

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

AI-based Fashion Designer

Fábio Filipe Vilela Oliveira



Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Rui Maranhão Abreu, PhD

Co-Supervisor: Abubakar Zakari, PhD

July 14, 2021

AI-based Fashion Designer

Fábio Filipe Vilela Oliveira

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Luís Filipe Teixeira, PhD

External Examiner: Jonathan Rubin, PhD

Supervisor: Rui Maranhão Abreu, PhD

Co-Supervisor: Abubakar Zakari, PhD

July 14, 2021

Abstract

Fashion is one of the most essential arts of civilization. It plays a key role in our daily lives from the moment we get dressed in the morning, providing confidence and self-esteem. It is also a way used by individuals to express their feelings or make a statement. In the 18th century, the industrial revolution paved the way for the mass production of apparel. With that, globalization has presented us with great diversity at low costs. Nevertheless, the design and shipment of new clothes to consumer markets can take a significant amount of time. This poses a problem for fashion companies because consumer trends change rapidly. What customers wanted to purchase in the past is not necessarily the same as today, leading at best to unhappy consumers, and at worst to unsold stocks and financial losses.

This dissertation introduces an Artificial Intelligence (AI) pipeline for the automatic generation of new patterns and designs, allowing fashion designers to start working from prototype garments generated by the AI pipeline instead of the ground up. Thus, the proposed pipeline introduces several improvements to the whole fashion process. For once, it reduces the time to market of new products. This, in turn, means that clothes catalogs' of fashion brands becomes more aligned with ongoing consumer trends. Moreover, the clothes' prices can be lower because the process is less expensive due to the introduction of automation. Additionally, the designs' quality will be improved because designers can target their efforts to already existing generated clothing and spend more time fine-tuning details to meet consumer demands.

The proposed pipeline performs the tasks of i) automatic generation of clothing garments and ii) dressing real-world fashion models in the generated garments. The pipeline is based on Artificial Neural Networks and comprises two components: an Unconditional Garment Design component and a Garment Try-On component. First, the Unconditional Garment Design component generates images of novel garments. Then, these garments are served as input to the Garment Try-On component, which produces images of fashion models dressing the garments. Thus, the proposed pipeline aids the work of fashion designers by allowing them to base new products on AI-generated content. As a result, the design efforts for new fashion products are reduced, and fashion designers' productivity is improved.

The proposed AI pipeline was evaluated in terms of performance metrics and a user survey. The collected performance metrics (Fréchet Inception Distance and Structural Similarity Index Measure) indicate that images produced by the pipeline are of high quality. As for the user survey, a total of 38 volunteers were asked questions regarding i) the realness and ii) the marketability of generated garments. It was observed that 52% of the volunteers failed to distinguish between generated and real garments on more than half of the survey questions. Furthermore, the marketability of real and generated garments as reported by the volunteers is suggested to be the same. Therefore, the proposed pipeline generates realistic clothing that fulfills the fashion needs of consumers.

Keywords: Artificial Intelligence, Deep Learning, Fashion, Generative Design, Fashion Design

Resumo

A moda é uma das artes essenciais da humanidade. Desempenha um papel fundamental na nossa vida diária a partir do momento em que nos vestimos de manhã, proporcionando confiança e auto-estima. É também uma forma utilizada por indivíduos para expressar os seus sentimentos e ideias. No século XVIII, a revolução industrial abriu o caminho para a produção em massa de vestuário. Com isso, a globalização apresentou-nos uma grande diversidade a baixos custos. No entanto, a concepção e o envio de novas peças de roupa para as lojas pode levar um tempo significativo. Isto coloca um problema às empresas de moda, porque as tendências dos consumidores mudam rapidamente. O que os clientes queriam comprar no passado não é necessariamente o mesmo que hoje, levando na melhor das hipóteses a consumidores infelizes, e na pior das hipóteses a stocks não vendidos e a perdas financeiras.

Esta dissertação introduz uma *pipeline* de Inteligência Artificial (IA) para a geração automática de novos padrões e designs de roupa, permitindo aos estilistas de moda criar novas peças de vestuário a partir de protótipos de roupa gerados pela pipeline de IA. Assim, a *pipeline* proposta introduz várias melhorias para o mercado da moda. Por uma vez, reduz o tempo necessário para colocar novos produtos no mercado. Isto, por sua vez, torna os catálogos das marcas de moda mais alinhados com as tendências dos consumidores. Além disso, os preços das peças de roupa podem ser mais baixos porque o processo de design é menos dispendioso devido à introdução da automatização. Além disso, a qualidade dos designs de roupa são melhorados porque os estilistas podem orientar os seus esforços para vestuário gerado por IA e gastar mais tempo a afinar detalhes para satisfazer as exigências dos consumidores.

A *pipeline* proposta gera automaticamente novas peças de roupa e veste modelos de moda com as peças de roupa geradas. A *pipeline* é baseada em Redes Neurais Artificiais e é composta por dois componentes: um componente de *Unconditional Garment Design* e um componente de *Garment Try-On*. Primeiro, o componente de *Unconditional Garment Design* gera imagens de novas peças de roupa. Em seguida, estas peças de roupa são utilizadas como input para o componente de *Garment Try-On*, que produz imagens de modelos de moda a vestir essas peças de roupa. Assim, o pipeline proposto auxilia o trabalho dos estilistas de moda, permitindo-lhes basear novos produtos em roupa gerada por IA. Como resultado, o processo de criar novos produtos fica mais rápido, levando ao aumento da produtividade dos estilistas.

Foi realizado um inquérito de forma a avaliar o desempenho da *pipeline* de IA proposta. Foram feitas perguntas a 38 voluntários sobre i) a realidade e ii) a comercialidade da roupa gerada pela *pipeline*. Cerca de 52% dos voluntários falharam em distinguir vestuário real de vestuário gerado pela *pipeline* na maioria das perguntas. Além disso, a comercialidade do vestuário real e gerado tal como indicado pelos voluntários é semelhante. Em suma, a *pipeline* proposta gera novas peças de vestuário realistas que satisfazem as necessidades de moda dos consumidores.

Keywords: Inteligência Artificial, Redes Neurais, Moda, Design Gerativo, Design de Moda

Acknowledgements

Throughout the writing of this dissertation, I have received an unmeasurable amount of support and assistance. I wish to thank the people who contributed to the success of this dissertation, both directly by advising me and indirectly by just being present in my life.

Firstly, I would like to thank my supervisor, Prof. Rui Maranhão, for all the kind guidance he provided me during this chapter of my life. Thank you for your guidance and continuous support. I extend my gratitude to Abubakar Zakari. Thank you for all the time you spent overseeing my research work and for steering me on the path of success.

I desire to thank Orlando Ribas from XNFY Lab for his unconditional support during the latest months and for providing the computing resources essential to running this thesis experiments. Thank you for your compassionate attitude and consistent encouragement.

I would also like to thank my family and all of my friends. Thank you from the bottom of my heart for shaping the person I am today. The success of this thesis was only made possible by all of you.

Thank you,
Fábio Oliveira

*“There is a time for work, and a time for love.
That leaves no other time.”*

Coco Chanel

Contents

1	Introduction	1
1.1	Background	1
1.2	Problem Statement	2
1.3	Motivation	3
1.4	Research Questions	4
1.5	Research Objectives	4
1.6	Document Structure	4
1.7	Chapter Summary	5
2	Fundamentals	7
2.1	Machine Learning	7
2.1.1	Overview	7
2.1.2	Standard Tasks	8
2.1.3	Types of Machine Learning Systems	9
2.1.4	Evaluating Performance	12
2.1.5	Summary	13
2.2	Artificial Neural Networks	14
2.2.1	Basics	14
2.2.2	Training	17
2.2.3	Deep Learning	18
2.2.4	Regularization	18
2.2.5	Convolutional Neural Networks	19
2.2.6	Summary	20
2.3	Generative Adversarial Networks	21
2.3.1	Basics	22
2.3.2	Influential GAN variants	23
2.3.3	Improving Training	25
2.3.4	Evaluating performance	27
2.3.5	Summary	30
3	AI for Fashion: Literature Review	31
3.1	Pose Transfer	31
3.2	Garment Translation	32
3.2.1	Garment Try-on	33
3.2.2	Garment Transfer	34
3.3	Garment Design	35
3.3.1	Unconditional Garment Design	35
3.3.2	Garment Design Conditioned on Text	36

3.3.3	Garment Design Conditioned on Existing Clothing	36
3.4	Garment Retrieval	37
3.5	Garment Attribute Manipulation	38
3.6	Datasets	39
3.7	Summary	40
4	Methodology	43
4.1	Research Process	43
4.2	Proposed Pipeline	45
4.3	Unconditional Garment Design	45
4.3.1	Generator Architecture	46
4.3.2	Discriminator Architecture	50
4.3.3	Regularization	52
4.3.4	Sampling Techniques	53
4.3.5	Loss Functions	55
4.4	Garment Try-On	55
4.4.1	Geometric Matching Module (GMM)	56
4.4.2	Try-On Module (TOM)	57
4.5	Summary	58
5	Experimentation	59
5.1	Data Collection	59
5.2	Libraries Used	59
5.3	Experiments	60
5.3.1	Experiment 1: Unconditional Garment Design	60
5.3.2	Experiment 2: Garment Try-On	62
5.4	User Survey	64
5.5	Summary	65
6	Results and Discussion	67
6.1	Experiment 1: Unconditional Garment Design	67
6.1.1	First Training Stage	67
6.1.2	Second Training Stage	68
6.2	Experiment 2: Garment Try-On	70
6.3	User Survey	73
6.3.1	Question Group 1	73
6.3.2	Question Group 2	76
6.3.3	Results Summary	77
6.4	Summary	77
7	Conclusion	79
7.1	Summary of Research Findings	79
7.2	Research Contributions	80
7.3	Threats to Validity	81
7.4	Challenges Faced	81
7.5	Future Research Directions	82
7.6	Final Words	83
A	User Survey	85

List of Figures

1.1	Double jeopardy concerning fashion brands that are slow to market.	3
2.1	Traditional Programming compared to Machine Learning.	8
2.2	Training a spam filter with a labeled dataset.	10
2.3	Applying a clustering algorithm to group similar users.	11
2.4	Using unlabelled data points to augment the performance of supervised learning.	12
2.5	Reinforcement Learning training overview.	13
2.6	Classifying a new point in an instance-based learning system.	14
2.7	An illustration of an artificial neuron with 3 input connections.	15
2.8	An illustration of an Artificial Neural Network with 2 hidden layers	16
2.9	An illustration of the convolution operation.	20
2.10	An illustration of the transposed convolution operation.	20
2.11	An illustration of a 2x2 average pooling operation applied to a 4x4 input.	21
2.12	Overview of the GAN game between Generator and Discriminator.	22
2.13	The two principles which inspired the Inception Score.	29
3.1	Overview of the task of Pose transfer.	32
3.2	Overview of the two problems of Garment Translation.	33
3.3	Overview of the task of Garment Try-on.	33
3.4	Overview of the task of Garment Transfer.	34
3.5	Overview of the task of Unconditional Garment Design.	36
3.6	Overview of the task of Garment Design Conditioned on Text.	36
3.7	Overview of the task of Compatible Garment Design.	37
3.8	Overview of the task of Garment Redesign.	37
3.9	Overview of the task of Garment Retrieval.	38
3.10	Overview of the task of Garment Attribute Manipulation.	39
4.1	Overview of the research process conducted.	44
4.2	Overview of the Proposed Solution.	45
4.3	Architecture of the Mapping Network of the XGAN Generator.	47
4.4	Architecture of the Synthesis Network of the XGAN Generator.	48
4.5	Overview of the Style Block component of the XGAN Generator.	49
4.6	Overview of the ToWavelets component of the XGAN Generator.	49
4.7	Overview of the Discriminator network of XGAN.	50
4.8	Overview of the Discriminator Block component of the XGAN Discriminator.	51
4.9	Overview of the Discriminator Epilogue component of the XGAN Discriminator.	52
4.10	Overview of the Data Augmentation regularization method.	54
4.11	Overview of input pre-processing of the CP-VTON+ framework.	56

4.12	Overview of the inputs and outputs of the Geometric Matching Module of the CP-VTON+ framework.	57
4.13	Overview of the inputs and outputs of the Try-On Module of the CP-VTON+ framework.	58
5.1	Sample of images from the dresses dataset.	61
5.2	Sample of images from the women upper apparel dataset.	62
5.3	Sample of image pairs from the women upper apparel dataset.	63
5.4	Evaluation process for the Garment Try-On Component.	64
6.1	FID metric per training epoch of Experiment 1.	68
6.2	Generated images during the first training stage of Experiment 1.	69
6.3	Generated images during the second training stage of Experiment 1.	70
6.4	Generated Try-On images of Experiment 2.	72
6.5	User survey answers for the single selection questions.	74
6.6	Number of correctly answered single selection questions.	75
6.7	Answers for the second question group of the user survey.	76

List of Tables

2.1	Common practical applications of Machine Learning.	9
3.1	Comparison of some widely used fashion datasets.	41
5.1	Hyperparamaters for the different Garment Try-On training runs.	63
6.1	Performance results for Experiment 2	71
6.2	Percentage of correct answers for the single selection questions.	73

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
CAD	Computer-Aided Design
CNN	Convolutional Neural Network
DL	Deep Learning
EMA	Exponential Moving Average
FID	Fréchet Inception Distance
GMM	Geometric Matching Module
GAN	Generative Adversarial Network
ML	Machine Learning
PoC	Proof-of-concept
IS	Inception Score
RGB	Red Green Blue
SSIM	Structural Similarity Index Measure
TOM	Try-On Module

Chapter 1

Introduction

1.1 Background

Fashion is one of the most essential arts of civilization, no less than painting and sculpture. It plays an important role in our daily lives from the moment we get dressed, and for some, it is part of their identity. Emotions and personal feelings can be broadcast using the appropriate set of apparel. Political figures have long recognized the significance of clothing, using them to manipulate the public and show desirable personality types [74].

Explorers and travelers were the first to document the dress styles of the different societies they encountered while making expeditions around the world. Some of them brought back drawings and samples of clothing, sparking off a desire to understand them [40]. Eventually, the study of clothing came to be an essential part of anthropology, providing guidance to comprehend different cultures and traditions [6].

Nowadays, consumers look at fashion as wearable art and style as a form of artistic expression. However, that was not always the case. Before the 18th century, consumers had to make their own clothes or buy them from a small set of local producers. Naturally, they had few options, and the social constructs of fashion and style played little importance in their daily lives. Standardized clothes were the norm for the working class, and expressing identity through clothing was reserved for the upper classes [74]. Later, with the industrial revolution, the trends shifted. It became possible to mass-produce clothes, which allowed fashion brands to flourish and consumers to choose from great diversity [44].

More recently, with the progress of science and technology, these domains have ventured into fashion leading to the emergence of fashionable technologies. Thus, fashion and technology have become inter-connected [80]. The car industry collaborates with fashion designers, and fashion brands design mobile phones [13]. Furthermore, it is becoming increasingly common to see wearable technology forms being developed and adopted as forms of adornment. Smartwatches are one example of this; some are even capable of substituting phones [14]. Whether they should be classified as a technology product or a fashion product is up for debate [16]. The lines that separate

fashion from science and technology are becoming blurred as a result of the ongoing technological revolution.

It can be unarguably stated that fashion is intrinsic to society's beliefs and culture. Consequently, fashion trends evolve organically and are influenced by many unpredictable factors. The era of globalization has brought the consumer in-vogue style at low prices but at the cost of standardization. Fashion is no longer determined by regional factors, but by global capitalism [6]. What might have been deemed stylus apparel in the past is unlikely to be considered fashionable as of today and even less so tomorrow. To satisfy customers, big retail brands such as *Zara* and *H&M* renovate their collections with ever-shorter intervals, sometimes as often as a month [87]. Nevertheless, companies struggle to remain ahead and predict the trends that are yet to come.

