out in [10], namely that the second term in the GAN objective eq. (3.3), $\mathbb{E}_{\mathbf{z}\sim p_z}[log(1-D(G(z)))]$ may saturate when the discriminator is perfect. It saturates as a consequence of the flat sigmoid (logistic) function, typically used as activation function for the output layer of the discriminator. However, as suggested in [10] the GAN objective eq. (3.3) can be changed from minimizing $\mathbb{E}_{\mathbf{z}\sim p_z}[log(1 - D(G(z)))]$ to maximizing $\mathbb{E}_{\mathbf{z}\sim p_z}[log(D(G(z)))]$. This change of objective function is often referred to as the "log trick". In practice this provides stronger gradients leading to less training instability. Moreover, the change in the objective has the same theoretical min-max optimum as the original GAN objective eq. (3.3).

The new objective function may help training for some applications. However, empirical results show that also the new objective function may fail to provide sufficient gradients for the generator. The reason was investigated in [27], they show that optimizing the new objective function is equivalent to minimizing $KL(p_{\mathbf{g}}|p_{data}) - 2JSD(p_{\mathbf{g}}|p_{data})$, and argue that this objective function during stable training assigns extremely high cost for generating unrealistic samples. However, they also show that gradient updates for this objective are Cauchy distributed with zero mean and infinite variance, which holds as reason for the observed training instability. By this observation the authors of [27] suggest to change the objective to a more soft metric, and propose to use the Wasserstein distance also called the earth mover distance:

$$|\,\mathbb{E}_{\mathbf{x}\sim p_{data}}[f(x)] - \mathbb{E}_{\mathbf{z}\sim p_z}[f(z)]| \tag{3.5}$$

Theoretically the benefits of optimizing the Wasserstein distance are as follows: The Wasserstein distance is softer compared to the Kullback-Leibler divergence and the Jensen-Shannon divergence in the sense that it will provide gradients even if the distributions are disjoint. In addition the Wasserstein distance gives a measure of the closeness of the distributions. In [1] empirical results from evaluating the performance of using Wasserstein distance objective are presented, those results shows that samples obtained while using the Wasserstein distance can in many cases out-preform results obtained using the original GAN objective.

3.2.2 Mode Collapse

Training instability makes it remarkably difficult to train a generator under the GAN framework, however, failure modes can occur also while the training is stable. Mode collapse is one of those failure modes. During a mode collapse the parameters of the generator starts to generate a single mode, i.e a single images. The reason, while the discriminator discriminates between samples individually no gradient feedback is given to tell the generator to generate within a variation similar to training data. This can result in a situation where the generator starts to generate a single mode. A mode which the discriminator finds likely to belong to the real data distribution. After the collapse the discriminator learns that a single mode is generated and starts pushing it around forever [36].

Suggested extensions of the GAN algorithm that deal with the problem can be found in the literature, where the most straight forward techniques rely on extending the discriminator such that it has the ability to discriminate between several samples in combination. Applying batch normalization [16] in the discriminator has therefore been shown to increase the performance [32]. In [36] they further extend on the idea of discriminating between several samples. They introduce a new technique called mini batch discrimination. The idea is to still classify each sample as real or generated, but they allow the classification decision to use information about other samples in the same batch. This is implemented by adding an extra layer in the end of the discriminator with trainable parameters. The output from that layer represent the statistics over that batch, and are used internally in the discriminator during the classification decision.

The disadvantage with the mini batch discrimination technique is that the technique includes an introduction of new parameters to learn, which might slow down learning. In [18] a technique that does not include adding new trainable parameters is used, they call the technique standard deviation discrimination, and they argue that their technique both simplify the computation and implementation but also improves the variation of generated samples compared to mini batch discrimination. The idea of standard deviation discrimination is simple. They calculate the standard deviation for each feature and each pixel over the entire mini batch, then they calculate the average over those standard deviations, resulting in a single number. The average standard deviation is then concatenated into a feature map, resulting in one constant feature map, which is added at one layer in the discriminator network. Hence the technique results in the discrimination decision being based on statistics from all the samples in the mini batch.

4

Methods

In this chapter the method used for the experimental setup of this thesis will be described. The chapter starts with a description of the used data set. Next the models architectures are described, followed by a section that provides information regarding the used evaluation metrics. The chapter is concluded with a section where all the described elements are put together for describing the entire training procedure.

4.1 ModaNet - Data set

Training a generator for image generation under the GAN framework requires a great amount of data. As the aim of this thesis is to generate fashion images and corresponding semantic segmentation maps, a large data set including both images and semantic segmentation maps was required. Therefor we use a data set called ModaNet [48] which includes 55.176 fully annotated street-fashion images.

Figure 4.1: Subset of samples in ModaNet. The upper panel shows examples of RGB images in ModaNet. The lower panel shows the corresponding semantic segmentation map to each image.

The images in ModaNet has been collected such that the images in the data set only contain one person in each image. The images has also been chosen to ensure high diversity in human poses, the data set is not limited to the frontal pose and do both includes full body and half body poses. All images in ModaNet has an equal resolution of 400 pixels width and 600 pixels height. As a consequence of all images having the same resolution, some of the images has been padded, by adding a white panel to the image. Further all images are represented in the RGB format, meaning that they are represented using three channels one for each of the color

Table 4.1: Classes and sub-classes in ModaNet. The sub-classes describes which attributes that each class contains of.

	Classes	Sub-Classes
1	bag	bag
2	belt	belt
3	boots	boots
4	footwear	footwear
5	outer	coat, jacket, suit, blazers
6	dress	dress, t-shirt dress
7	sunglasses	sunglasses
8	pants	pants, jeans, leggings
9	top	top, blouse, t-shirt, shirt
10	shorts	shorts
11	skirt	skirt
12	headwear	headwear
13	scarf&tie	scarf,tie

modes red, green, and blue. Hence each image is a three dimensional array with a size of $(W \times H \times 3)$, where W denotes the width and H denotes the image height. In the upper panel of fig. 4.1 examples of images from ModaNet are shown.

All images in ModaNet are fully annotated by human annotators. This means that each image has a corresponding ground truth semantic segmentation map, i.e each pixel is labeled into one of the 13 predefined classes. Where each class represent a unique fashion attribute and some of the classes represents a union over sub-classes of fashion attributes. The different classes and the sub-classes are shown in Table 4.1. In fig. 4.2 the proportion of instances per class in the ModaNet data set is shown, note that the figure implies a certain class imbalance.