The fast-changing needs of consumers introduces variance that is hard to manage. The fashion industry does its best to forecast demand to avoid financial losses arising from unsold stocks. Whether brands fail or succeed is largely defined by their flexibility and responsiveness [65]. Fashion markets are highly competitive and exhibit the following characteristics, making them synonymous with rapid change [18]:

- *Short life-cycles* – clothes are designed to capture the mood of the moment and thus have a shelf life of a few weeks or months before they become undesirable by consumers.
- *High volatility* – demand for products is rarely stable and may be influenced by arbitrary external factors such as the weather, movies, and footballers.
- *Low predictability* – due to the volatility of demand, it is almost impossible to predict future trends.
- *High impulse purchasing* – many buying decisions by consumers are made when they are confronted with the product and not planned. Availability of stock is thus crucial.

In order to remain competitive in the fast-paced fashion world, brands seek new methods to decrease the time to market of new apparel. One way to achieve that is by shortening the design stage of the fashion process. Providing fashion designers with already existing prototypical clothing, which they can then fine-tune and refine to consumer markets' needs, leads to a reduction in the design time of new products. On this account, the present work proposes and implements a pipeline that unites the tasks of i) unconditional generation of clothing and ii) showing the generated clothing on real-world human models.

1.2 Problem Statement

To address the volatility of fashion markets, fashion retailers used to rely mainly on their capability of forecasting demand months ahead of the actual time of purchase by the consumer [28]. However, the fierce competition in the fashion industry aligned with the low accuracy of forecasts gave a change to a new paradigm – *fast fashion* – characterized by the ambition of bringing clothes as fast as possible to market. This change provided consumers with the latest fashion trends from fashion shows in a timely manner never seen before [8]. Nonetheless, quick is never too quick, and

companies keep trying to reduce time to market in a continuously improving process to stay ahead of the competition. Otherwise, they risk facing financial losses in two ways. First, they might miss significant sales opportunities, and second, when the products arrive at stores, consumer demand might be fading, leading to the likelihood of mark-downs [18]. As such, new methods that reduce the time-to-market of new fashion products are sought by fashion brands to remain competitive in the volatile fashion world. Figure 1.1 illustrates this double jeopardy.

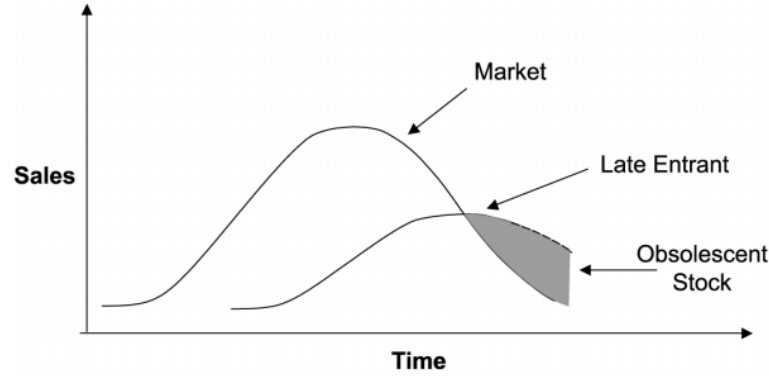


Figure 1.1: Short fashion life-cycles makes time-to-market crucial. Reproduced from [18].

The fashion process encompasses multiple stages, from designing clothing to delivering them physically to stores. Improving the shipping logistics stage of the fashion process is difficult. However, the design phase of fashion apparel can be improved by novel AI methods that generate prototypical clothing and display such clothing on fashion models, thereby helping fashion designers conceive new garments faster and thus reduce the time to market of new fashion apparel. Nevertheless, no existing AI methods can perform both tasks: generating clothing and displaying the clothing on fashion models.

1.3 Motivation

As a way for brands to reduce time to the market, computer-aided design (CAD) has emerged as the *de facto* standard for the design of new clothing items [38]. Furthermore, CAD decreases human efforts and financial costs in the fashion industry [86]. Recent advances in the field of Artificial Intelligence (AI) present novel opportunities to complement existing CAD software to further shorten the fashion design process. Contemporary literature works have proved that Artificial Neural Networks (ANNs), a type of Machine Learning algorithm, are able to synthesise high quality clothing [82, 107, 2] and dress real-world models with different garments [30, 90, 59]. The unpredictability of fashion markets motivates the need for novel AI-based methods that can reduce the design time of new garments.

With the above considerations in mind, the present work proposes and implements an ANN-based pipeline for i) the automatic generation of fashion garments and ii) displaying those garments on real-world models. We argue that this proposed solution can bring tremendous value to

the whole fashion process by reducing the time to market of new fashion products by speeding up the design phase, which in turn means that brands item catalogs' become more aligned with ongoing consumer trends. Moreover, the clothes' prices can be lower because designing clothes becomes cheaper due to requiring less human intervention. The clothes designs' quality may also be improved because fashion designers can target their efforts to already existing generated clothing and spend more time fine-tuning details to meet consumer demands.

1.4 Research Questions

The following research questions are defined to guide the research work of the present dissertation:

- **Research Question 1** - What are the existing studies on fashion datasets and the applications of AI to fashion?
- **Research Question 2** - How can ANNs be used to automatically generate clothing?
- **Research Question 3** - How can ANNs be used to dress fashion models with different clothing?
- **Research Question 4** - What is the performance of the proposed ANN-based pipeline in terms of performance metrics and user surveys?

1.5 Research Objectives

Considering the formerly defined research questions for this dissertation, a list of research objectives are proposed:

- **Research Objective 1** - To investigate the existing studies on fashion datasets and the application of AI to fashion.
- **Research Objective 2** - To propose and implement an ANN-based pipeline that sequentially performs i) automatic generation of clothing items and ii) dressing fashion models with the generated clothing.
- **Research Objective 3** - To evaluate the performance of the proposed ANN-based pipeline in terms of performance metrics and user surveys.

1.6 Document Structure

In this section, a brief overview of the entire thesis is given. Hence, the thesis is structured into six chapters with the goal of providing a comprehensive view of this research.

Chapter 1 - Introduction

In this chapter, the background and motivation of this research are outlined. Furthermore, the problem statement is also provided. This chapter also gives the research questions and objectives of the study.

Chapter 2 - Fundamentals

This chapter presents a general overview on Machine learning fundamentals, which is vital for this study. The types of machine learning systems were also outlined and discussed. The chapter further discusses ANNs with details on CNNs and its basic layers. Studies utilizing these networks were also highlighted in detail. The chapter also gives a basic background literature on Generative Adversarial Networks (GANs) and some related works on its existing applications.

Chapter 3 - AI for Fashion: Literature Review

This chapter gives a detailed literature review on AI for fashion. Sub-domains such as pose transfer, garment translation, garment design, garment retrieval, and garment attribute manipulation were discussed in detail. Furthermore, the studies and works on these domains were also highlighted and discussed in detail. Lastly, the chapter outlines existing datasets that are used for fashion in AI.

Chapter 4 - Methodology

The general methodology utilized in conducting this research is discussed in detail. Hence, each stage of the research process is broadly discussed from the literature review, down to the problem analysis, methodology, and the evaluation of the proposed pipeline. Lastly, the proposed pipeline methodology, which comprises two components that perform the Unconditional Garment Design and Garment Try-On tasks, is presented in detail.

Chapter 5 - Experimentation

This chapter describes the distinct experimental setups carried out for the implementation and evaluation of the proposed pipeline. The chapter also highlights the training datasets and the software libraries used for the individual experiments of this research.

Chapter 6 - Results

This chapter presents and discusses the results of the different experiments carried out in this study. The evaluation based on the user surveys is also given.

Chapter 7 - Conclusion

This chapter focuses on providing a general summary of the research findings with the identified research limitations of the research. The chapter also outlines the thesis contributions with recommendations for future research.

1.7 Chapter Summary

Fashion is wearable art that provides us with self-esteem and confidence. It is a way to broadcast feelings and emotions, and for some, it is part of their identity. Nevertheless, fashion is highly volatile. Garments lose their stylistic appeal as each day passes and new designs emerge. Fashion brands are always trying to stay one step ahead and predict what the consumer markets will desire in the future. However, this is hindered by the fact that fashion markets are inherently unpredictable. One way to mitigate this issue is for brands to design and ship clothes faster to retailers. This thesis proposes and implements an AI method, namely an ANN-based pipeline, that generates fashion garments and dresses them on fashion models. This proposed solution complements and speeds up the current workflow of fashion designers by allowing them to start design efforts from

existing prototypical clothing. In turn, this leads to a reduction in design time and time-to-market of new fashion apparel, helping fashion brands tackle the volatility of consumer markets.

Chapter 2

Fundamentals

This chapter focuses on the concepts of AI that compose the building blocks of this work. The chapter starts by presenting an overview of Machine Learning, which is the field of AI that studies algorithms that learn from data. Furthermore, the chapter also discusses Artificial Neural Networks, which is a type of Machine Learning algorithm inspired by the human brain. Lastly, the chapter familiarizes the reader with Generative Adversarial Networks, which is a discipline that stems from the intersection of ANNs and Generative Models.

2.1 Machine Learning

The field of Machine Learning (ML) is attracting an ever-increasing number of companies looking to extract information from big amounts of data efficiently and with little supervision from humans. Furthermore, ML algorithms are becoming increasingly sophisticated and have recently surpassed human performance in complex tasks, such as image and speech recognition in the last years [48]. As such, some argue that ML is driving the new industrial revolution [88, 19].

In the following subsections, the fundamentals of ML will be presented.

2.1.1 Overview

ML is a branch of Artificial Intelligence concerned with the study of algorithms that learn without being explicitly programmed. This definition was first presented in 1959 by Arthur Samuel [79]. A recent and more precise characterization of ML algorithms is provided by Tom Mitchell [61]:

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T , as measured by P , improves with experience E .

– Tom Mitchell, *Introduction to Machine Learning* (1997)

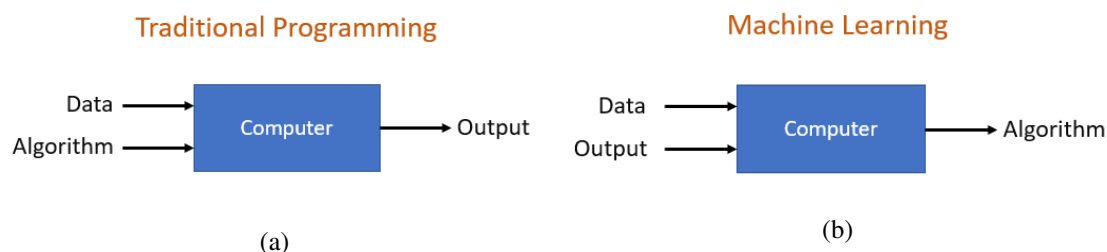


Figure 2.1: Traditional Programming (a) compared to Machine Learning (b).

In the traditional programming paradigm, depicted in Figure 2.1a, an algorithm handcrafted by a programmer is applied to the data, and the output is the result of the computation. In contrast, an ML system creates an algorithm from the data and the corresponding output, as depicted in Figure 2.1b. Briefly, in traditional programming, a developer creates the algorithm, whilst in ML, an algorithm is automatically created based on the data, thus requiring less human efforts.

The practical applications of ML are vast and new ones keep surging as the field progress. The most common ones are listed in Table 2.1.

2.1.2 Standard Tasks

When applying ML to a new problem, one must first understand which type of task underlies the problem because that will dictate which algorithm can be considered; most can only be used in specific scenarios. Most problems can be classified as follows [63, 64]:

- **Regression** is the task of predicting a real numeric value. E.g., predicting house prices given characteristics of the house, such as the number of bedrooms and the size of the living space.
- **Classification** is probably the most common ML task. It consists of assigning a category to data. For instance, attributing categories to documents and images is a classification task.
- **Ranking** is the task present in scenarios that involve learning to order items according to some criterion. One popular scenario arises in web search engines where a user enters a search query, and an ordered set of the most relevant web pages is returned.
- **Clustering** involves assigning categories to data, but unlike classification, no information about the categories is known before-hand. Some algorithms that perform clustering also do not require knowing the number of categories in advance. Customer segmentation is a typical scenario in which clustering techniques are used.
- **Dimension Reduction** consists of reducing the dimensionality of the data, losing the least amount of information possible. A common use case for dimension reduction is video preprocessing, which results in compressed videos that are drastically inferior in file size without incurring a severe loss of quality.

Machine Learning Applications	Use Cases
Document classification	• Assigning a topic to a document. • Classifying if a news article is true or fake. • Detecting spam and phishing emails.
Natural language processing	• Labeling words according to their part of speech. • Recognizing named entities in text. • Determining the parse tree of a given sentence.
Speech Processing	• Translating spoken language into text. • Synthesizing speech from sentences. • Identifying speakers in a speech corpus.
Computer Vision	• Classifying objects in images. • Identifying faces in photographs. • Optical character recognition (OCR).

Table 2.1: Common practical applications of Machine Learning.

2.1.3 Types of Machine Learning Systems

Having a clear understanding of what aspects define the different types of ML systems is important when choosing which one to use in tackling a given problem. What makes one type of system more suitable for a specific scenario might make it inadequate for others. Due to the many different types of systems, grouping them into categories is nontrivial. Nevertheless, several criteria have been proposed to classify them according to:

- **The amount of human supervision they receive during training** (supervised, unsupervised, semi-supervised, and reinforcement learning).
- **The capability to incrementally learn on the fly** (online versus batch learning).
- **How they generalize to unseen data** (instance-based versus model-based learning).

Next, an overview of each of the criteria used to classify the systems is presented.

2.1.3.1 Classification according to the amount of supervision received during training

The principal way to classify ML systems is according to the amount of human supervision they receive during training. The four main paradigms, supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning, are characterized as follows [84, 63, 77]:

- **Supervised Learning** is the type of learning in which the system sees a set of labeled examples during training, i.e., it is provided with the input data along with the desired solutions, the ground truth. This presents the most common scenario associated with classification

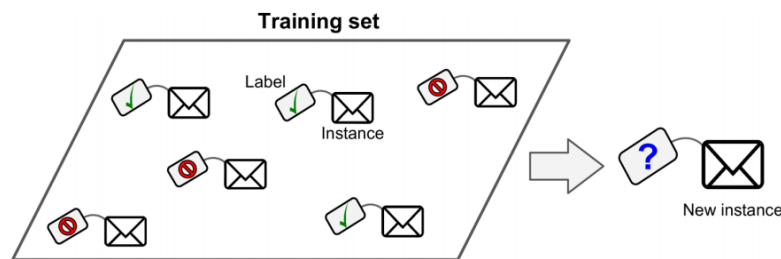


Figure 2.2: Training a spam filter with a labeled dataset (an example of supervised learning). Reproduced from [25].

and regression tasks. For instance, spam filters are often trained on a dataset of emails containing their respective labels (spam or non-spam) and must learn to classify new emails, as illustrated in Figure 2.2. The main challenge arising in this type of learning is the difficulty of obtaining large amounts of labeled data in some applications (e.g., automatic vehicle driving and medical diagnostics) [25]. Moreover, an ever-increasing amount of information is created every day, and most of it is unstructured and thus unlabeled, posing a hindrance for supervised learning. For these reasons, many influential AI scientists believe that the future of ML lies in unsupervised learning [31].

- **Unsupervised Learning** dictates that a set of unlabeled examples is presented to the system during training, and it has to learn the structure of the data in order to make predictions on unseen data. One popular application of unsupervised learning is clustering. For instance, we may have a dataset containing the behavior of some website visitors, and we wish to segment the visitors into groups that exhibit common characteristics. This scenario is presented in Figure 2.3. One big drawback present in this type of learning stems from the fact that few or no labeled examples can be used to test the system's performance. As such, it becomes difficult to evaluate the system and verify it is working as intended.
- **Semisupervised Learning** falls somewhere between supervised learning and unsupervised learning. The system receives a set containing both labeled and unlabeled data during training. Thus, semisupervised learning arises in settings where unlabeled data is cheap to obtain but labeling it is costly. The underlying concept behind this type of learning is that augmenting the labeled dataset with unlabeled examples will increase the system's performance, as illustrated in Figure 2.4. In practice, it is hard to employ because most supervised learning algorithms cannot leverage unlabeled data.
- **Reinforcement Learning** is a paradigm that drastically differs from others in the sense that it resembles more the way humans learn - through interactions with the environment. As such, it is more suitable for physical real-world tasks such as robotics. In reinforcement learning, the ML system is called an agent, and training occurs in many steps through interactions with the environment (which may be simulated), which result in a set of rewards. The objective of the agent is to maximize the sum of rewards across a set of sequentially

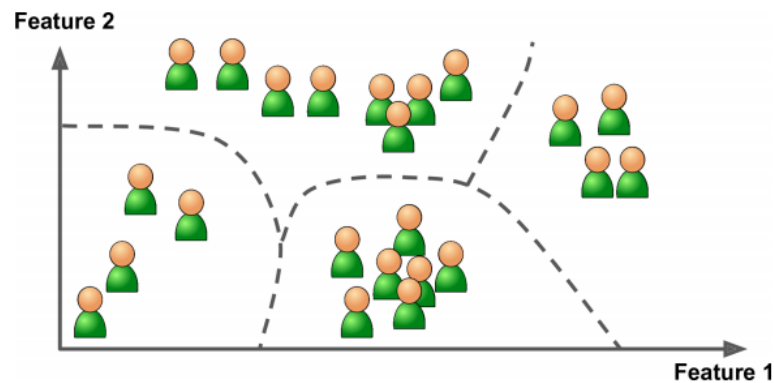


Figure 2.3: Applying a clustering algorithm to group similar users according to some similarity metric (an example of unsupervised learning). Reproduced from [25].

performed actions. The strategy the agent learns, called a policy, is a mapping from environment stimuli to actions. An overview of this paradigm is presented in Figure 2.5.

2.1.3.2 Classification according to the capability of incrementally learning

Another important criterion to classify ML systems is according to whether or not the system can learn from an incoming stream of data. In most cases, this is not an important distinction to think about, but in some scenarios, it may be at the core of the decision in selecting which ML algorithm to use. Two main categories, which are batch and online learning, are presented [63, 25]:

- **Batch learning** is a paradigm in which the system is not able to learn incrementally as new data points are added to the dataset and must be retrained using all the available data every time it needs to be updated. This may take a lot of time and computing resources. Hence, it poses a challenge because in some scenarios, the system must inherently learn as new data arrives. For instance, in recommendation systems, when a user performs an action, it is expected that the new recommended content is presented in a timely manner.
- **Online learning** differs from batch learning in the sense that the system can be incrementally trained from an incoming stream of new data points. Each learning step is fast and computing-efficient, as such, the system learns about new data on the fly just as it arrives. Another key point that separates online learning algorithms from most batch learning ones is the ability to learn from datasets that may not entirely fit in memory. At each training step, only part of the data is required, hence, they are suitable for big datasets.

2.1.3.3 Classification according to how they generalize to unseen data

The last criterion used to classify ML systems is by the method they generalize to unseen data. Since the fundamental task of ML is to make predictions on unlabelled data points not seen during training, systems that generalize better perform better. Thus, according to this criterion, ML

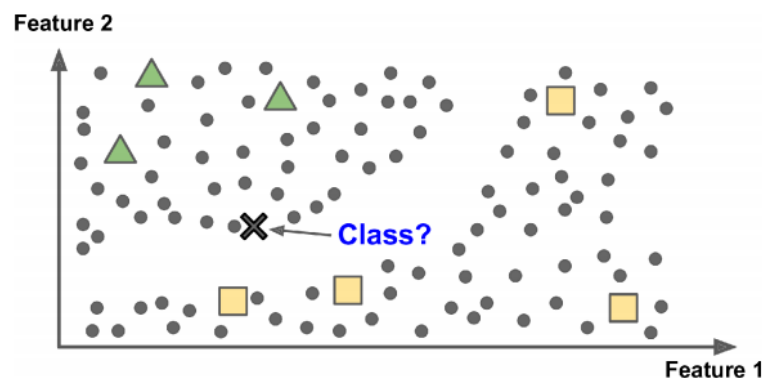


Figure 2.4: The colored and grey points represent labeled and unlabeled data, respectively. The ML system can leverage the heterogeneity of the data to make more accurate predictions (an example of semisupervised learning). Reproduced from [25].

systems can be classified into two paradigms, which are instance-based learning and model-based-learning, described as follows [25]:

- **Instance-based learning** is the simplest form of learning as it does not involve figuring out patterns in the data. During training, the ML system only needs to memorize the training data, hence, the training is almost instantaneous. As a result, this type of system tends to grow linearly in size with relation to the dataset size. All memorized data are compared to the unseen data point according to some similarity metric to make predictions. An output is then calculated by taking in the degree of similarity between the points. In summary, Training is fast, but predictions take long, making these types of models unsuitable for real-time scenarios. Figure 2.6 illustrates this type of learning.
- **Model-based learning** is different from instance-based learning, because generalization occurs by learning patterns from the data and building a model. For instance, to classify objects in images, the system may learn to look for characteristics such as the number of edges present in the image. These algorithms are also known as parametric algorithms, because a set of parameters are learned during training, composing a model.

2.1.4 Evaluating Performance

Evaluating ML systems before they are deployed to a real-world production environment is important because they may perform much worse on unseen data - a phenomenon called **overfitting**. Poorly performing systems may result in consequences ranging from financial losses to life-threatening (e.g., stock market portfolio management and medical diagnosis). Thus, it is crucial to know before-hand how the system will behave after training. This is because in some scenarios, their performance may not be acceptable enough to merit their use.

One prevalent method to gauge the performance of a ML system is to evaluate it using data not provided during training. To do so, the dataset that would be used to train the model is split into two

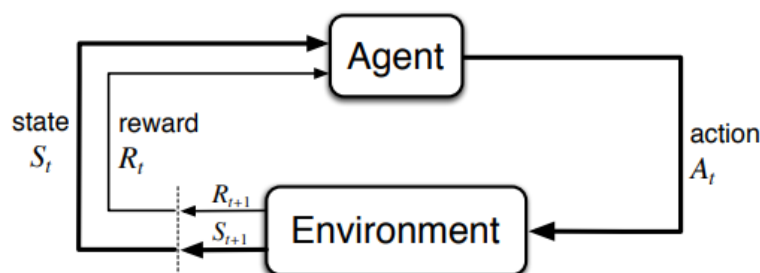


Figure 2.5: In reinforcement learning, the agent learns through rewards received by interacting with the environment. Reproduced from [84].

parts: a **training set** and a **validation set**. The system is trained on the former, and its performance is calculated from the latter. This gives an unbiased view of the system's performance because the system does not use the data to evaluate it during the training phase. Such methodology is named **holdout cross-validation**. However, some other evaluation methods exist. For instance, in **k-fold cross-validation**, instead of 2 sets, k sets are created and k iterations are then performed. In each iteration, the system is trained using $k-1$ sets, and the remaining one is used for evaluation. This results in k measurements of performance, and their arithmetic mean is the final performance measure. Additionally, one can also calculate the standard deviation of the multiple measurements to gain more insights about the variance of the system.

Performance metrics quantify performance and are specific to the kind of task the system is supposed to perform. Mean squared error and accuracy are the most common ones for regression and classification, respectively.

2.1.5 Summary

ML is the field of AI that studies algorithms that learn from data without being explicitly programmed. The number of tasks ML can be used for is vast and is ever-growing. Some common ones include **classification** (e.g., sentiment analysis of movie reviews) and **regression** (e.g., predicting stock prices). ML systems are first trained on some dataset, and then they are able to make predictions for new data points. Their performance can be quantitatively measured using performance metrics, such as **mean squared error** and **accuracy**. Usually, **holdout cross-validation** is used to get an unbiased view into the system's performance after they are trained and before they are deployed to a real-world setting. This validation method consists of splitting the data available for training into two parts, a **training set** and a **validation set**. The systems are trained on the former and evaluated on the latter with the use of **performance metrics**.

ML systems can be classified according to: the **amount of supervision they receive during training** (supervised, unsupervised, semi-supervised and reinforcement learning), **ability to learn incrementally** (batch vs online learning) and **how they generalize to unseen data** (instance-based vs model-based learning).

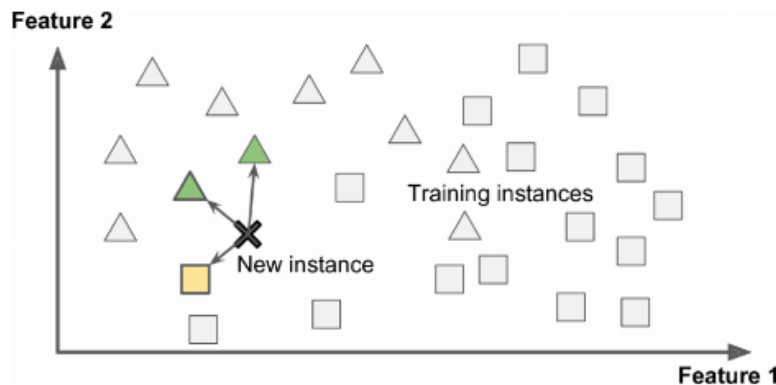


Figure 2.6: Classifying a new point in an instance-based learning system. All training examples are compared to the point that is to be classified. Reproduced from [25].

2.2 Artificial Neural Networks

ANNs are at the core of most recent AI breakthroughs. They made a remarkable impact in the fields of Computer Vision and Natural Language Processing, and their applications include state-of-the-art autonomous driving and language translation. Despite ANNs being ML algorithms, a big difference set them apart from the rest, namely, they perform end-to-end feature extraction, meaning that feature engineering is rarely needed when employing them, reducing the amount of domain expertise needed to practice ML. As such, they can be applied to a myriad of problems with minimal tinkering, which has led to their mass adoption by companies looking to build intelligent systems.

The rest of this section introduces the reader to the fundamentals of ANNs.

2.2.1 Basics

ANNs are a type of ML algorithms that can be applied for unsupervised, supervised, semi-supervised, and reinforcement learning tasks. Furthermore, they learn incrementally and generalize to unseen data by building a model based on patterns learned from the data (more on the classification of ML systems can be found Chapter 2.1.3).

The building blocks of ANNs are named **neurons** due to their resemblance with the biological neuron. These artificial neurons are arranged in a direct acyclic graph; as such, information flows only in one direction, from the input neurons to the output neurons. This information takes the form of decimal numbers. A simple depiction of the neuron is presented in Figure 2.7.

Each neuron receives information from many other neurons as input and calculates a function of those inputs, referred to as the **neuron's activation**. The computation of the activation of a neuron is done in two steps:

- First, the inputs coming from the connected neurons are multiplied by a learnable **weight matrix**.

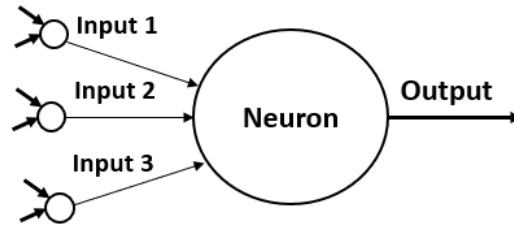


Figure 2.7: An illustration of an artificial neuron with 3 input connections.

- The results are summed, and the resulting scalar is passed through a nonlinear function referred to as the **activation function**.

This operation is defined in equation 2.1, where y is the resulting neuron activation, x_i is the activation of input neuron i and w_i is the respective weight associated with that input neuron. A bias term θ is added to the weighted sum of the inputs. f is an activation function. Having a weight matrix W allows the neuron to choose to pay more attention to some inputs and less or no attention to others.

$$y = f\left(\sum_i x_i w_i + \theta\right) \quad (2.1)$$

Activation functions act as a way to introduce nonlinearity operations into the network, increasing their representational power. In the absence of activation functions, the output of an ANN would be a linear calculation of its input, as such, stacking multiple neurons would yield no performance improvements because the result of many algebraic linear operations can be expressed by a single linear operation (i.e., multiple layers of neurons would not be more expressive than a single layer).

Popular activation functions include:

- **Sigmoid function:**

$$f(x) = \frac{1}{(1 + e^{-x})} \quad (2.2)$$

- **Hyperbolic Tangent (Tanh):**

$$f(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})} \quad (2.3)$$

- **Rectified Linear Unit (ReLU):**

$$f(x) = \max(0, x) \quad (2.4)$$

- **Leaky Rectified Linear Unit (LReLU)**, where γ is an hyperparameter.

$$f(x) = \begin{cases} -\gamma x & , \text{if } x \leq 0 \\ x & , \text{if } x > 0 \end{cases} \quad (2.5)$$

Neurons are arranged in a variable number of layers. The neurons of layer i receive as input the activation of the previous layer $i - 1$. A layer is classified as **fully connected** when each of its neurons are connected to *all* the neurons of the previous layer. Furthermore, the layers can also be classified according to their position in the ANN:

- The **input layer** is where data enters the network. Unlike the other layers, the input layer activations are calculated directly from input data and not from the activation of other neurons.
- The **hidden layers** are the middle layers of the network and perform intermediate computations.
- The **output layer** presents the output of the ANN, which can be a single scalar value or a vector of multiple scalar values. The output layer contains as many neurons as the number of outputs (1 in the case of a single scalar value or n neurons if the output is a vector of dimension n).

Figure 2.8 depicts an ANN composed of several fully connected layers.

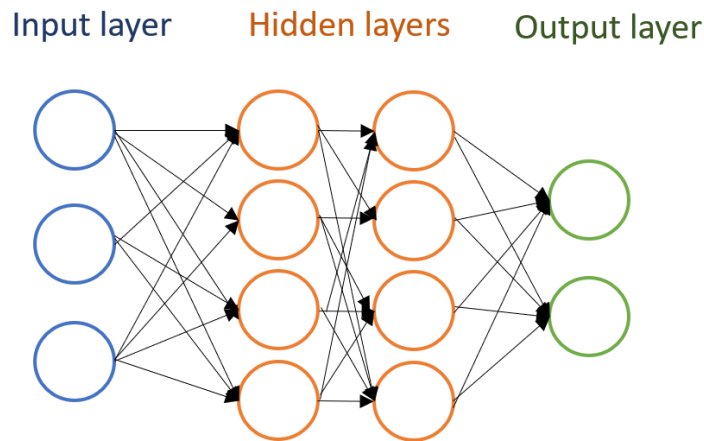


Figure 2.8: An illustration of an ANN with 2 hidden layers.

Typically, the neurons of the output layer use the **softmax** activation function when the output of the network defines class probabilities (and thus all output neurons' activation need to sum to 1). No activation function is used when the output is a simple scalar value, as is the case in regression tasks. Equation 2.6 presents the softmax function, where y_i is the intermediate output of neuron i

defined by the weighted sum of its inputs with its weight matrix W .

$$f(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \quad (2.6)$$

Note that with the softmax function, the activation of one neuron depends on the other neurons of the same layer, unlike the previously presented activation functions. Softmax is essentially a generalization of logistic regression for classification problems with more than 2 classes.