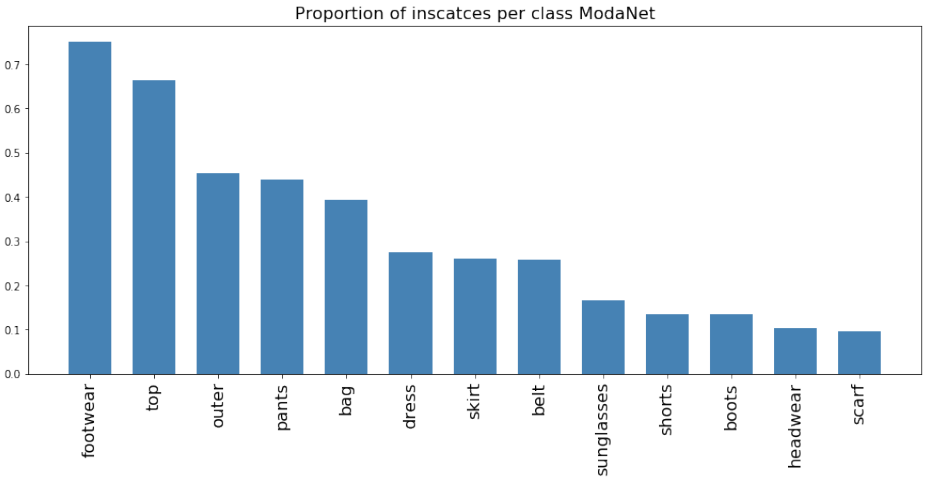

Figure 4.2: Proportions of instances per class in descending order in ModaNet data set.

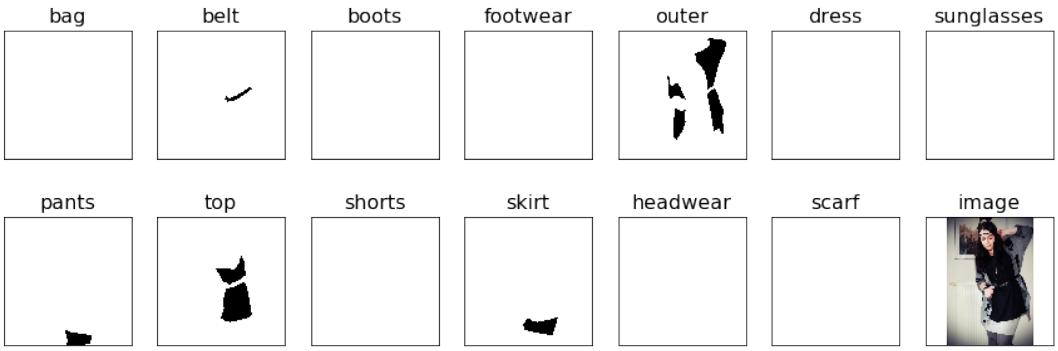

Figure 4.3: Example of an one-hot encoded semantic segmentation map, i.e one map per channel. The figure in the lower right corner shows the corresponding image.

The semantic segmentation maps can be represented in different formats. Either they can be represented as a two dimensional array with the size of $(W \times H)$ taking integer values in the range $(0, .., C)$, where $C = 13$ in the case of ModaNet. Each integer indicates one of the categories shown in table 4.1 and 0 indicates background. The two dimensional representation of the ModaNet semantic segmentation maps are illustrated in the lower panel of fig. 4.1. The two dimensional representation is not used during training, the reason, fashion classes do not have a natural ordering, therefore we can not learn the relations using a gradient based algorithm. Instead we use a one-hot encoded representation of the semantic segmentation map. In this representation the semantic segmentation maps has a size of $(W \times H \times C)$, in other words we have one channel for each of the classes, where the values are binary i.e $[0, 1]$. For example, given an semantic segmentation map from the ModaNet data set with the value 1 at pixel coordinate (100,100,6) indicates that there is a dress at (W,H)=(100,100) in the corresponding image. An example of an semantic segmentation map where the 13 channels are out-rolled are shown in fig. 4.3

4.2 Models

For the experimental setup of this thesis three different GAN models were used and in this section the architecture of the used models will be described. We start with describing the model used for generating fashion images and then continue with describing the two different models used for generating fashion images and semantic segmentation maps.

4.2.1 Style GAN - for image generation

The model used for generating images completely replicates the work done by [19] and will be refereed to as Style GAN, we use same hyper-parameter setting as in the original Style GAN setup. Further, the Style GAN model is an extension of the Pro GAN model introduced in [18]. One of the Pro GAN model strengths is the progressively growing of the generator and the discriminator network. The

progressive growing precedes by starting with a small generator and discriminator which can generate 8×8 resolution images. Then the networks are trained using Algorithm 1, next we grow both networks by adding a new block of layers such that the new output/input resolution is doubled. The procedure is then repeated until the networks has capacity to generate and discriminate images with the target resolution. The progressively growing of the networks encourage training stability and speeds up training. The progressive growing property is inherited by the Style GAN model.

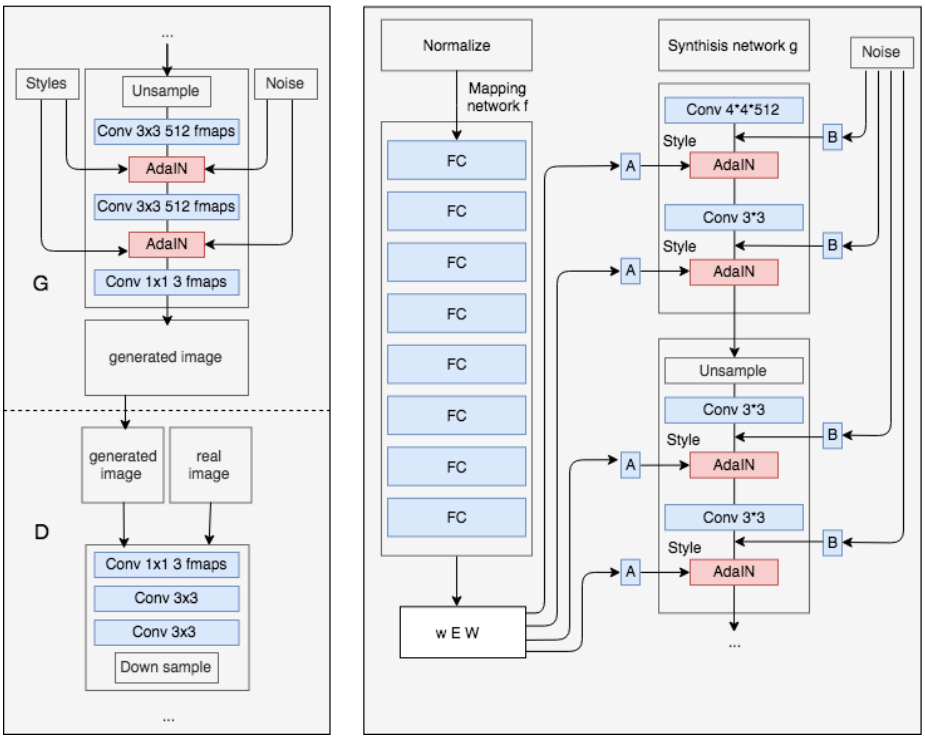

Figure 4.4: Figure illustrating the architecture of style GAN. The left figures shows the layers in each building block, where the upper part shows the generator and the lower the discriminator. The right figure shows the entire architecture of the Style GAN generator network.