2.2.2 Training

ANNs are trained to minimize the value of **loss functions**, which define how wrong the outputs of the networks are in relation to some function of the ground truth. Popular loss functions are described next:

- The **squared error loss** function is used when the output of the network is a scalar (regression), where y is the ground truth and $\hat{y}$ is the output of the network:

$$L = (\hat{y} - y)^2 \quad (2.7)$$

- The **cross-entropy loss** function is used when the output is a vector of probabilities (classification), where $\hat{y}_c$ is the value of the output neuron corresponding to the correct class C after applying the softmax function:

$$L = -\log(\hat{y}_c) \quad (2.8)$$

Training is performed using the backpropagation algorithm [75] and gradient descent:

- The **backpropagation algorithm** propagates the errors throughout the layers of the network.
- **Gradient descent** updates the network parameters to minimize the errors.

The errors of the network are represented in **gradients**, which are multi-dimensional vectors containing the derivative of the parameters of the network with relation to the loss function. The update for weight w_i in time step t is performed according to equation 2.9, where $\frac{dw_{i,t}}{dL}$ is the derivative of the weight with relation to the loss function, α is the **learning rating**, an hyperparameter which defines the magnitude of the updates to the weights.

$$w_{i,t+1} = w_{i,t} - \alpha * \frac{dw_{i,t}}{dL} \quad (2.9)$$

Training is complete (i.e., it has converged) when the loss function stops decreasing significantly ($L_t - L_{t+1} < \epsilon$). Gradient descent is not an exact method, and thus the values of the network's weights after convergence correspond only to a *local minima* of the loss function. Any function can serve as the loss function as long as it is differentiable, making ANNs suitable for a multitude of different problems, unlike other ML algorithms.

More complex optimization methods that leverage momentum are used to speed up the training time. One of the most widely used is the Adam [45] optimization method. The method proposes using an exponentially decaying average of past squared gradients in addition to momentum, resulting in bigger changes in weights which often do not receive significant updates (e.g., weights of neurons composing the first layers of the network).

2.2.3 Deep Learning

ANNs are universal function approximators and therefore can approximate any continuous function with arbitrary precision, with the only requirement being that the network contain a sufficient number of neurons and layers [9]. Nevertheless, training complex networks is a challenging task that remained unattainable until the last decade. An increase in the computational power of graphic cards joined with new techniques for training ANNs popularized **Deep Learning**, which is the practice of using ANNs with many hidden layers. The turning point came in 2012 when Alex Krizhevsky *et al.* [46] demonstrated that deep networks can be applied with great success to difficult image classification tasks, outperforming traditional computer vision algorithms. The network proposed by Alex Krizhevsky *et al.* achieved the first place in the ImageNet LSVRC-2012 challenge [76], a competition in which the objective was to develop algorithms capable of classifying images according to which object they represent. Each image of the ImageNet LSVRC-2012 dataset contains an object belonging to one of the 1000 possible categories. Notably, the network architecture designed by the authors - AlexNet - resulted in approximately half the error rate of competing approaches. Since then, new architectures have surpassed human performance in the ImageNet challenge [32].

2.2.4 Regularization

Adding more neurons or layers to an ANN tends to increase its performance in a task because the ANN can represent functions of increasing complexity. However, this usually comes at the cost of an increased risk of overfitting to the training data because the ANN can learn patterns from the noise present in the data. Regularizations are a class of techniques that address this issue. Some popular regularization techniques are presented next.

- **Weight decay** consists in adding a new loss term to the loss function to prevent the ANN's weight parameters from becoming too big. Usually, this term is given by the sum of the squared weights (L_2 regularization). The intuition behind weight decay is that large weights make the ANN's output heavily reliant on specific neurons' activation, making it generalize worse to unseen data.
- **Dropout** [83] sets the activation of some neurons to zero during training, with a probability given by an extra hyperparameter p . Therefore, at each training iteration, only a random subset of the ANN's neurons are used to calculate the output. Dropout reduces the co-adaptation of neurons on adjacent layers, encouraging their weights to be more uniformly

distributed, increasing the generalization of the ANN. At test time, each neuron's output is scaled, ensuring a deterministic output.

- **Data augmentation** consists in artificially increasing the size of the training dataset by applying transformations to the existing training examples. In the case of images, the set of possible transformations include rotation, cropping, and flipping the images in the X or Y-axis. The reasoning behind this regularization approach is that networks should generalize to small perturbations in the training set. Data augmentation is done just-in-time, meaning that the transformations are performed during the training cycle and not before. Thus, the dataset size on disk remains constant, making data augmentation a versatile method to improve the generalization of ANNs.
- **Batch Normalization** [36] makes networks converge faster by normalizing the output of hidden layers through re-centering and rescaling, which, in turn, allows the use of higher learning rates because the distribution of each hidden layer's weights becomes more uncorrelated as they rely on the normalized outputs of the previous layer. The normalization is performed across the entire batch of training examples, hence, the name Batch Normalization. This leads to non-deterministic behavior during training, introducing noise in the network and guiding the learning process to cope with it, reducing the risk of overfitting. During test time, however, the behavior is deterministic as the normalization is done in relation to a set of parameters learned during training.

2.2.5 Convolutional Neural Networks

Scaling ANNs to images is challenging due to the number of weights required to connect pixel values to neurons. **Convolutional Neural Networks** (CNNs) address this issue by using shared weights. All neurons that compose a layer share the same weight matrix. Furthermore, CNNs perform local convolution operations to minimize the number of weights needed to achieve good performance. Widely used CNNs layers include convolution, transposed convolution, and pooling layers:

- **Convolution layers** are comprised of a number of filters, also known as kernels, which perform convolution operations with the input data of the layer resulting in a number of **feature maps** equal to the number of filters. Each value of the resulting feature maps are passed through an activation function. The input to the first convolution layer of a CNN is usually a colored image containing 3 channels (RGB), each having two dimensions (width and height), i.e., the input is three 2-dimensional feature maps. In popular CNN architectures, the number of channels of the feature maps tends to increase as data flows through the network, but the width and height of the feature maps decrease. The filters are matrices of learnable weights, but unlike the case with fully connected layers, the same filter is applied to different patches of the input, resulting in shared weights. Figure 2.9 depicts a 3x3 filter being applied to a 4x4 input vector, resulting in a 2x2 feature map.

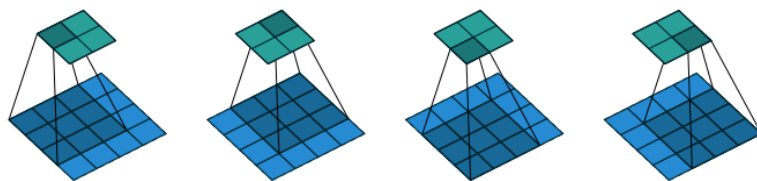


Figure 2.9: An illustration of the convolution operation. Input data is colored blue and the resulting feature map is colored green. Reproduced from [70].

- **Transposed convolution layers** are used for upsampling feature maps, increasing their dimensions (width and height). Unlike common upsampling methods such as bilinear interpolation or nearest-neighbor interpolation, transposed convolution upsamples its input according to learnable weights. This type of layer is used when the output of the CNN needs to be a multidimensional vector (for instance, in image-to-image translation tasks such as image segmentation). Transposed convolutions have the same connectivity pattern as normal convolutions, but its operation is done in the reverse order, increasing the dimension of the input instead of reducing it, as depicted in Figure 2.10.

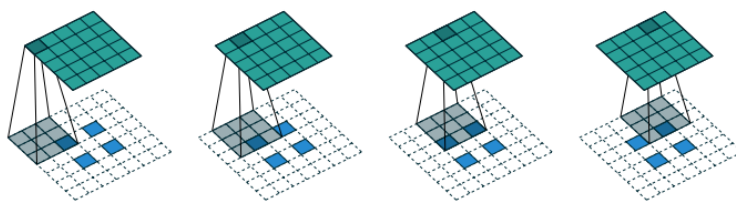


Figure 2.10: An illustration of the transposed convolution operation. Input data is colored blue and the resulting feature map is colored green. Reproduced from [70].

- **Pooling layers** have no weights and are used to downsample feature maps. They apply operations such as arithmetic averaging across nonintersecting patches of the input feature map, as shown in Figure 2.11. Pooling layers are used to make the internal representation of the data as it flows through the CNN more compact and spatially invariant.

2.2.6 Summary

ANNs are a type of ML algorithm inspired by the human brain. The building blocks of ANNs are **neurons**. Each neuron receives input from many other neurons in the form of decimal numbers, performs a weighted sum according to a **weight matrix** learned during training, and then passes the resulting value through an activation function in order to calculate its output, which is also known as the **neuron's activation**. Neurons are arranged in **layers**, and the neurons at layer i only receive input from neurons at layer $i-1$. A layer is designated as **fully connected** when each of its neurons is connected to all the neurons of the previous layer. The data in an ANN flows

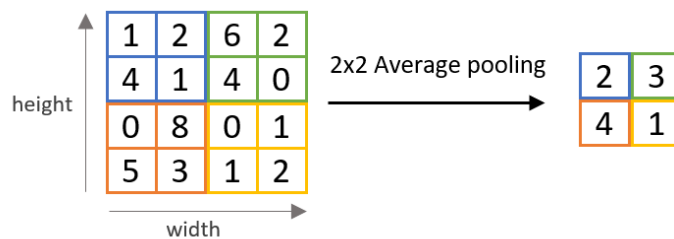


Figure 2.11: An illustration of a 2x2 average pooling operation applied to a 4x4 input.

from the **input layer** to the **hidden layers** and finally to the **output layers**. When the number of hidden layers exceeds two, the ANN is referred to as a deep ANN. Training of ANNs is performed using **gradient descent** and the **backpropagation algorithm** to minimize a **loss function**, which defines how much the output of the ANN deviates from the ground truth. Typical loss functions include **squared error** for regression and **cross-entropy** for classification.

Complex ANNs with many layers and many neurons present an increased risk of overfitting to the training data. To mitigate this issue, **regularization** techniques can be used to direct the learning process towards a solution that generalizes better to unseen data.

Convolutional Neural Networks (CNNs) are ANNs that perform well on images despite requiring less weights. Fully connected layers typically found in ANNs don't scale well to high-dimensional input such as images due to the amount of weights required for the neurons, which makes their training unstable. Popular CNNs layers include **convolution**, **transposed convolution**, and **pooling** layers.

2.3 Generative Adversarial Networks

Generative modeling consists of using a model to generate new examples that resemble those from an existing dataset, such as generating new photographs similar to those in a dataset of existing photographs. In discriminative modeling (e.g., regression and classification), the aim is to learn a model that approximates a conditional probability $P(Y|X)$ or a direct mapping from input x to ground truth y , whilst in generative modeling, the goal is to learn the prior distribution $P(X)$. Learning how to produce new realistic samples is fundamentally harder than learning how to make a decision given some input, making generative modeling a notoriously difficult problem in ML.

Generative Adversarial Networks (GANs) [27] are an application of ANNs to generative modeling that has found great success. Recent work proved that GANs can generate artificial facial photographs indistinguishable from real ones [43] and realistic deepfakes [100].

The rest of this section presents the fundamentals of GANs.

2.3.1 Basics

The main idea of GANs is to set up a game between two players: a **Generator** and a **Discriminator**. Both of these players are realized by ANNs. The Generator produces fake data samples, and its goal is to fool the discriminator into classifying the fake data as real. The Discriminator examines samples determining if they are real or fake and is trained to distinguish between real and fake data. Analogously, the Generator can be thought of as a counterfeiter producing fake money and the Discriminator as a police agent, discriminating legitimate and fake money.

The GAN game is illustrated in Figure 2.12.

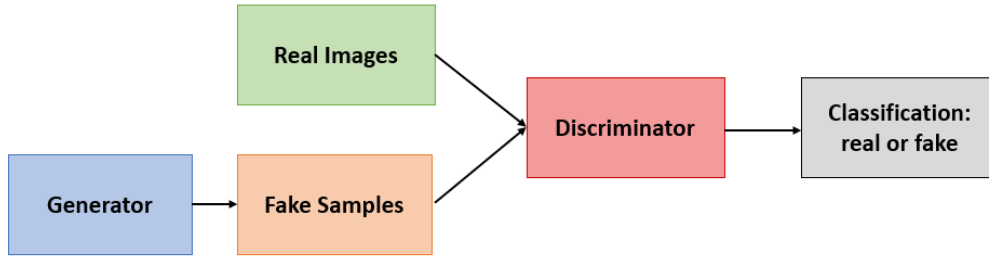


Figure 2.12: Overview of the GAN game between Generator and Discriminator.

The generator G defines a distribution p_g over data space $\mathbf{x}$ by building a mapping from a prior noise distribution $p_z(\mathbf{z})$ to data space as $G(\mathbf{z})$. The discriminator D outputs a single scalar $D(\mathbf{x})$ representing the probability that sample $\mathbf{x}$ came from real data rather than p_g . D is trained to correctly classify between real samples as real and fake samples as fake. Simultaneously, G is trained to maximize the probability of D classifying fake samples as real. This formulation corresponds to a two-player minimax game with the value function $V(G, D)$ presented in equation 2.10:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))] \quad (2.10)$$

Both networks, D and G , are trained simultaneously as presented in Algorithm 1. Ideally, after enough training, the samples generated by G are similar to real ones such that the output of D becomes close to 50% for all samples, real and fake, meaning that D is no better at classifying samples than a random classifier.

Given that D and G have sufficiently representational power, it can be proved that after enough training iterations, G recovers the true data distribution, i.e., $p_g = p_{\text{data}}$ [27]. In other words, from a random vector, $\mathbf{z}$, G can generate samples that follow the same distribution as those contained in the training dataset.

Furthermore, it can be proved that the GAN value function 2.10 leads to minimizing the Jensen–Shannon divergence between the distribution represented by the Generator p_g and the real data distribution p_{data} [27].

Algorithm 1 Training procedure for GANs using gradient descent and the backpropagation algorithm.

```

for number of training iterations do
  1. Train Discriminator
  for  $k$  steps do
    • Sample batch of  $m$  fake samples from generator  $G$ .
    • Sample batch of  $m$  real samples from the training set.
    • Calculate Discriminator's predictions and update its weights to maximize the minimax value function.
  end for
  2. Train Generator
  • Sample batch of  $m$  fake samples from generator  $G$ .
  • Calculate Discriminator's predictions and update the Generator's weights to minimize the minimax value function.
end for

```

2.3.2 Influential GAN variants

Since GANs were introduced in 2014 by Goodfellow *et al.* [27] many GAN variants have been proposed to tackle different issues the original GAN framework did not address, such as high-resolution image synthesis and representation learning. Some influential GAN variants are presented next.

- **Conditional GAN** [60] extends the GAN framework to a conditional setting by making both the Generator and Discriminator conditioned on some extra information $\mathbf{y}$ (e.g., class labels). The conditioning is performed by feeding $\mathbf{y}$ into both the Discriminator and Generator as additional input.
- **Deep Convolutional GAN** [68] demonstrated that a CNN architecture for the Generator and Discriminator results in higher quality images when compared to using ANNs with fully connected layers, as was the case in the original GAN.
- **InfoGAN** [15] proposed an extension to the GAN framework which improved disentanglement of the latent variables $\mathbf{z}$, allowing interpretable and meaningful representations to be learned by the Generator. This is achieved by adding a new loss term to the GAN game value function, leading the Generator to maximize the mutual information between a subset of noise variables $\mathbf{z}$ and generated samples.
- **Super-Resolution GAN** [49] demonstrated that GANs can be used for image upscaling problems, preserving image texture unlike competing methods, and leading to state of the art results. They achieved that by adding a new loss term to the GAN game value function known as the perceptual loss [39].
- **Bidirectional GAN** [21] proposed adding a new network to the GAN game, designated as *Encoder*, that converts samples from data space $\mathbf{x}$ to the Generator's latent variables $\mathbf{z}$. This permits the embedding of real images with latent variables, which allows for semantic editing of the real images by manipulating the values of the latent variables.