4.2.1.1 Style GAN - generator

The architecture of Style GAN generator G is shown in the right panel of fig. 4.4. As seen in the figure the Style GAN generator consists of two parts. The first part is called the mapping network and is a multi layer perceptron (MLP) with 8 layers. Where all 8 layers uses 512 neurons and leaky ReLU [26] activation with $\alpha = 0.2$. The second part is called the synthesis network and consists of n block layers. Each of the block layers consists of four different types of layers, layer one is a 2 dimensional up sampling layer, layer two and tree are convolutional layers, with kernel size 3 and 512 feature maps. The last layer is also a convolutional layer with kernel size 3 and 3 feature maps. All intermediate layers in the synthesis network uses leaky ReLU activation with $\alpha = 0.2$. The architecture of each block layer is shown in the

left upper panel of fig. 4.4. The number of block layers n in the synthesis network is dependent of the target resolution. We need to add one new block layer for each time we double the resolution starting from resolution 8×8, i.e we need n blocks for target resolution $2^{2+n} \times 2^{2+n}$.

The information flow from the input $\mathbf{z}$ to the output image in the Style GAN generator G, differs from what's normally used in the GAN literature. In Style GAN the latent code do not directly control the synthesis network. Instead the latent code $\mathbf{z}$ is mapped through the mapping network into a feature representation $\mathbf{w}$. Then by a learned affine transformation, $\mathbf{w}$ is mapped into the so called styles $\mathbf{y} = (\mathbf{y_s}, \mathbf{y_b})$ (the name is borrowed from style transfer literature [19]). The style then controls by the adaptive instance normalization operation (AdaIN) [15] the output of each convolutional layer in the synthesis network. The AdaIN operation is defined by eq. (4.1) where $\mathbf{x_i}$ is the output feature map:

$$AdaIN(\mathbf{x_i}, \mathbf{y}) = \mathbf{y_{s,i}} \frac{\mathbf{x_i} - \mu(\mathbf{x_i})}{\sigma(\mathbf{x_i})} + \mathbf{y_{b,i}}, \tag{4.1}$$

hence the AdaIN operation aligns mean and variance of each feature map to match those of the styles $\mathbf{y}$. In addition to controlling the synthesis network by the styles, noise is added by a learned scaling factor to each feature map in the convolutional layers. The adding of noise has been shown to increase the generators ability to generate images with stochastic variation, such as placement of hairs, freckles etc [19].

4.2.1.2 Style GAN - discriminator

The Style GAN discriminator D is the mirrored version of the synthesis network, without the adding of noise and the AdaIN operation. The discriminator takes an image as input, then the images is feed through the same type of blocks as in the generator, but the discriminator uses down sampling instead of up sampling as can be seen in the left lower panel of fig. 4.4.

4.2.2 Style C GAN - for image and semantic segmentation map generation

For the aim of generating both images and semantic segmentation maps the Style GAN model was extended to a new architecture which will be referred to as Style C GAN. All new elements of Style C GAN is stated below, model specifics that are not stated remains the same as for the Style GAN architecture.

4.2.2.1 Adding feature maps

To have the ability to generate both images and semantic segmentation maps we add an extra set of C feature maps to the last convolutional layer (see fig. 4.4) in the Style GAN discriminator D and the generator G. Hence the number of feature maps in the last convolutional layer of each block in the Style C GAN discriminator D_c and generator G_c will now equal $3+C$, i.e the input and output is now an object

of dimension $(W \times H \times 3 + C)$ that allocates 3 channels for the image and C channels for the semantic segmentation map. Therefore $C = 13$ when generating all classes in ModaNet.

4.2.2.2 Progressive growing in the channel dimension

When training Style C GAN we start from a pre-trained Style GAN network pair D and G. We then apply progressive growing in the channel dimension to translate D, G to the Style C GAN generator discriminator pair D_c, G_c. The progressive growing in the channel dimension is accomplished using two steps. In step one we add one new feature map such that the model can generate images and corresponding one class semantic segmentation maps. Where the one class semantic segmentation map is the resulting map from labeling all fashion classes as 1 and background as 0. When adding the new feature map we do not use random initialization of weights, instead we initialize the weights by copying the weights allocated for the red channel in the RGB image. The reason, pre-initialized weights trained on a similar task can speed up convergence, this is what we usually call transfer learning [31]. In step two we grow the network such that the model has capacity to generate $C = 13$ classes i.e all classes in ModaNet. We hence add 12 new feature maps. We initialize the 12 new feature maps by using the weights from the feature map added in the previous step, i.e the feature map trained for generating the one class semantic segmentation map. The training precedes by using Algorithm 1.

4.2.3 Style C GAN 2D - for image and semantic segmentation map generation

For the last model we combine the Style GAN discriminator D with the Style C GAN generator G_c and discriminator D_c. Hence in the new setup the Style C GAN generator G_c is trained in supervision of two discriminators D and D_c. The purpose of the first discriminator D is to learn to discriminate between real images and generated images, whereas the second discriminator D_c aims to learn if the combination of generated images and segmentation maps is real or generated, hence the second discriminator needs to learn to distinguish if pairs of images and semantic segmentation maps are coherent or not coherent.

For the new model setup the architectures for each of the networks D, G_c and D_c remains the same as explained in section 4.2.1 and section 4.2.2. However for training using two discriminators we add a discriminator and hence slightly adjust the training procedure which is explained below.

4.2.3.1 Adding discriminator

When training G_c using two discriminators we first use the same training procedure as for Style C GAN explained in section 4.2.2. We hence start with training the generator for generating images in target resolution. Then we apply progressive growing in the channel dimension for both the generator G and the discriminator D. Hence we translate G to G_c, and D to D_c, i.e we translate to a generator discriminator

pair that has capacity to generate both images and semantic segmentation maps. However, when training using two discriminators we keep a copy of the image discriminator D, and we train D to discriminate between images throughout the whole training procedure.

4.2.3.2 Changing training procedure

The new training procedure after adding the discriminator is as follows. First we use G_c to generate a batch of objects of dimension $(W \times H \times 3 + C)$. We then feed the generated batch together with the real batch of same dimension to the discriminator D_c, and preform a gradient step using Algorithm 1. Second we use G_c to again generate a batch of objects of dimension $(W \times H \times 3 + C)$, however this time we only keep the object that allocates the three first channels i.e the image. Then we feed the generated batch of images together with a real batch of images to the discriminator D, and preform a gradient step using Algorithm 1.

4.3 Evaluation Metric

Evaluating sample quality from generative models is challenging [2], in particular for GAN models, since the loss function of GANs is not interpretable as a measurement between different models performance [36]. The most straight forward approach would be to use human annotators for evaluating the image quality [6]. However, this approach could potentially be both time consuming, costly, and in addition different annotators may have different objectives implying an unreliable measurement. Luckily, the GAN literature includes a variety of different qualitative measurements provided to evaluate the generated samples, where the different measurements might be beneficial for different setups [45]. In this thesis we will use two of the most popular measurements, namely Inception score (IS) [36] and, Fréchet Inception distance (FID) [14].

However, it is important to note that both the IS and the FID has limitations in ability to measure both quality and variation in samples. In order to compensate for some of those limitations we do an qualitative similarity evaluation between generated and real data, and in addition we investigate the effect of traversing the latent space. At last, we evaluate the generated semantic segmentation maps and corresponding generated images by evaluating the samples performance on the semantic segmentation task section 2.4.

4.3.1 Inception score

The Inception score is by far the most used metrics for evaluating performance of a generative image model [45]. The Inception score relies on the Inception v3 network [41], a classification network trained on ImageNet [5]. Where ImageNet is a data set consisting of 1.2 million RGB images from 1000 classes. Therefore given an input $\mathbf{x}$ the output of the Inception v3 network is the class conditional distribution over

the 1000 ImageNet classes. The definition of the Inception score given a generative model G is given below:

$$\text{IS(G)} = \exp(\ \mathbb{E}_{\mathbf{x}\sim p_g}\, KL(p(y|\mathbf{x})\ ||\ p(y))\ ). \tag{4.2}$$

Where $\mathbf{x}$ is a sample drawn from the generative model G, $p(y|\mathbf{x})$ is the class conditional distribution given by the Inception v3 network for a generated sample $\mathbf{x}$, $KL(p(y|\mathbf{x})\ ||\ p(y))\ )$ is the KL-divergence between the class conditional distribution and the marginal class distribution $p(y)$ which is approximated by $\hat{p}(y)$ given by,

$$\hat{p}(y) = \frac{1}{N}\sum_{i=1}^{N} p(y|\mathbf{x_i}) \tag{4.3}$$

where N is the number of samples. In [36] it was proposed to use a great amount of images to give a reliable estimation of the Inception score. They suggest to use a set of at least 50k images and to split the set into 10 batches, then calculate the mean and the standard deviation of the Inception score over the batches.

The Inception score is high if the two following conditions are full-filled: Condition one, the class conditional distribution $p(y|\mathbf{x})$ for individual samples has low entropy, i.e the model finds it likely that the image belongs to a certain class implying sharpness in individual images. Condition two, the marginal class distribution $p(y)$ has high entropy, i.e the marginal class distribution is close to uniform implying large variety in generated samples.

In conclusion a high Inception score indicates sharp images and large variation between samples i.e no mode collapse. In addition high Inception score has been shown to align well with human judgment [36]. Hence, the higher Inception score the better. However it is important to note that the Inception score has limitations [2]. First of all the value of the Inception score does not penalize a generative model that memorize and generate images that can be found in the data. Instead the Inception score encourage a model that has generated the exact same images. Second, the Inception v3 network is trained on ImageNet which includes 1000 classes, hence using the Inception v3 network to calculate Inception score for a generative model trained on a data set with classes that do not align with the classes in ImageNet might yield misleading results [33]. A better solution might hence be to use the Inception v3 network trained on the same data set as used for training the generative model. However, to align with other works we use the Inception v3 network trained on ImageNet to calculate the Inception score. It is therefore important to note that the Inception score for models trained on ModaNet can only be compared to the Inception score for other models trained on ModaNet.

4.3.2 Fréchet Inception distance

The Frèchet Inception distance (FID) was proposed in [14] with the aim of improving the Inception score. While the Inception score provides a score of the quality of generated images the FID tries to measure the distance between generated and real data samples. Hence the lower FID the better. Similar to the Inception score the

FID has been shown to correlate with human judgment [14].

When calculating the FID one calculates the Fréchet distance [8] also referred to as the Wasserstein-2 distance [42] between the generated and real data distribution. The Fréchet distance measures the distance between two Gaussian distributions, the FID therefore relies on the assumption that the 2084 dimensional output from the pool3 layer in the Inception v3 network follows an multidimensional Gaussian distribution. The mathematical definition of the FID is given below,

$$FID = ||\mu_r - \mu_g||^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}), \tag{4.4}$$

where $X_g \sim N(\mu_g, \Sigma_g)$, $X_r \sim N(\mu_r, \Sigma_r)$ is the 2084 dimensional output from the pool3 layer in the Inception v3 network for the generated and real samples respectively.

In comparison to the Inception score, the FID can in larger extent indicate the performance of generators generating using different training data sets. Since the FID measures the distance between generated and real data. However, similar to the Inception score the FID benefits a generator generating the same images that are already included in the real data set. Therefore in next section we provide a qualitative evaluation metric that tries to evaluate if the generative model generated images that are included in the real data set.

4.3.3 Similarity evaluation in feature space

As mentioned in section 4.3.1 and section 4.3.2 both the Inception score and the FID fails in measuring if the generative model is generating images that are already contained in the real data set. We therefore propose to preform a qualitative similarity evaluation between generated and real data samples. We do this by presenting the 10 nearest neighbours from the real data set given a generated image. Where the nearest neighbours are given as the nearest images in L_2 norm. However, it is well known that the L_2 norm in pixel space does not reliably measure the distance between images [43, 7]. Small changes in pixel values can give large impact on the L_2 norm. Therefore we calculate the L_2 norm nearest neighbours in feature space, using the pool3 layer in the Inception v3 network. This technique for evaluating similarity between images is also used in [4], however, they use the fc7 layer in VGG-16 [39] and the avgpool layer in ResNet-50 [13], we use the Inception v3 network for simplicity.

4.3.4 Traverse the latent space

To further investigate the variation in generated samples we provide visualizations of the effect of traversing the latent space. The evaluation is done by choosing two latent codes z_1^*, and z_2^*, where both codes yields a unique generated image $x_1^* = G(z_1^*)$ and $x_2^* = G(z_2^*)$, where G denotes the generator. We then calculate the vector $w = z_1^* - z_2^*$ between the first and second latent code. Using w we can move from z_1^* to z_2^* and collect latent codes z_j on the line connecting z_1^* with z_2^*. I.e collect

latent codes $z_j : z_j = z_1 - \frac{j}{N}w$, where N is the number of codes we want to collect. We then visualize the corresponding generated images $x_j = G(z_j)$ and observe the variation.