- **CycleGAN** [106] showed that GANs can be used in image-to-image translation tasks with unpaired data, i.e., tasks in which the input and output are both images and no correspondence exists between images of the two domains in the training set (e.g., a training set containing images of zebras and horses). This is achieved by using two GANs and adding a new loss terms to the GAN game value function named *cycle consistency loss*, defined as the L1 loss between an image and its transformation after the GAN transforms it domain A to domain B and then back from domain B to domain A.
- **StarGAN** [17] addressed the scalability issues of CycleGAN, extending it to more than two image domains without incurring in a significantly increase of computational power. Only one Generator exists in the StarGAN framework independent of the number of domains of the training data, unlike in CycleGAN, which requires two generators for each possible domain pairing.
- **StackGAN** [103] proposed a novel augmentation technique to Conditional GAN which enables generating images from textual descriptions. The authors decompose the problem in two stages: first, a Generator conditioned on the input text embeddings produces a low quality resolution image which is then fed to another Generator which upsamples the image.
- **Pix2Pix** [37] is a GAN variant tailored for paired image-to-image translation tasks, i.e., tasks in which the input and output are both images and a correspondence exists between them (e.g., semantic segmentation). The authors proposed using a U-Net architecture [71] for the Generator and a novel PatchGAN architecture for the Discriminator. PatchGAN consists of a CNN network that is applied independently to several patches of the input image outputting a scalar value indicating the probability of real/fake; the final Discriminator prediction is given by the arithmetic mean of PatchGAN predictions for all image patches.
- **ProGAN** [41] presented a new training methodology for GANs to generate high resolution images. First the Generator and Discriminator operate at a low resolution and then progressively new layers are added to increase the output dimension. The authors empirically found the methodology to fasten and stabilize training.
- **StyleGAN** [42] introduced a novel architecture for the Generator which favors disentanglement of style in the noise variables and can separate high-level attributes from stochastic effects. One key change that StyleGAN proposes for the Generator is passing the noise variables $\mathbf{z}$ into a noise mapping ANN which outputs a set of weights that are then injected in adaptive instance normalization layers [35] throughout the rest of the network.
- **Self-Attention GAN** [102] proposed using self-attention convolutional layers [91] in GANs to address the problem of long-range dependencies: the Generator is able to create samples using cues from all feature locations and the Discriminator can verify if detailed features in distant portions of the image for consistent.
- **BigGAN** [12] was the first GAN to generate high quality and high resolution images on the ImageNet ILSVRC 2012 dataset [76], comprised of images from 1000 classes. BigGAN demonstrated that big networks for the Generator and Discriminator, with 4x more

parameters than competing approaches, result in very realistic looking samples. Training big networks was a big challenge due to the instability of GAN training but the authors proposed novel techniques to address the issue.

- **RealnessGAN** [96] demonstrated that changing the output of the Discriminator from a single scalar denoting a probability of real/fake to a vector of scalars denoting multiple probabilities significantly improves training speed and stability.

2.3.3 Improving Training

Getting GANs to converge to solutions in which the generated samples resemble real data is notoriously hard [26]. Common problems that plague the training of GANs include:

- **Vanishing Gradients:** the Discriminator becomes significantly better than the Generator and thus is able to distinguish between fake and real samples almost perfectly. This makes the training come to a halt because the gradients of the Discriminator become close to zero, leading to the weights of the Generator no longer being updated [4].
- **Mode Collapse:** the Generator converges to a solution that only captures one mode of the real data distribution, e.g., only generating images of dogs despite the training set containing images of all animals. This usually occurs when the Generator is able to fool the Discriminator in some distribution modes but not on others; this leads to the Generator producing samples of that mode with increasing frequency and less so in others, despite their frequency on the training set [58].
- **Hard to reach Nash equilibrium:** the Discriminator and Generator play a two-player min-max game and are trained simultaneously to find a Nash equilibrium. However, the updates to each player during optimization are performed independently. Updating the weights of the players concurrently cannot guarantee convergence [78].

To improve the training of GANs, namely convergence and training time, several methods have been introduced. Some propose changing the value function of the GAN game, and others focus on high-level training techniques or small architecture changes [20]. The most widely used methods are presented next.

2.3.3.1 Alternative GAN game value functions

Many different loss functions for GANs have been proposed over the years. For a systematic review of those, we refer the reader to the work of Kurach *et al.* [47]. Some loss functions which are currently widely used are:

- **Non-saturating loss** [27] is proposed in the same paper that introduced GANs. The authors argued that the GAN game value function may not provide sufficient gradients for the Generator to learn well during early training. The authors proposed making the Generator maximize $\log D(G(\mathbf{z}))$ instead of minimizing $\log(1 - D(G(\mathbf{z})))$.

- **Least squares loss** [56] suggests using a squared error loss instead of the classic cross-entropy loss employed in the GAN game, making the gradients more stable throughout training. The authors prove that minimizing the least squares loss leads to minimizing the Pearson χ^2 divergence between the distribution represented by the Generator p_g and the real data distribution p_{data} . The least squares loss for the Discriminator and Generator is presented in Equation 2.11.

$$\begin{aligned}\min_D V_{\text{LSGAN}}(D) &= \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [(D(\mathbf{x}) - 1)^2] + \frac{1}{2} \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [(D(G(\mathbf{z})) + 1)^2] \\ \min_G V_{\text{LSGAN}}(G) &= \frac{1}{2} \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [(D(G(\mathbf{z})))^2],\end{aligned}\tag{2.11}$$

- **Wasserstein loss** [5] is a loss based on the Wasserstein distance, also known as the Earth Mover's distance. Informally, this distance corresponds to the energy cost of transforming the mass of one probability distribution to match the shape of another distribution. The Wasserstein loss provides stable gradients throughout training and also offers additional convergence guarantees when the distributions are located in nonoverlapping low dimensional manifolds, as is often the case with real world data [66]. The Wasserstein GAN value function is presented in equation 2.12.

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [D(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [D(G(\mathbf{z}))].\tag{2.12}$$

This loss requires removing the Discriminator's final activation function in order for the Discriminator output to be an unbounded scalar. Furthermore, the Discriminator must represent a 1-Lipschitz continuous function, i.e., the norm of its gradients with respect to the input must not exceed 1. To this end, the original work that introduced the Wasserstein loss suggests clipping the Discriminator's weights to some small interval $[-c, c]$. Gulrajani *et al.* [29] showed that performing gradient penalty instead of weight clipping results in higher quality generated samples and stabler training. To perform the gradient penalty, an additional loss term is added to the GAN value function.

2.3.3.2 Normalization

Normalization refers to a set of techniques that aim to normalize weights or layer outputs to follow a probability distribution or abide by a constraint. Although normalization is not typically employed in GANs, recent work has shown that for complex datasets comprised of many classes, it stabilizes training significantly [12]. Spectral Normalization, the most popular method of normalization for GANs, is presented next.

- **Spectral Normalization** [62] is a simple normalization technique used to bound the spectral norm of the Discriminator's weights, improving its stability and ensuring it is β -Lipschitz continuous, where β is an hyperparameter. Spectral Normalization can thus serve as a replacement method for weight clipping and weight penalty in the Wasserstein loss. Moreover,

the authors empirically found spectral normalization to result in higher quality generated samples when compared to other normalization techniques. Recent work has shown that the Generator also benefits from Spectral Normalization [102].

2.3.3.3 Training Techniques

Five techniques have been proposed by Salimans *et al.* [78] to improve the training of GANs. These techniques are as follows:

- **Feature Matching** consists of adding a new loss term $\|\mathbb{E}_{x \sim p_{\text{data}}} f(x) - \mathbb{E}_{z \sim p_z(z)} f(G(z))\|_2^2$ to the Generator's loss function to make it produce samples that match the activation of real data on some hidden layer $f(x)$ of the Discriminator.
- **Minibatch Discrimination** allows the Discriminator to look at the similarity between multiple data samples before deciding if each sample is real or generated. This helps mitigate the Mode Collapse problem because the Discriminator can infer if the Generator is producing samples of some data mode with unrealistic frequency.
- **Historical Averaging** suggests adding a new loss term $\|\Theta - \frac{1}{t} \sum_{i=1}^t \Theta_i\|^2$, where Θ_i are the weights to both players loss functions in order to stabilize weight updates. This makes the weights updates on timestep i reliant on the values of the weights in several previous timesteps.
- **One-sided label smoothing** proposes changing the labels of real data from 1.0 to 0.9 on the Discriminator's loss function in order to increase its robustness.
- **Virtual Batch Normalization** modifies Batch Normalization to make it normalize the input in relation to a reference batch of examples that are chosen once at the start of training. The main advantage is making Batch Normalization no longer rely on other samples of the same batch, which had the problem of making its outputs inter-dependent leading to non-deterministic behaviour during training.

2.3.4 Evaluating performance

The values for the GAN game value function do not correlate with how realistic the samples created by the Generator are, making it difficult to determine when training should be stopped. Furthermore, this issue makes it hard to benchmark performance of multiple GANs without inspecting the generated samples manually.

Two widely used metrics that quantify the quality of generated samples of image synthesis GANs are the **Inception Score** (IS) [78] and the **Fréchet Inception Distance** (FID) [33]. Both metrics depend on a Inception-v3 ANN [85] trained on the ImageNet dataset [76]. The Inception-v3 is a deep CNN architecture with a good balance between performance and computational cost, making it one of the most popular architectures of the past decade. However, any deep CNN trained on a big dataset could be used instead of Inception-v3. Nevertheless, to allow for an

unbiased comparison of results across different literature works, the Inception-v3 model trained on ImageNet by Szegedy *et al.* [85] has remained the standard choice.

The two metrics, IS and FID, are described next.

2.3.4.1 Inception Score (IS)

The IS evaluates GANs by the quality and diversity of samples produced. It does that by calculating the statistics of the Inception-v3 network outputs when given a batch of generated images. The IS is defined by Equation 2.13, where $p(y|\mathbf{x})$ is the conditional label distribution, $p(y) = \int_{\mathbf{x}} p(y|\mathbf{x})p_g(\mathbf{x})$ is the marginal class distribution, $\mathbf{x}$ is a generated image and $D_{KL}(p||q)$ is the KL-divergence of distributions p and q . High IS values indicate good performance.

$$\text{IS}(G) = \exp \left(\mathbb{E}_{\mathbf{x} \sim p_g} D_{KL}(p(y|\mathbf{x}) || p(y)) \right), \quad (2.13)$$

The IS is usually approximated by calculating the IS for 10 batches of 5.000 generated images and then averaging the scores obtained to get the final value [7]. First an empirical marginal distribution is constructed from samples $\mathbf{x}^{(i)}$, according to Equation 2.14:

$$\hat{p}(y) = \frac{1}{N} \sum_{i=1}^N p(y|\mathbf{x}^{(i)}), \quad (2.14)$$

And then an approximation to the the expected KL-divergence term can be calculated according to Equation 2.15:

$$\text{IS}(G) \approx \exp \left(\frac{1}{N} \sum_{i=1}^N D_{KL}(p(y|\mathbf{x}^{(i)}) || \hat{p}(y)) \right). \quad (2.15)$$

Looking at the IS formula, we can see that it is based on the following two principles:

- **The generated images should contain meaningful objects**, leading to a conditional class distribution $p(y|\mathbf{x})$ with low entropy, as depicted in Figure 2.13a. The Inception network classification should indicate strong confidence that there is a single class present in the image, i.e., only one class probability should have a high value.
- **The generated images should be diverse**, making the marginal class distribution $p(y)$ high entropy, as depicted in Figure 2.13b. The Inception network predictions should resemble a uniform distribution given the entire batch of generated data, i.e., the generated images should belong to many categories.

It has been found that the IS correlates fairly well with human judgment [78]. However, multiple works have highlighted some limitations of the IS:

- The fact that the Inception-v3 network is trained on the ImageNet dataset may lead to misleading results when the IS is applied to GANs which are trained on datasets dissimilar to ImageNet [72, 7].

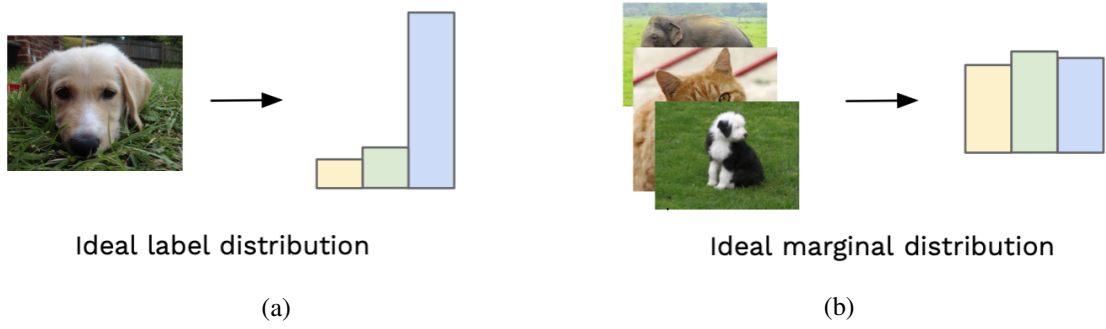


Figure 2.13: The IS is high for GANs which produce images that are individually representative of a single class (a) but collectively, represent a vast number of classes (b). Reproduced from [89].

- IS does not penalize GANs which produce similar intra-class images, i.e., overfitted GANs. A GAN that generates one image for each class, as reported by the Inception-V3 network, attains a high IS [7].
- IS is affected by image resolution, making it difficult to compare the image quality of GANs with different output dimensions [11].

2.3.4.2 Fréchet Inception Distance (FID)

The FID was proposed in order to address the shortcomings of the IS. It is inspired by recent research, which has shown that the activations from hidden layers close to the output layer of trained deep CNNs correlate well with human perception and contain meaningful semantic information about images in a concise representation [39, 104]. The FID is calculated from a hidden layer close to the output layer, namely the last pooling layer of Inception-v3, in contrast with the IS, which is computed from the output class predictions. Moreover, the FID also uses real data in order to quantify the performance of GANs. More precisely, it measures the similarity of the statistics of the activations of the last pooling layer between real and generated images.

To calculate the FID, the mean and covariance of the activations of the last pooling layer of Inception-v3 are estimated from fake and real data, obtaining μ_r , μ_g , Σ_r , Σ_g . Then the FID is computed according to Equation 2.16:

$$\text{FID}(G) = \|\mu_r - \mu_g\|_2^2 + \text{Tr}\left(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{\frac{1}{2}}\right), \quad (2.16)$$

The FID summarizes the pooling layer's activations as multivariate Gaussian distributions and measures the Fréchet distance between the real and fake data distributions. As such, low values for the FID indicate better GAN performance than high values, unlike the case with the IS.

Overall, the FID is a better performance metric when compared to IS for multiple reasons:

- The FID correlates better with human judgment and is more robust to noise in the images [33].
- The FID penalizes GANs which do not produce diverse intraclass images [11].

- The FID presents less misleading results when applied to GANs trained on datasets dissimilar to ImageNet. This is because the last pooling layer contains less class-specific information than the output layer [11].

Nevertheless, the IS is usually presented along with the FID in contemporary literature work in order to provide a common benchmark for GANs proposed when the FID was not yet commonly used.

2.3.5 Summary

Generative Adversarial Networks (GANs) are an application of ANNs to generative modeling. Generative modeling consists of learning a prior distribution $P(X)$ given samples drawn from it. In other words, it consists of learning to produce new samples that resemble the ones in a dataset. GANs learn a mapping from a noise input to real data space. They are comprised of a **Generator** network that produces samples following the same data distribution as those comprising the training dataset and a **Discriminator** network that distinguishes between real and fake samples, providing feedback to the Generator in the form of error gradients. **The whole system is trained end-to-end using gradient descent and backpropagation.**

The competition between the Generator and Discriminator can be formulated as a **two player minimax game**. A single optimal solution exists, the Nash equilibrium, in which the Generator captures the real data distribution perfectly and the Discriminator can no longer discern between real and generated images. However, GANs often fail to converge due to problems such as **vanishing gradients** (the Discriminator gets significantly better than the Generator and its gradients become very small, leading to a stagnation in the Generator's weight updates) and **mode collapse** (the Generator produces samples that are not diverse with relation to the real distribution modes). Furthermore, there are no convergence guarantees because the updates to each player's weights are done concurrently. Several methods to address these issues have been proposed. These include alternative loss functions, normalization, and high-level training techniques.