4.4 Putting it all together from raw data to final model

We conclude this chapter with putting it all together which leads us to describing the entire training procedure, from raw data to evaluating the final model. We start with describing how the data was divided into a training, a validation and a test set. Secondly how the data was pre-processed. Thirdly we describe the specific parameters related to the training procedure and at last we describe how the models where evaluated.

4.4.1 Training, validation and test split

As mentioned in section 4.1 consists ModaNet data set of 55.176 fully annotated street images, however only 52.346 was available for downloading. From the set of 52.346 images we randomly chose images to create the training set the validation set and the test set, the amount of images in each set was chosen to be 45.346, 2000, 5000 correspondingly.

4.4.2 Data pre-processing

We apply two pre-processing steps for the training images. The same two steps are applied for images both when training Style GAN, Style C GAN, and Style C GAN 2D. Step one was to resize the images to have an equal height H and width W, such that the images fit the model architecture. The resizing transformation was defined such that it perseveres the aspect ration of the humans. Step two was normalizing the training images to range in between $[-1, 1]$. We normalize using a hard range between -1 and 1, as in the original Style GAN setup.

The semantic segmentation maps used for training the Style C GAN was pre-processed using three steps. First step was to resize the maps, using exact same procedure as for the images. In the second step we shifted the binary values in the maps such that they are symmetric around zero i.e such that $0 \rightarrow -0.5$ and $1 \rightarrow 0.5$. The third step was to relax the problem, therefore we add Gaussian noise with mean 0 and standard deviation 0.01 to each map. As mentioned in section 3.1 adding noise to images compensate the sharpness of generated images. However, this is not a problem for the semantic segmentation maps generation since when finish training we set all value less than 0 to 0 and all values grater than 0 to 1. We did not compare the results of adding noise with not adding noise, the reason, there was nothing indicating that the adding of noise decreased performance.

4.4.3 Training schedule

All models where trained using Algorithm 1 to convergence at the target resolution of 512×512 pixels. The target resolution was chosen as consequence of the size of the training images in ModaNet. Further as loss function we use the "log-trick" GAN objective, i.e we train using the following objective function,

$$\max_G \max_D \mathbb{E}_{\mathbf{x} \sim p_{data}}[log(D(x))] + \mathbb{E}_{\mathbf{z} \sim p_z}[log(D(G(z)))]. \tag{4.5}$$

We use same hyper-parameter settings as proposed in the Style GAN paper, the ones related to the training procedure is stated as follows. We optimize the objective function using the Adam optimizer [20], with exponential decay $\beta_1 = 0.0$, $\beta_2 = 0.99$ and we use resolution dependent learning rate α ranging between $= 0.001$ to $\alpha = 0.002$ for resolution (8×8) to target resolution (512×512). We also use a resolution dependent mini-batch size ranging from 128 samples in each mini-batch to 16 samples in each mini-batch. The reason why we do not perform any hyper-parameter search is due to time and computational limitations.

All models are trained using 4 GPUs each, and the entire training procedure takes approximately 1-2 weeks for each model.

4.4.4 Evaluation

The models where evaluated using the metrics presented in section 4.3. We do all model selections based on the best recorded FID (see section 4.3.2), using the validation data set. Whereas all reported FID are evaluated using the test data set.

5
Results

In this chapter the results from the experimental setup is presented. The chapter is divided into four sections, where the results of each section aims to answer each of the stated research questions (see section 1.1). All results are extracted using the generative models with best validation FID, i.e for each model training is stopped when the models has reached its optimum validation FID at resolution 512×512.

5.1 Semantic segmentation maps generation

(R_1) Is it possible to use GANs to generate realistic pairs of images and semantic segmentation maps?

By the aim of answering research question R_1 we present results from Style C GAN and Style C GAN 2D. The results that will be presented are, visualizations of images and semantics segmentation maps, and a comparison between the proportion of instances per class in the generated set of semantic segmentation maps and the proportion of instances per class in ModaNet.

In fig. 5.1 and fig. 5.2 the outputs from four latent codes using Style C GAN and Style C GAN 2D are shown. For each output the generated image, the generated semantic segmentation map, and the generated semantic segmentation map on top of the generated image is shown.

Figure 5.1: Four outputs generated by Style C GAN at resolution 512×512 trained on ModaNet. For each output the generated image, the generated semantic segmentation map and the generated semantic segmentation map on top of the image is shown.

Figure 5.2: Four outputs generated by Style C GAN 2D at resolution 512×512 trained on ModaNet. For each output the generated image, the generated semantic segmentation map and the generated semantic segmentation map on top of the image is shown.

The results in fig. 5.1 and fig. 5.2 indicates that both Style C GAN and Style C GAN 2D can generate semantic segmentation maps that are coherent with the semantics of the generated image. We further note that there is no significant difference in how well the generated semantic segmentation maps fits the generated images while comparing Style C GAN fig. 5.1 and Style C GAN 2D fig. 5.2.

In fig. 5.3 the proportion of instances per class is presented for ModaNet data set, for 50K of generated images from Style C GAN, and for 50K of generated images from Style C GAN 2D. The classes are ordered in decreasing order with regard to instance proportions in ModaNet data set.

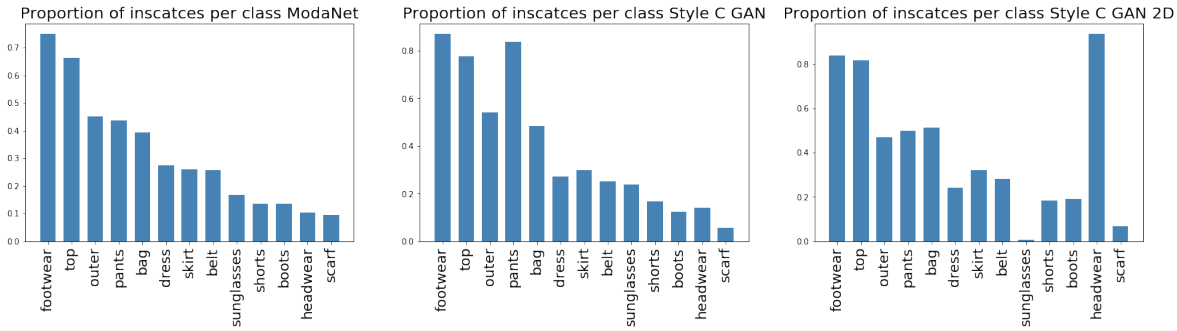

Figure 5.3: The left figure shows proportions of instances per class in ModaNet data set, the middle figure shows proportions of instances per class from a data set of 50K images generated by Style C GAN, the right figure shows proportions of instances per class from a data set of 50K images generated by Style C GAN 2D.