Evaluating GANs is difficult because the loss functions they optimize do not directly correlate with the quality and diversity of generated samples. The **Inception Score** (IS) and the **Fréchet Inception Distance** (FID) are two widely used performance metrics for image synthesis GANs. Both metrics rely on a Inception-v3 network trained on ImageNet. The IS is calculated from the output layer of Inception-v3, attributing high values to GANs that produce a diverse set of images that distinctively contain meaningful objects. The FID is calculated from the last pooling layer of Inception-v3 and compares the similarity of statistics of the activations of this layer for real and generated images. FID low values correlate with good performance. One big difference separates the two metrics: the IS only uses generated images to quantify performance, whilst the FID uses both real and generated images. Overall, the FID is a better metric because it addresses many issues of the IS, presenting more meaningful scores that correlate better with the quality and diversity of generated samples.

The next chapter gives a detailed literature review on AI for fashion.

Chapter 3

AI for Fashion: Literature Review

AI is one of the main driving forces of the ongoing scientific revolution. ANNs have proved that computers can match human intelligence in many tasks, but only recently have they excelled in creating subjective content such as art. Despite AI comprising of many fields, it is ANNs that have seen tremendous success in fashion applications. As such, this chapter focuses on presenting literature works that involve ANNs applied to fashion problems.

Each section of this chapter explores an application of AI to fashion and describes corresponding influential works. The last two sections contain a description of popular fashion datasets and a summary of this chapter's content.

3.1 Pose Transfer

Pose Transfer, as depicted in Figure 3.1, is the task of synthesizing an image of a person in a new pose based on (i) a reference image of the person and (ii) the target pose. Importantly, the synthesis process must preserve the clothing and identity of the person. Pose Transfer can be employed in online fashion stores, allowing clients to see how the clothing item looks under different perspectives through the synthesis of novel images of fashion models in configurable poses. Use cases of Pose Transfer outside the fashion industry include creating training data of people in rare poses to use in ML algorithms or creating new images of a person from an existing photograph to serve new purposes.

Pose Transfer is a difficult task for multiple reasons. First, the human body is nonrigid, which hinders the process of transforming misaligned body parts into new positions. Second, clothing attributes such as texture and shape are hard to preserve during image generation. Third, creating training objectives that are suitable for Pose Transfer is difficult because a small number of training datasets exists for this task [55].

Jia *et al.* [55] state that developing an end-to-end framework for Pose Transfer is difficult because such framework has to generate correct poses and detailed appearance in a single step.

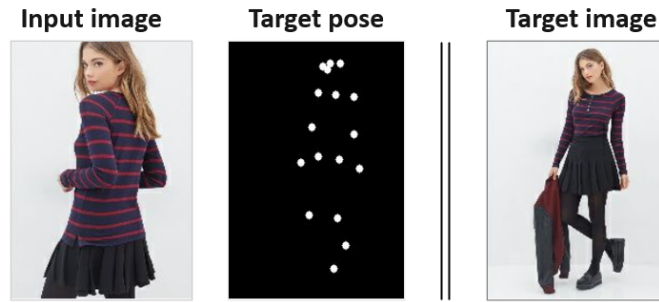


Figure 3.1: **Pose Transfer** consists of synthesising a target image that a) contains the same person and clothing apparel present in a input image and b) the body of the person is in a target pose. Images reproduced from the DeepFashion dataset [53].

As such, the authors introduced a supervised learning framework that splits the problem into two stages. The pose integration stage takes in a reference image of the person and the pose as inputs and generates a blurred image of the person in the new pose. The image refinement stages takes in the output image of the pose integration stage and makes it more realistic through the use of adversarial training.

Motivated by the need to tackle the problem of Pose Transfer with unpaired data, Pumarola *et al.* [67] devised a framework that learns in a fully unsupervised manner and is trained similarly to CycleGAN[106]. First, the framework takes a reference image of the person and the desired pose and outputs an image of the person in the desired pose. It then takes in this output and the initial pose and generates an image of the person in the initial pose. The reasoning being that the final output should resemble the reference image of the person. Three loss functions are simultaneously optimized: adversarial loss, a pose loss, and a perceptual loss.

3.2 Garment Translation

Garment Translation is the task of synthesizing an image of a person wearing a target clothing item based on a reference image of that person and an image containing the clothing item. Importantly, the person's identity and pose need to be preserved, and the clothing item must retain its texture and shape. The target clothing image can be of two types: a product image or an image of a person wearing a clothing item, thus leading to two different problems of Garment Translation:

- **Garment Try-on:** the target clothing is present in a *product image*, not being worn by any person, as depicted on Figure 3.2a.
- **Garment Transfer:** the target clothing item is present in a *image of a person wearing the clothing item*, as depicted on Figure 3.2b.

Garment Transfer is more complicated than Garment Try-on due to the increased difficulty of extracting the clothing item from an image of a person wearing them *vs* extracting the clothing item from a product image. Particularly, in Garment Transfer, the clothing item might be partially



Figure 3.2: In **Garment Try-on**, the clothing item is extracted from a product image (a) whilst in Garment Transfer, the clothing item is extracted from an image of a person wearing them (b). Clothing images reproduced [22].

occluded and not perfectly oriented towards the camera, while in Garment Try-on, none of these issues are present.

The two problems are explored in the next subsections.

3.2.1 Garment Try-on

Garment Try-on, as depicted in Figure 3.3, works by synthesizing an image of a person wearing a specific fashion item given (i) a reference image of that person and (ii) a product image portraying the clothing item. Garment Try-on can be used to allow fashion customers to try on new clothes before buying them and without leaving their home. Moreover, trying on new clothes virtually can be significantly less time consuming and more convenient than doing it physically.



Figure 3.3: **Garment Try-on** consists of synthesising a target image that a) contains the same person of a reference in the same pose and b) the person is wearing the clothing item depicted in a different image. Clothing images reproduced from [90].

Han *et al.* [30] proposed an unsupervised learning framework that performs Garment Try-on in three stages. The first stage, which is the person representation stage, takes in the reference image and produces three outputs: a pose map, a body shape mask, and an image with the hair and face of the person. The second stage, which is the coarse image generation stage, takes in

the outputs of the previous stage and the product image and then produces a clothing mask and a blurred image of the person wearing the item. The third stage, which is the refinement stage, takes in the blurred image, the product image, the clothing mask and produces the final result, an image of the person wearing the clothing item depicted in the product image. During training, the framework requires the usage of off-the-shelf software, namely, a Pose Estimator and a Human Parser.

Wang *et al.* [90] noted that the proposed framework of Han *et al.* struggles with preserving key-characteristics of the clothing item, such as the text, texture, and logos. Furthermore, the framework fails to produce decent results when there is a large spatial displacement between the pose of the person and the clothing item. As such, they introduced a new framework that addresses these problems explicitly through the use of a geometric matching module to improve the alignment of the clothing items and the body of the person. Later, Minar *et al.* [59] improved this framework to better tackle rich-textured and long-sleeved clothing.

Dong *et al.* [22] introduced a method for Virtual Try-On that can not only transfer clothes to a person but also synthesize an image of the person in a different pose. Their proposed framework presents a solution for both Virtual Try-On and Pose Transfer; it consists of three stages. Stage I constructs a segmentation map of the body in the reference image. Stage II removes the original clothing from the reference image, warps the target clothing, and, based on the segmentation map, concatenates it with the reference image, producing a coarse approximation of the final result. And lastly, Stage III transforms the coarse approximation to be more realistic.

3.2.2 Garment Transfer

Garment Transfer, as depicted in Figure 3.4, works by synthesizing an image of a person wearing a fashion item that is being worn by a different person given (i) a reference image of the person, and (ii) an image of the other person wearing the fashion item. Garment Transfer can be used to allow ordinary people to see themselves wearing the apparel of celebrities and fashion icons without doing any prohibitively expensive shopping.

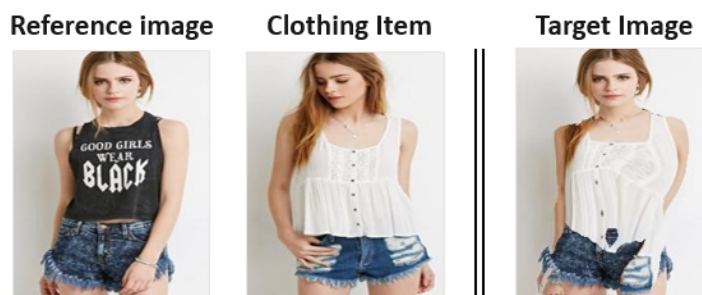


Figure 3.4: **Garment Transfer** consists of synthesising a target image that a) contains the same person in the same pose of a reference image and b) the person is wearing a clothing item depicted in a different image. Clothing images reproduced from [95].

Raj *et al.* [69] argue that Garment Transfer is a challenging task that requires a) disentangling the features of the clothing from the body pose and shape, and b) realistic synthesis of the garment texture on the new body. As such, they introduce a framework that tackles these subproblems with two task-specific stages. The first stage, which is the warping stage, takes in the reference image and the clothing item image and generates a clothing segmentation map consistent with the target pose. The second stage, which is the texturing stage, takes in the previous stage's output and the clothing item image and produces the detailed target image, consistent with the target pose and with the clothing texture. Furthermore, the authors introduced a weakly supervised training procedure based on data augmentation to permit training in the absence of paired training data for the same clothing items being worn by people in different poses.

3.3 Garment Design

Garment Design is the task of synthesizing an image of a novel garment that looks realistic and aesthetic appealing. AI applications for Garment Design can be an important part of the workflow of fashion designers, allowing them to start working on already existing clothing items and fine-tuning their details to consumer demands in fashion markets. Garment Design can be done unconditionally or based on some criteria, leading to distinct problems:

- **Unconditional Garment Design:** consists of generating an image of a garment without any prior defined criteria.
- **Garment Design Conditioned on Text:** consists of generating an image of a garment guided by a textual description of the desired garment.
- **Garment Design Conditioned on Existing Clothing:** consists of generating an image of a garment conditioned on an image of an already existing garment.

These three problems are explored in the next subsections.

3.3.1 Unconditional Garment Design

Unconditional Garment Design, as depicted in Figure 3.5, consists of generating an image of a novel garment in which the generation process is not subject to any defined condition or criteria. Thus, AI methods for this task can generate any type of clothing item. However, this may pose a hindrance in situations where a particular type of fashion item is desired, requiring the generation of many garment images until an appropriate one is found.

Rostamzadeh *et al.* [73] used ProGAN [41] to synthesize high-resolution 1024x1024 images of clothing items being worn by fashion models. The ProGAN framework consists of adding layers to the Generator and Discriminator of GANs in order to gradually produce images of increasing dimensions. The images presented by the authors using this method exhibit global coherence, span a variety of poses and contain a diverse set of garments of different types, colors, and designs. Similarly, Yildirim *et al.* [98] applied StyleGAN [42], a framework that improves on ProGAN, to tackle the same issue.



Figure 3.5: **Unconditional Garment Design** consists in synthesizing images of garments in an unconditional manner, i.e., not guided by any criteria. Images reproduced from [73].

3.3.2 Garment Design Conditioned on Text

Garment Design Conditioned on Text, as depicted in Figure 3.6, consists of generating an image of a novel garment given a textual description of the desired garment. This AI application allows fashion designers to quickly come up with a prototypical clothing item based on a textual description of the item they wish to design.

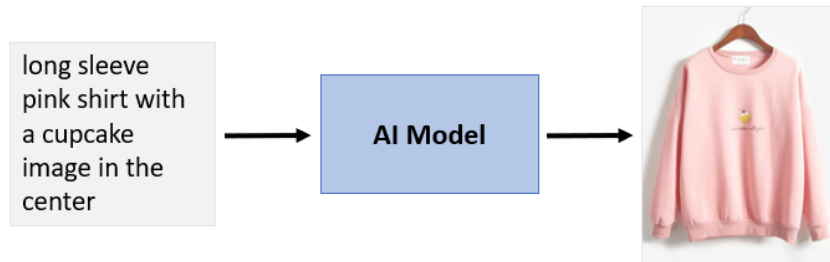


Figure 3.6: **Garment Design Conditioned on Text** consists in synthesizing images of garments given a textual description of the garments. Garment image reproduced from the DeepFashion dataset [53]

Rostamzadeh *et al.* [73] applied StackGAN [103] to generate images of fashion models wearing a particular garment given a only textual description of the garment. StackGAN is comprised of two stages: the first synthesizes a blurred image of the final result and the second stage refines the results of the first to make them look realistic. More recently, Ak *et al.* [3] introduced a framework for image synthesis conditioned on text with attention modules for the text parsing and applied it to the same task, generating fashion models wearing a described garment.

3.3.3 Garment Design Conditioned on Existing Clothing

Garment Design Conditioned on Existing Clothing consists of generating an image of a garment conditioned on an image of an already existing garment. Based on the desired result, two distinct approaches exist:

- **Compatible Garment Design:** Liu *et al.* [52] proposed a framework that generates an image of a garment that complements a garment whose image is given as input, as depicted

in Figure 3.7. In other words, the proposed framework generates a new fashion item that, when combined with the fashion item given as input, creates an outfit that is fashionably compatible, and aesthetically pleasing.

- **Garment Redesign:** Singh *et al.* [82] introduced a framework that, given an image of a garment, generates an image of a different garment having a similar design, as depicted in Figure 3.8. For fashion brands, this permits generating new garments similar to others in high demand by consumers, leading to the design of clothing aligned with ongoing social trends.

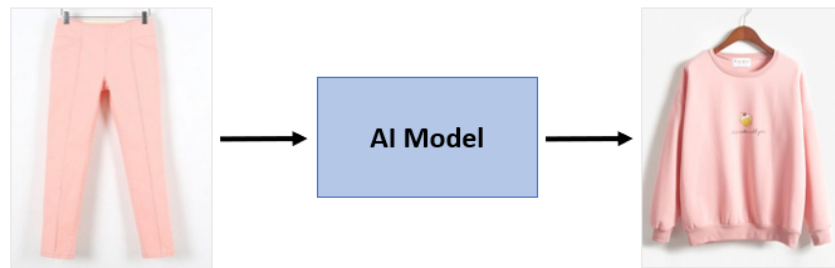


Figure 3.7: **Compatible Garment Design** consists of synthesizing images of a garment that is fashionably compatible with the garment portrayed in an image given as input. Garment images reproduced from the DeepFashion dataset [53]

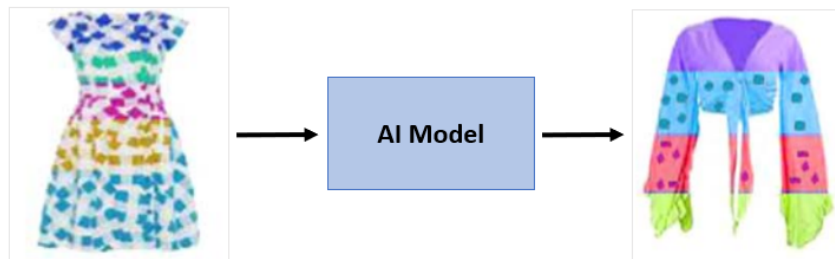


Figure 3.8: **Garment Redesign** consists of synthesizing an image of a garment that is of a different type but has a similar design as the one given as input. Garment images reproduced from [82].