From fig. 5.3 we note that Style C GAN seems to have generated images where class instances occur with similar frequencies as for ModaNet data set, the pants class seems however to be overrepresented. Style C GAN 2D on the other hand seems to have generated images where the instances per class is quite similar to ModaNet data set, except from the headwear and sunglass classes. From visualizing generated

images it was further noted that Style C GAN 2D sometimes interpret sunglasses as headwear, which can explain why the sunglass class is underrepresented. The lower left corner of fig. 5.2 illustrates one example of where sunglasses are interrupted as headwear. It was further noted that to learn classes where number of instances are few, such as sunglasses, a larger number of training iteration was needed compared to well represented classes.

5.2 Image comparison

(R_2) Comparing generated images with images generated together with corresponding semantic segmentation map, how will the quality of generated images be affected?

In this section we address research question R_2 and thereby compare results regarding image quality for images generated with and without semantic segmentation maps. In table 5.1 qualitative results of the image quality using IS (see section 4.3.1) and FID (see section 4.3.2) for all used models, i.e Style GAN , Style C GAN and Style C GAN 2D are presented. The reported IS have been calculated using 50K of generated images and the reported FID have been calculated using the test set and 50K of generated images.

Table 5.1: IS and FID for comparing image quality for Style GAN, Style C GAN, and Style C GAN 2D. IS is calculated as the mean IS of five batches using 10K generated images in each batch and FID using the test set and 50K generated images.

Model	IS	FID
Style GAN	5.06 ±0.11	12.34
Style C GAN	4.79 ±0.12	25.10
Style C GAN 2D	4.69 ±0.12	20.97

From table 5.1 we note that Style GAN preforms best both regarding IS and FID, implying that the image quality of image samples generated using Style GAN generator have higher quality compared to the other two models. Hence generating both images and semantic segmentation maps seems to affect image quality negatively. We also note that Style C GAN and Style C GAN 2D have similar IS but that Style C GAN 2D performs better regarding FID.

In fig. 5.4, fig. 5.5, and fig. 5.6 visualizations of generated samples from Style GAN, Style C GAN, and Style C GAN 2D are presented. All images are generated in resolution 512×512, however, to get a better understanding of the variation between images, we visualize generated images in different sizes.

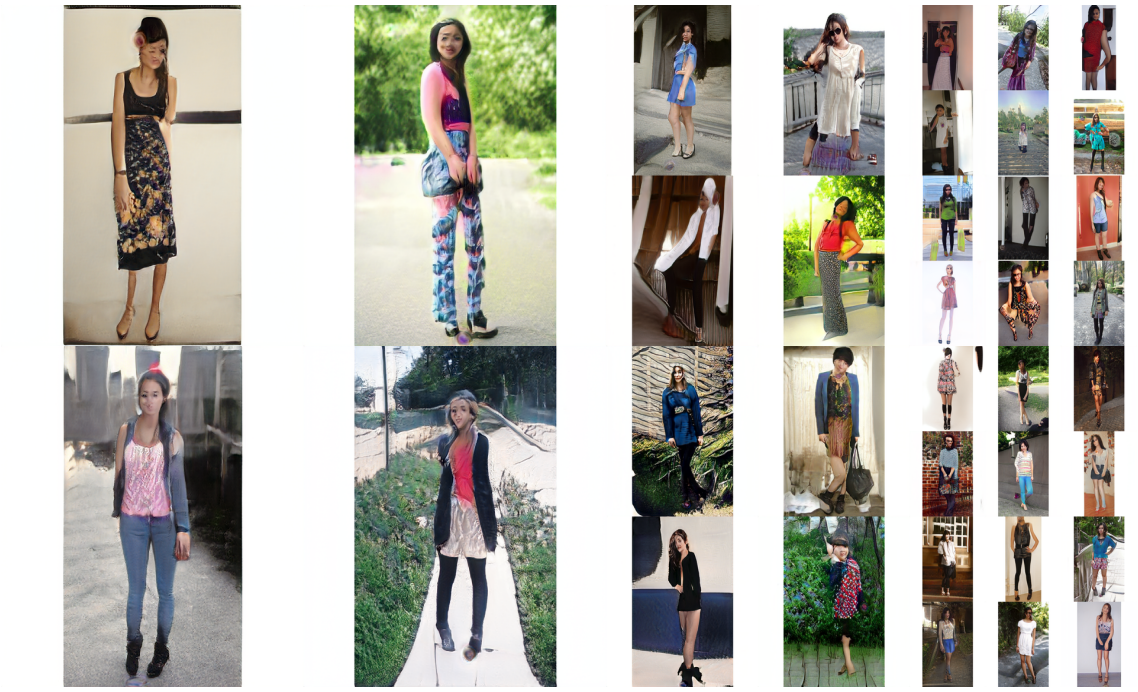

Figure 5.4: Samples produced by Style GAN generator at resolution 512×512 trained on ModaNet. The FID calculated using test set was 12.34 and the IS was 5.06.