3.4 Garment Retrieval

Garment Retrieval, as depicted in Figure 3.9, is the task of synthesizing an image of a garment being worn by a person in a reference image. The synthesis process must preserve the garment shape and design. Moreover, the synthesized image must have an empty or white background, and the garment must be oriented towards the camera and fully visible. Use cases of Garment Retrieval include retrieving the clothes worn by celebrities in photographs or extracting the clothing of fashion models without taking new photographs.

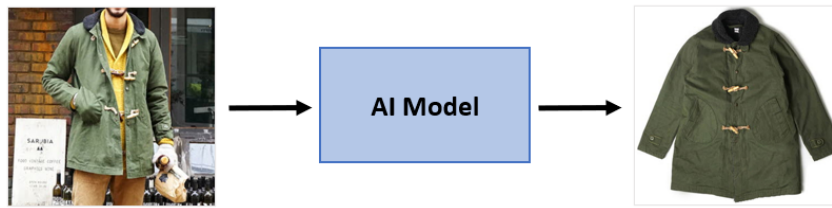


Figure 3.9: **Garment Retrieval** consists of synthesising the image of a garment that a person in a reference image is wearing. Garment Images reproduced from [99].

Yoo *et al.* [99] proposed a Garment Retrieval framework comprised of a Generator, a real/fake Discriminator, and a domain Discriminator. The Generator takes in the reference image of the person wearing the garment and outputs an image of the garment. The real/fake Discriminator distinguishes between real garment images and images produced by the Generator. The domain Discriminator takes in pairs of images of people wearing garments and the corresponding garments and outputs the probability of the image pair being associated, i.e., the probability of the garment image depicting the garment being dressed by the person. The Generator is trained to fool both discriminators.

Similarly, Zhang *et al.* [101] devised a framework composed of a Generator, a Classifier, and a domain Discriminator. The Generator and Domain Discriminator behave the same as in the framework proposed by Yoo *et al.*. However, the proposed framework by Zhang *et al.* replaces the real/fake Discriminator with a Classifier. The Classifier outputs category predictions (e.g., coat, dress) for the garment images. Hence, the Generator must simultaneously a) produce real-looking images that fool the domain discriminator and b) generate garment images that preserve the category of the garments on the reference images.

3.5 Garment Attribute Manipulation

Garment Attribute Manipulation, as depicted in Figure 3.10, consists of altering the attributes of a garment present in a reference image. In other words, a new image of the garment is synthesized with the desired attributes changed. Garment Attribute Manipulation allows editing existing garment images in a faster way than using Computer-Aided Design. Furthermore, it can extend Garment Synthesis methods to achieve better clothing diversity.

Zhu *et al.* [107] proposed a method that alters a garment a person is wearing based on a reference image of the person wearing the garment and a textual description of the desired garment. The framework is comprised of a network that encodes the text into a vector representation, two Generators, and two Discriminators. The first Generator creates a segmentation map and a design encoding of the reference image. The second Generator takes in the outputs of the first Generator as well as the encoded text and outputs the target image. Each Generator is paired with a Discriminator.

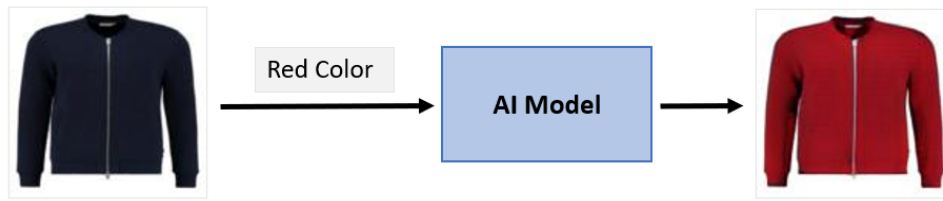


Figure 3.10: **Garment Attribute Manipulation** consists of synthesising a new image of an existing garment with some attributes altered. Garment Images reproduced from [1].

Ak *et al.* [1] introduced a framework that alters the garment attributes using an attention mask. The proposed framework is comprised of a Generator, a global Discriminator, and a local Discriminator. The Generator generates an image and an attention mask. The target image is obtained through a blending operation of the reference image, the generated image, and the attention mask. The global Discriminator looks at the full target image, while the local Discriminator only sees the image region where the attribute manipulation was performed (according to the attention mask).

3.6 Datasets

The literature works described in this chapter use deep ANNs, which require datasets of significant size during training. However, creating fashion datasets is a difficult process because most fashion images present on the internet are copyrighted or contain personal data, preventing the images from being used due to legal reasons. Nevertheless, over the last decade, some fashion datasets have been introduced and made publicly available. Some widely used fashion datasets include:

- **Fashion-MNIST** [97] is a simple dataset comprising of 28x28 grayscale images of fashion items from 10 categories. The dataset is based on the product catalog of the online fashion store Zalando¹. The images corresponded to small thumbnails of photographs taken by photographers of fashion products.
- **DARN** [34] is a dataset containing around 327,000 garment images collected from online stores and 91,000 photographs taken by consumers of those garments. The dataset includes a total of 179 different clothing categories and attributes. The images only depict upper clothing and their resolution is on average of 800x500.
- **DeepFashion** [53] contains over 800,000 richly annotated images from 50 fine-grained categories and over a thousand possible attributes. The images are taken from a variety of scenarios including fashion stores, street snapshots and consumer photographs.
- **DeepFashion2** [24] contains around 491,000 clothing images from fashion stores and consumers. However, unlike DeepFashion, each image depicts several garments being worn by a single person. Each of those garments is labeled with a bounding box, mask, dense

¹<http://www.zalando.com>

landmarks, and a category. Furthermore, all fashion store images in the dataset are paired with the corresponding consumer photographs.

- **DeepFashion-Synthesis** [107] is a subset of the DeepFashion dataset containing around 90,000 images which were manually captioned. The captions are brief and the total vocabulary size is around 500 words. In each image, the person is facing towards the camera and the background of the image is not severely cluttered. This dataset only depicts upper clothing images.
- **Fashion-Gen** [73] contains over 293,000 high-definition (1360x1360) fashion photographs taken of fashion models on a white background. Each image is paired with a description written by a professional stylist. The descriptions are extensive and the total vocabulary size is around 6800 words. The dataset contains clothing items from 48 categories, but no attributes are labelled.

Table 3.1 contains a comparison of the described datasets.

3.7 Summary

AI has come a long way in the last decade. ANNs, a type of ML algorithm, have seen tremendous success in a vast number of fields, including fashion. This chapter explored some applications of AI to fashion, namely:

- **Pose Transfer:** the task of synthesizing an image *of a person in a different pose*.
- **Garment Translation:** the task of synthesizing an image *of a person wearing a different garment*.
- **Garment Design:** the task of synthesizing an image *of a novel garment*.
- **Garment Retrieval:** the task of synthesizing an image *of a garment based on a reference image of a person wearing it*.
- **Garment Attribute Manipulation:** the task of synthesizing an image *of a garment with some attributes altered (e.g., color) based on a reference image of the garment*.

However, these applications have only been possible with the existence of massive datasets of fashion images, such as the DeepFashion dataset.

The next chapter describes the methodology utilized in conducting the present research work.

Dataset	Year	Images	Categories	Observations
Fashion-MNIST [97]	2017	70,000	10	28x28 greyscale images of garments facing the camera.
DARN [34]	2015	327,000	20	Photographs upper garments being worn by fashion models and consumers.
DeepFashion [53]	2015	800,000	50	Photographs of fashion models and consumers wearing garments in diverse scenarios. Contains pairs of images of the same garment being worn by different people.
DeepFashion2 [24]	2019	491,000	13	Photographs of people wearing garments in diverse scenarios. Each garment in each photograph is labelled. Contains pairs of images of the same garment being worn by different people.
DeepFashion-Synthesis [107]	2017	90,000	19	Photographs of people wearing upper garments facing the camera and with little background cluttering. Each photograph is only accompanied by a textual description of the garment.
Fashion-Gen [73]	2018	293,000	48	Photographs of people wearing garments on white backgrounds. Each photograph is only accompanied by a textual description of the garment.

Table 3.1: Comparison of some widely used fashion datasets.

Chapter 4

Methodology

This chapter explains the general methodology that guided the research work. The general research process of this thesis is also outlined and articulated. The chapter also briefly discusses the proposed solution of the research. The methodology of the proposed ANN-based pipeline that is composed of the Unconditional Garment Design (Section 4.3) and Garment Try-On (Section 4.4) components is presented in detail.

4.1 Research Process

The research process was divided into four stages as depicted in Figure 4.1.

Firstly, the **literature review stage** focused on analyzing applications of AI to fashion in order to identify their contributions and limitations. Problems such as pose transfer, garment translation, garment design, garment retrieval and garment attribute manipulate were discussed in detail. Furthermore, studies and works addressing these problems were summarized. Furthermore, popular fashion datasets were studied and compared according to their size, image resolution, and image quality.

Secondly, in the **problem analysis stage**, the research problem and research gap were identified. It was found that no existing research work has focused simultaneously on generating images of garments and showing the garments on fashion models. Nevertheless, a framework capable of performing both of these tasks can help fashion designers design clothing faster and thus improve the time to market of new fashion products. In this stage, a problem statement was formulated, along with research questions and objectives.

Thirdly, the **research approach stage** focused on establishing a proposed solution to solve the research problem and fulfill the research objectives. An ANN pipeline comprised of two components was composed. The first component tackles Unconditional Garment Design, producing images of garments not subject to any user-defined criteria. The second component deals with Garment Try-On; it takes as input the images of garments generated by the unconditional garment

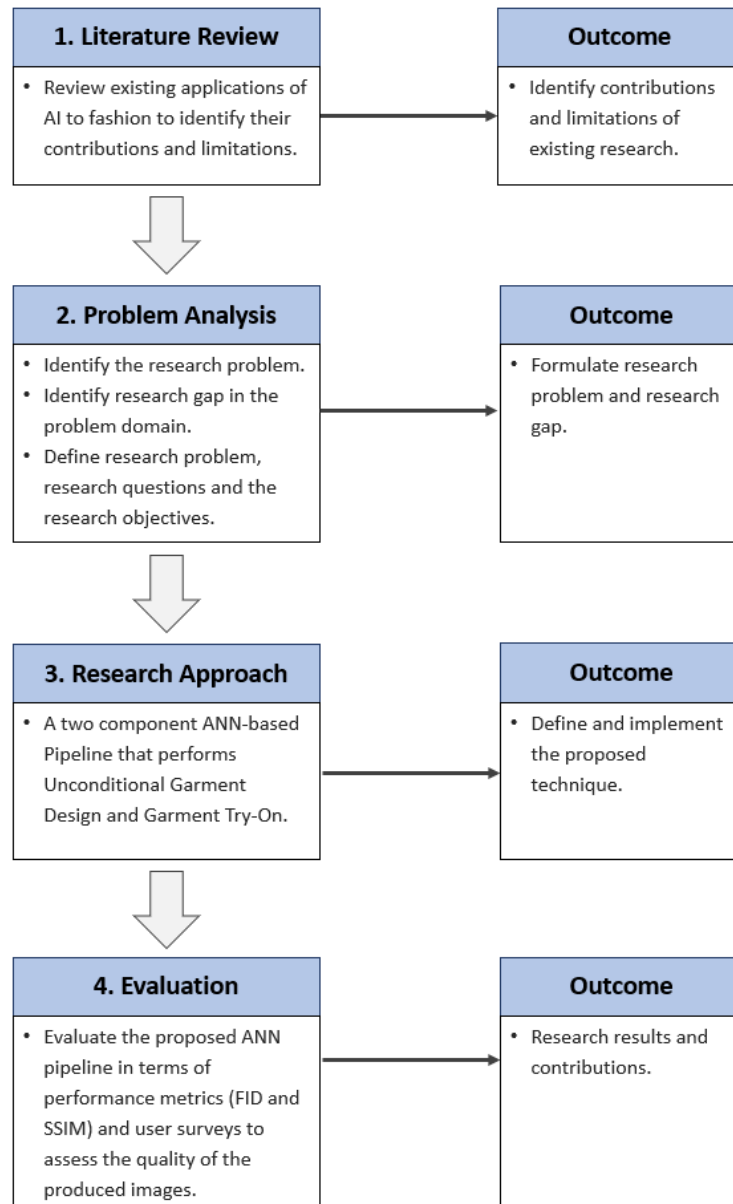


Figure 4.1: Overview of the research process conducted.

design component and dresses them on fashion models. An overview of the ANN pipeline is presented in Section 4.2 while the components are described in Sections 4.3 and 4.4.

Lastly, the **evaluation stage** focuses on measuring the performance of the proposed pipeline. Performance metrics (FID and SSIM) were collected from the components during and after training. Furthermore, user surveys were conducted with both real images and images generated by the proposed pipeline. Questions regarding the realness of the images were asked to volunteers in order to assess the applicability of the proposed pipeline to fashion design.

4.2 Proposed Pipeline

This dissertation proposes and implements an ANN-based pipeline comprised of two components, each addressing a specific task:

1. **Unconditional Garment Design:** generating an image of a garment without any prior defined criteria (p. 35).
2. **Garment Try-On:** generating an image of a person wearing a clothing item based on a reference image of that person and an image containing the clothing item only (p. 33).

Sequentially, the proposed ANN-based pipeline generates images of clothing garments and then displays them on fashion models. Figure 4.2 illustrates this process.

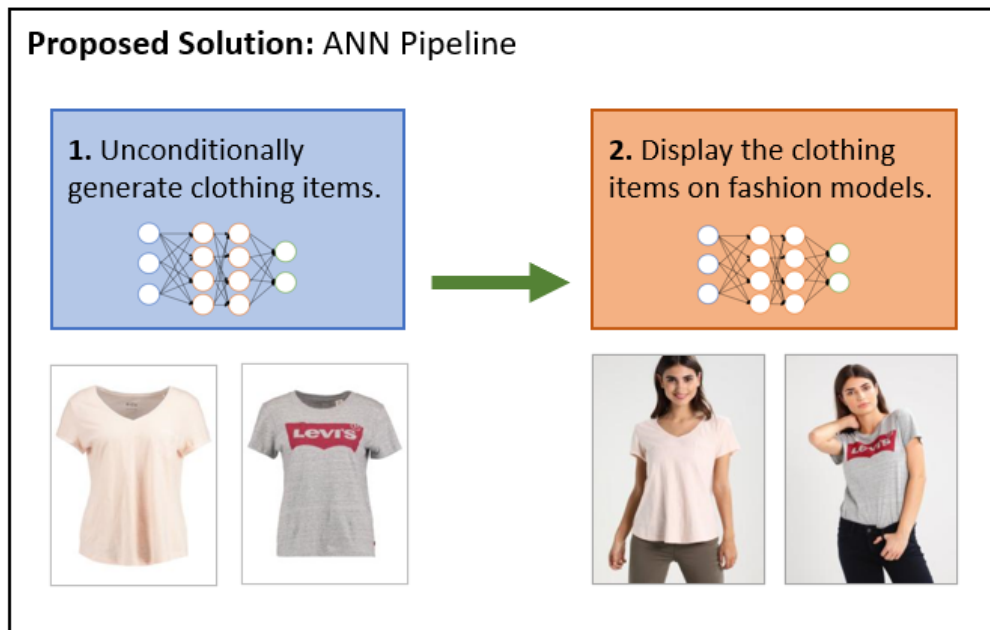


Figure 4.2: Overview of the Proposed Solution, a ANN pipeline that performs both Unconditional Garment Design and Garment Try-On. Clothing images reproduced from [59].

The next two sections describe the methods for Unconditional Garment Design and Garment Try-On which form the proposed ANN pipeline. To fully understand those methods, the reader is advised to read the ANN (p. 14) and GAN (p. 21) sections of the Fundamentals chapter (p. 7).