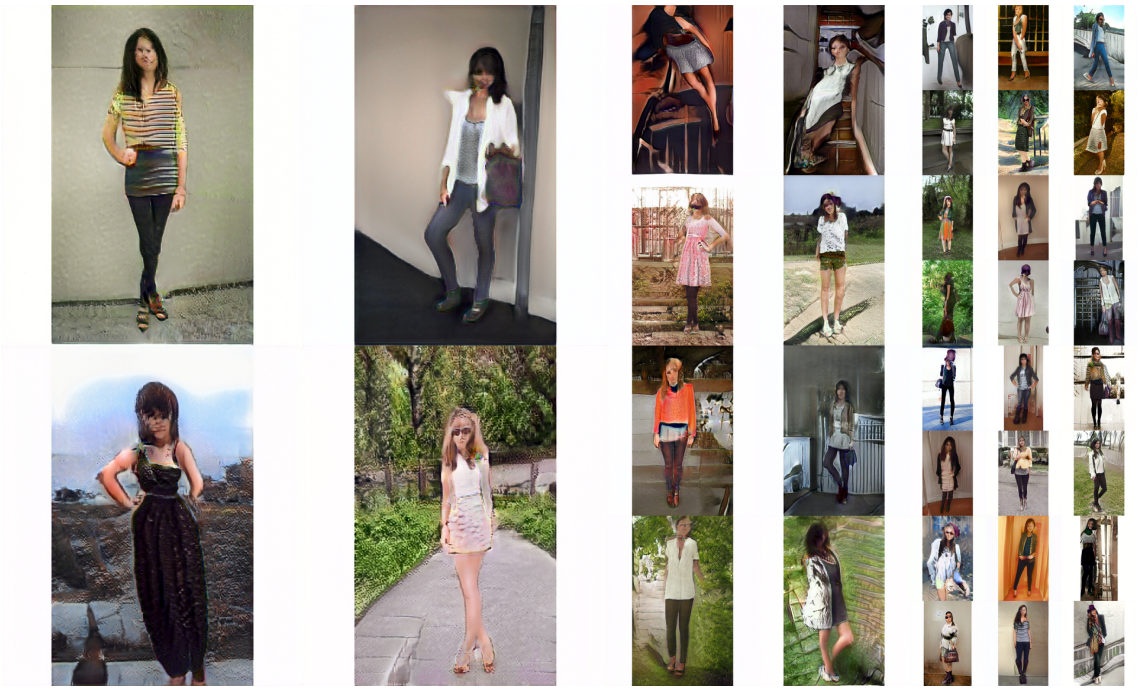

Figure 5.5: Samples produced by Style C GAN generator at resolution 512×512 trained on ModaNet. The FID calculated using test set was 25.10 and the IS was 4.79.

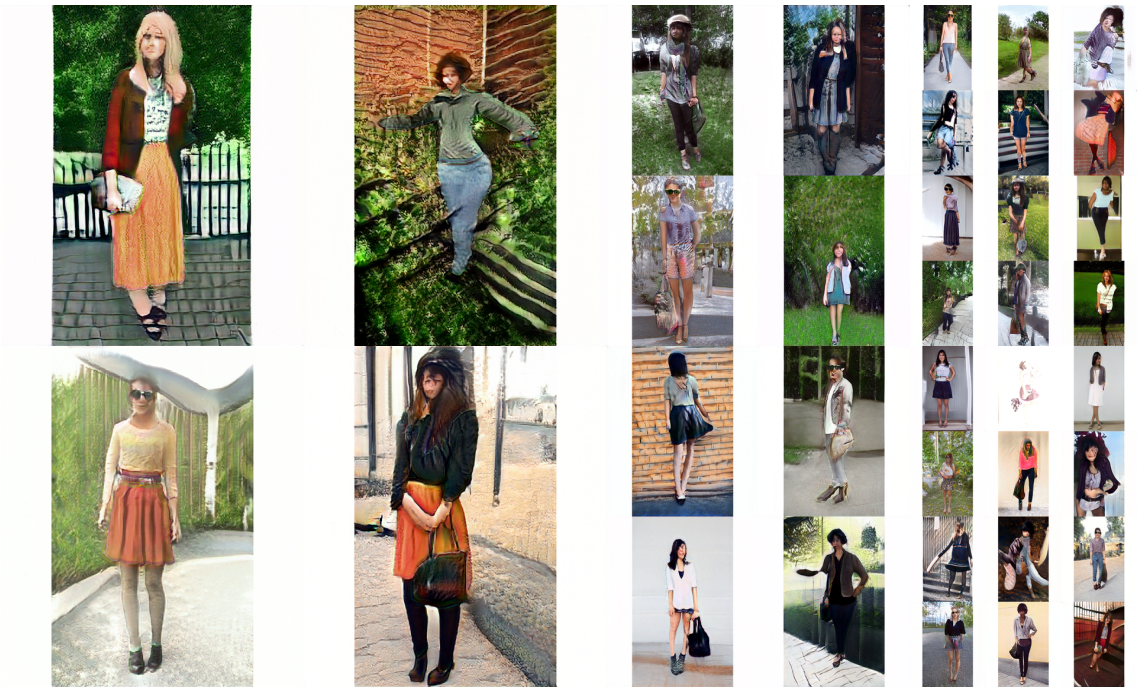

Figure 5.6: Samples produced by Style C GAN 2D generator at resolution 512×512 trained on ModaNet. The FID calculated using test set was 20.97 and the IS was 4.69.

Comparing fig. 5.4, fig. 5.5, and fig. 5.6 we note that for a human observer it is hard to distinguish differences in image quality between the models, although IS and FID differs. We further note that variation in images seems to be quite high.

5.3 Similarity evaluation

(R_3) What will be that variation of generated samples, are we able to generate images that do not already exists in the training set?

In this section we present results, which aims to answer the last research question, therefore we visualize the results from using the similarity evaluation in feature space discussed in section 4.3.3 and from traversing the latent space discussed in section 4.3.4.

We start with presenting results from performing similarity evaluation in feature space, in fig. 5.7 we presents results from using Style C GAN and in fig. 5.8 from style C GAN 2D. In both figure we visualize one generated image together with the 10 nearest neighbors in the feature space of the pool3 Inception V3 network.

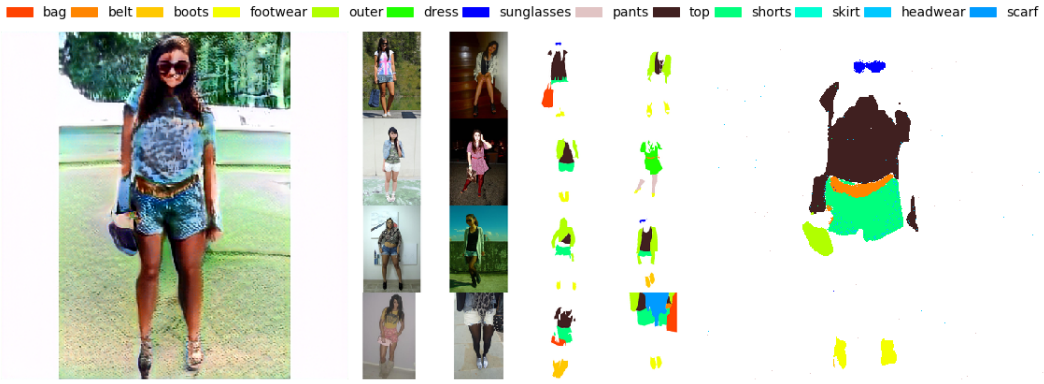

Figure 5.7: Visualization of similarity evaluation in feature space for Style C GAN. From left to right: generated image, 10 nearest neighbors in the feature space of the pool3 Inception V3 network, semantic segmentation maps of the 10 nearest neighbors, the generated semantic segmentation map.

.

Figure 5.8: Visualization of similarity evaluation in feature space for Style C GAN 2D. From left to right: generated image, 10 nearest neighbors in the feature space of the pool3 Inception V3 network, semantic segmentation maps of the 10 nearest neighbors, the generated semantic segmentation map.

We note form fig. 5.7 and fig. 5.8 that the images most similar in feature space is not equal to the generated image. However the nearest neighbors are similar both to the generated image and to each other. This implies that in feature space we are able to find similar images. Therefor the results indicate that it is very unlikely the two presented examples are not represented in the data set.

To further investigate the output variation of the generator we will now visualize the output response from traversing the latent space. In fig. 5.9 we present the results from interpolating between two latent codes, and generating images using Style C GAN. Similar results can be obtained using Style C GAN 2D.

Figure 5.9: Traversing the latent space. Interpolation between two latent codes, resulting samples generated by Style C GAN. The upper panel shows the generated images and the lower panel shows the generated semantic segmentation maps.

From fig. 5.9 we note that both poses, humans, backgrounds, and clothes changes as we interpolate between latent codes. We also note that the same human continuously appears in different poses and different clothes. Similar results can be obtained independently of the chosen latent codes. In the real data set we can not continuously interpolate between different images, the result therefor indicates that the generator is able to generate images that do not exist in the real data set.

6

Conclusion

The aim of this thesis was to investigate if it is possible to generate images and corresponding semantic segmentation maps. By extending on the Style GAN model we proposed two new models Style C GAN and Style C GAN 2D. It was shown that both models could successfully generate images together with coherent semantic segmentation maps. However, it was noted that samples generated by the Style GAN model gained better results with respect to the image quality compared to the proposed models for image and semantic segmentation maps generation. The reason, when generating both images and semantic segmentation maps the gradient feed back, from the discriminator will not only focus on the image quality as in the case of only generating images. The problem was partly resolved by introducing the Style C GAN 2D model where a discriminator was added, which did increase image quality compared to Style C GAN. However, Style C GAN 2D still lack the final performance obtained for Style GAN.

Further, the results implied that we could generate image with high variation and images that do not already exist in the data set. From traversing the latent space by interpolating between latent codes we found that the response in the generator was that clothes, poses, backgrounds, and persons continuously changed, this can not be observed in the real data set.

We did not compare our extended Style GAN models with extending on any other GAN model. However, in the beginning of the project different GAN models where tested by the aim of modelling the image distribution of ModaNet. The tested models could not train stable and both mode collapse and training instability was observed. We found that the distribution of ModaNet was hard to model due to large variation between images in the data set, both regarding clothes, backgrounds, and poses. The theory suggested to change the objective and use the softer metric also called Wasserstein distance to encourage training stability. However, we found that using Style GAN we could observe stable training while still using the original GAN objective with the log-trick. We therefor conclude that the progressive growing property that the Style GAN model inherited encourage training stability in a larger extension than chaining the objective.

6.1 Future work

In future work it would be interesting to use the generated data to train a semantic segmentation algorithm, it is however not obvious that the generated images would actually increase the accuracy on a semantic segmentation algorithm. If we want to increase the accuracy on test data we would like to use the generator to sample where the training data is not well represented, however, it is likely that also the generator finds it easier to generate from parts of the distribution where many real samples are contained. It would therefore be interesting to use the Style C GAN 2D model, which can be trained separately on both images and annotated images. We can therefore add training data of only images to see if we can capture an even larger distribution of annotated images, than the one with images and hand annotated ground truth's.

Another interesting direction, which also could increase test accuracy for the semantic segmentation algorithm, is to use the generator to compensate for the class imbalance observed in ModaNet. Instead of choosing arbitrary generated images when training the semantic segmentation algorithm we could choose images containing class instances which are not well represented in the real data set, for example we can choose more generated images containing scarf which is the class with the least amount of instances in ModaNet.

I addition it was noted that classes with few instances per class was learned later during training, and that Style C GAN 2D had not been able to capture the sunglasses class at convergence regarding the FID. For model selection in future work it would therefor be interesting to use an metric where both image quality and semantic segmentation maps are measured.

Bibliography

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[2] Shane Barratt and Rishi Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

[3] Yoshua Bengio, Eric Laufer, Guillaume Alain, and Jason Yosinski. Deep generative stochastic networks trainable by backprop. In *International Conference on Machine Learning*, pages 226–234, 2014.

[4] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[6] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.

[7] David A Forsyth and Jean Ponce. A modern approach. *Computer vision: a modern approach*, 17:21–48, 2003.

[8] Maurice Fréchet. Sur la distance de deux lois de probabilité. *COMPTES RENDUS HEBDOMADAIRES DES SEANCES DE L ACADEMIE DES SCIENCES*, 244(6):689–692, 1957.

[9] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[11] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a

local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[15] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[17] Anil K Jain, Jianchang Mao, and KM Mohiuddin. Artificial neural networks: A tutorial. *Computer*, (3):31–44, 1996.

[18] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[19] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2018.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[23] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[24] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[26] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[27] Léon Bottou Martin Arjovsky. Towards principled methods for training generative adversarial networks. *arXiv:1701.04862*, 2017.

[28] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[29] Marvin Minsky. S. papert. perceptrons: An introduction to computational geometry, 1969.

[30] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[31] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[32] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[33] Mihaela Rosca, Balaji Lakshminarayanan, David Warde-Farley, and Shakir Mohamed. Variational approaches for auto-encoding generative adversarial networks. *arXiv preprint arXiv:1706.04987*, 2017.

[34] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[35] Ruslan Salakhutdinov and Hugo Larochelle. Efficient learning of deep boltzmann machines. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 693–700, 2010.

[36] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[37] Hannes Schulz, Andreas Müller, and Sven Behnke. Investigating convergence of restricted boltzmann machine learning. In *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[38] Jonathon Shlens. Notes on kullback-leibler divergence and likelihood. *arXiv preprint arXiv:1404.2000*, 2014.

[39] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[40] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[41] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[42] Leonid Nisonovich Vaserstein. Markov processes over denumerable products of spaces, describing large systems of automata. *Problemy Peredachi Informatsii*, 5(3):64–72, 1969.

[43] Zhou Wang and Alan C Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, 26(1):98–117, 2009.

[44] Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov, and Roger Grosse. On the quantitative analysis of decoder-based generative models. *arXiv preprint arXiv:1611.04273*, 2016.

[45] Qiantong Xu, Gao Huang, Yang Yuan, Chuan Guo, Yu Sun, Felix Wu, and Kilian Weinberger. An empirical study on evaluation metrics of generative adversarial networks. *arXiv preprint arXiv:1806.07755*, 2018.

[46] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[47] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

[48] Shuai Zheng, Fan Yang, M Hadi Kiapour, and Robinson Piramuthu. Modanet: A large-scale street fashion dataset with polygon annotations. In *2018 ACM Multimedia Conference on Multimedia Conference*, pages 1670–1678. ACM, 2018.