4.3 Unconditional Garment Design

For the Unconditional Garment Design stage of the proposed pipeline, an ANN architecture based on the StyleGAN2 Generative Adversarial Network framework [43] was designed and implemented. The architecture is named XGAN due to this dissertation being the result of a collab-

oration with XNFY Lab¹. The XGAN architecture is based on StyleGAN2, the state-of-the-art framework for unconditional image generation as of the time of writing [10].

The XGAN architecture implements key architecture details from StyleGAN2 and the wavelet techniques for image synthesis described in [23] that have been shown to halve training speed. Furthermore, the data augmentation method introduced by [105] was implemented to improve image quality on small datasets.

The rest of this section presents XGAN in detail. Subsections 4.3.1 and 4.3.2 describe the *Generator* and *Discriminator* architectures, respectively. Subsection 4.3.3 details the regularization methods employed during training. Subsection 4.3.4 presents techniques employed during sampling that lead to higher quality images. Subsection 4.3.5 introduces the loss functions used during training.

4.3.1 Generator Architecture

The XGAN *Generator* is comprised of two different networks that are trained in an alternating fashion: a **Mapping Network** and a **Synthesis Network**. The architectures of these networks are described next.

4.3.1.1 Mapping Network

The Mapping Network is a simple 8-layer Multilayer Perceptron in which all layers comprised of fully connected neurons have the same input and output dimension (512). It maps the input noise $\mathbf{z}$ to a style vector $\mathbf{w}$ that is then fed as input to the Synthesis Network. The StyleGAN2 authors empirically found that applying a sequence of affine transformations on the input noise $\mathbf{z}$ (by passing it through the Mapping Network) results in features $\mathbf{w}$ that are less entangled, i.e., less correlated. This, in turn, allows for smoother interpolation between fake images and finer control over the synthesis process. Moreover, having a mapping network has been shown to improve image quality [42]. Before the noise is used as input to the Mapping Network, it is normalized using Pixel Normalization. This type of normalization preserves the direction of the input vector but makes its norm equal to 1. Figure 4.3 illustrates the architecture of the Mapping Network. Each fully connected layer is followed by a Leaky ReLU activation function.

¹<https://xnfy.com/>

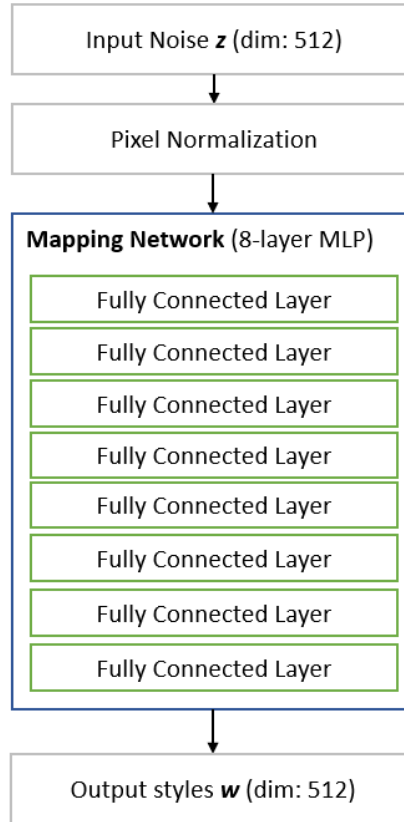


Figure 4.3: The Mapping Network is an 8-layer Multilayer Perceptron that maps input noise $\mathbf{z}$ to styles $\mathbf{w}$. Those styles are then used as input to the Synthesis Network.

4.3.1.2 Synthesis Network

The Synthesis Network is comprised of several building blocks named style blocks. Each style block performs convolution operations on its input. More precisely, the input to each style block is the i) input the output of the previous style block (3-dimensional vector), ii) the style vector $\mathbf{w}$ (1-dimensional) and iii) a noise vector $\mathbf{n}$ (1-dimensional; not to be confused with the input latent vector $\mathbf{z}$ of the Mapping Network). A learnable constant vector is used as the input for the first style block instead of the output of another style block. Furthermore, each style block acts as a feature map upsampler, doubling the resolution (width and height) of the input feature maps passed from the previous style block (an exception is the first style block lacking upsampling). Thus, the resolution of the data is progressively increased until matching the desired output dimension. After the final style block, the data is converted from the wavelet domain to RGB by applying the inverse haar wavelet transform. An high-level overview of the Synthesis Network architecture for 256x256 output image resolution is presented in Figure 4.4.

Each style block receives and produces two 3-dimensional vectors: $\mathbf{x}$ and *wavelets*. Each of those inputs is modified and then passed as input to the next style block. $\mathbf{x}$ contains "hidden" information that needs to be passed between style blocks to guide image synthesis (e.g., global

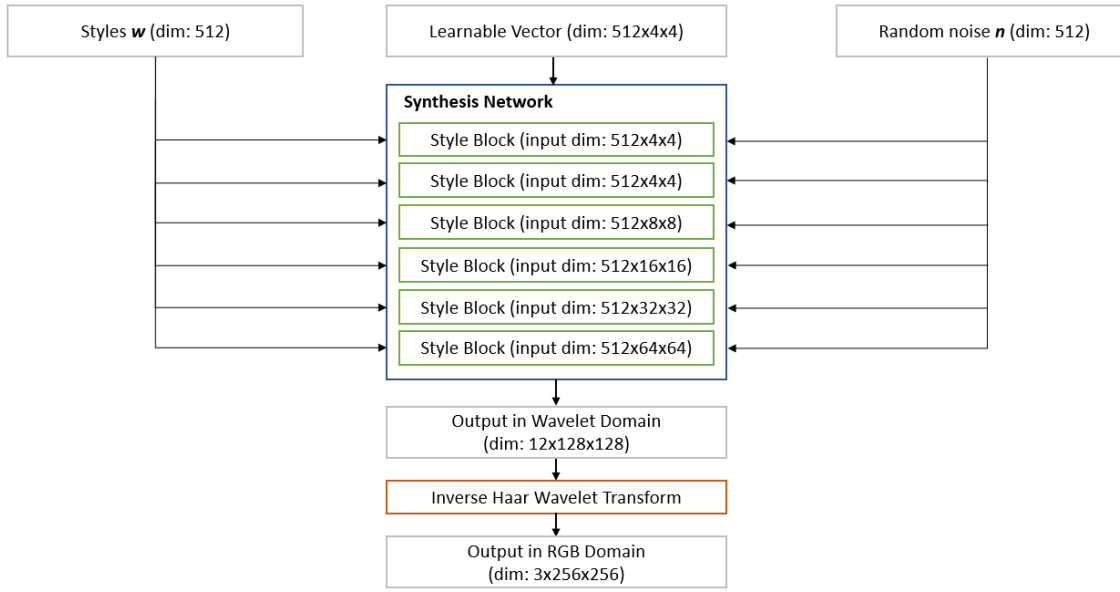


Figure 4.4: Overview of the Synthesis Network for a 256x256 output resolution Generator. The Synthesis Network is comprised of style blocks that are simultaneously optimized during trained.

structure information for the image). In contrast, *wavelets* contains a crude representation of the final image in a lower resolution and thus directly controls the final output of the Synthesis Network. Input x is modified by styled convolutions, affine transformations and leaky Leaky ReLU activation functions and then summed in a residual fashion [32] to *wavelets*. The x output of the final style block is discarded, and only *wavelets* is used in the inverse wavelet transform to produce the output image of the *Generator*.

At the core of the style blocks are styled convolution operations introduced by [43]. Styled convolutions perform 2D convolutions over their inputs based on weights that are calculated using the styles vector w and learnable parameters. Input x is passed through two styled convolutions (with 3x3 kernel, stride of 1, padding of 1). After each of these styled convolutions, an affine transformation of the input noise n is summed to the data, and then the Leaky ReLU activation function is applied. This allows the network to learn local pixel-level noisy details (e.g., small variations of strands of hair), leading to improved image diversity. After passing x through this sequence of operations, it serves as both the output to the style block and also as input to the ToWavelets block along with the style block inputs (*wavelets* and w). The output of the ToWavelets block is also an output of the style block. Figure 4.5 contains a visual depiction of the style block and the described operations. Figure 4.6 illustrates the ToWavelets block.

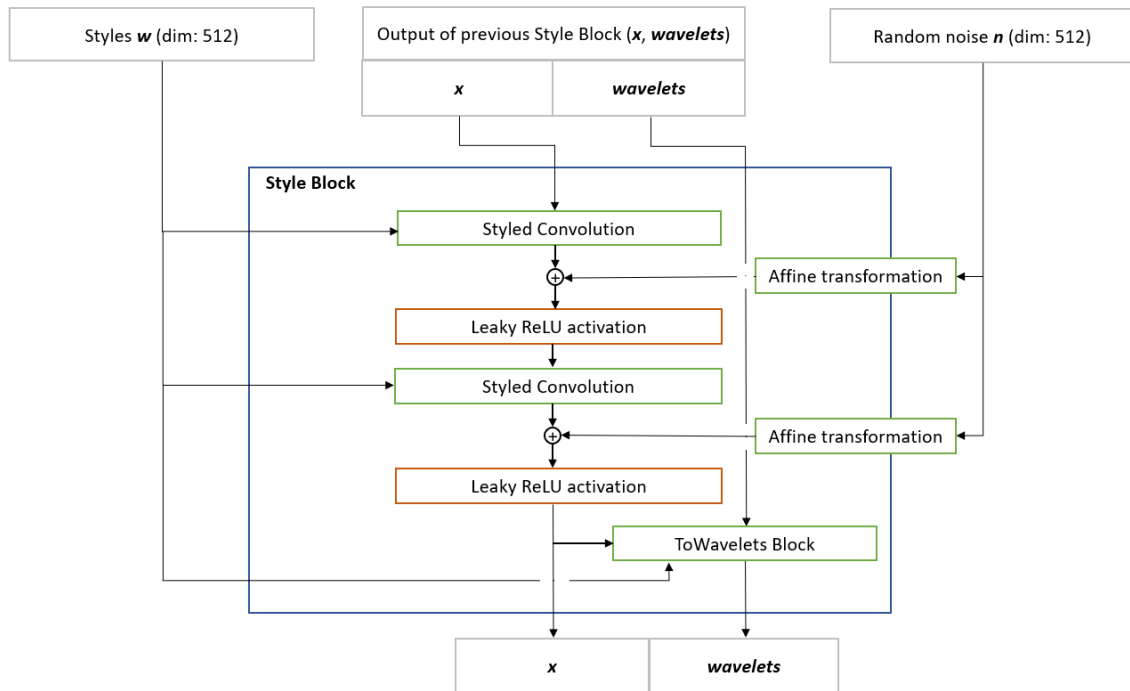


Figure 4.5: Overview of the Style Block of the Synthesis Network of the XGAN Generator.

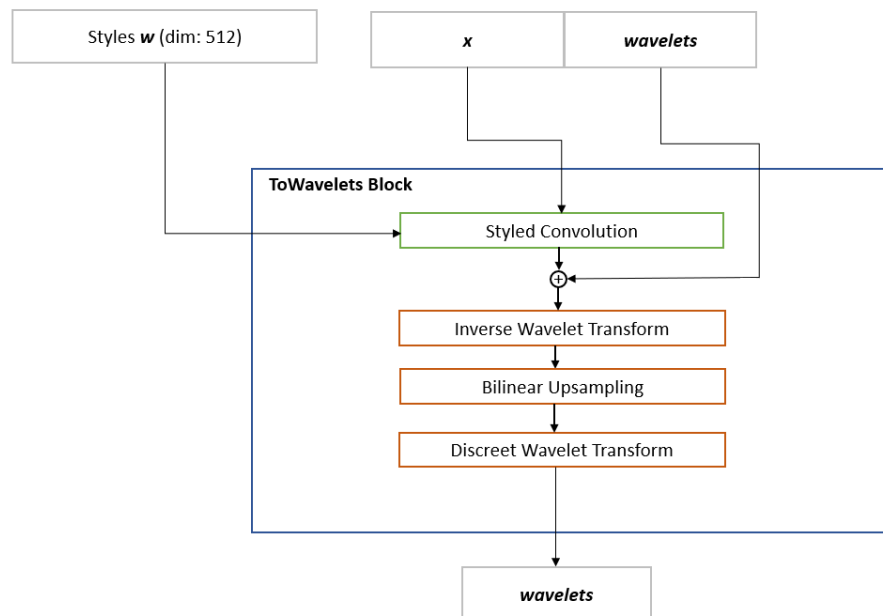


Figure 4.6: Overview of the ToWavelets component of the Style Block of the XGAN Generator.

4.3.2 Discriminator Architecture

Similar to the *Generator*, the *Discriminator* is also comprised of several blocks, named Discriminator Blocks, that progressively reduce the resolution of the input feature maps. The final block, denoted as Discriminator Epilogue, output a scalar value indicating the *Discriminator*'s prediction of how real the input image is. Figure 4.7 presents an overview of the architecture of the XGAN *Discriminator* for 256x256 image resolution.

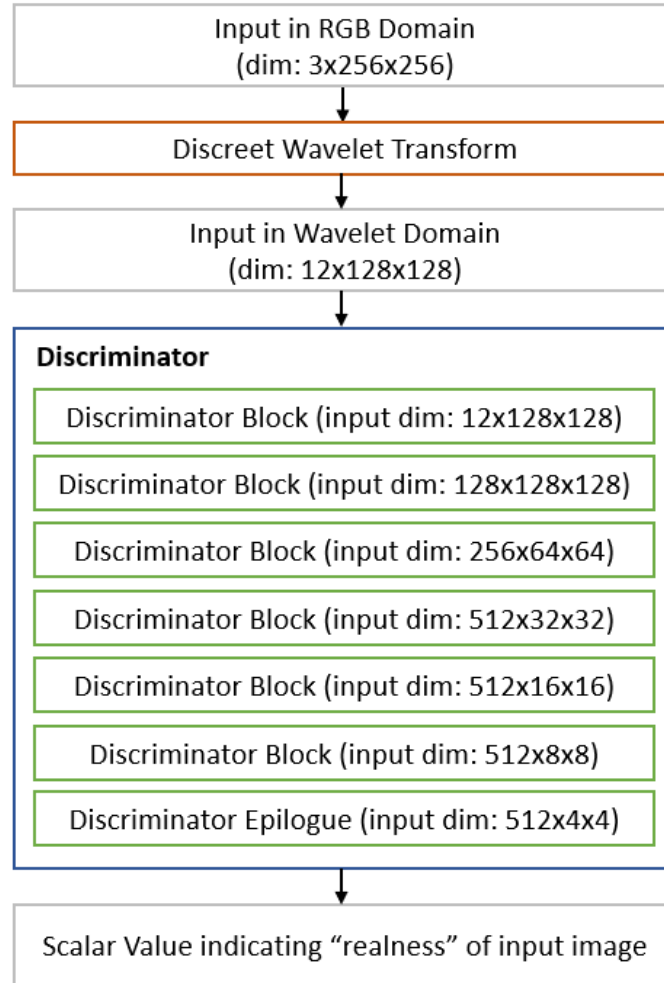


Figure 4.7: Overview of the Discriminator network of XGAN for 256x256 image resolution.

Like the Generator Style Blocks, each Discriminator Block receives two inputs and produces two outputs. The two inputs, *wavelets* and $\mathbf{x}$, are modified and passed as input to the next Discriminator Block. Input *wavelets* corresponds to a resized version of the original image passed as input to the *Discriminator* network; it is progressively downsampled in each Discriminator Block without any further modification. On the contrary, input $\mathbf{x}$ is modified by each Discriminator Block with a series of 2D convolutions, each followed by the leaky ReLU activation function. It is the only input used by the final block, the Discriminator Epilogue. Hence, only $\mathbf{x}$ is used to make the

final output prediction regarding how real the input image is. Figure 4.8 contains a visual overview of the Discriminator Block and Figure 4.9 illustrates the Discriminator Epilogue.

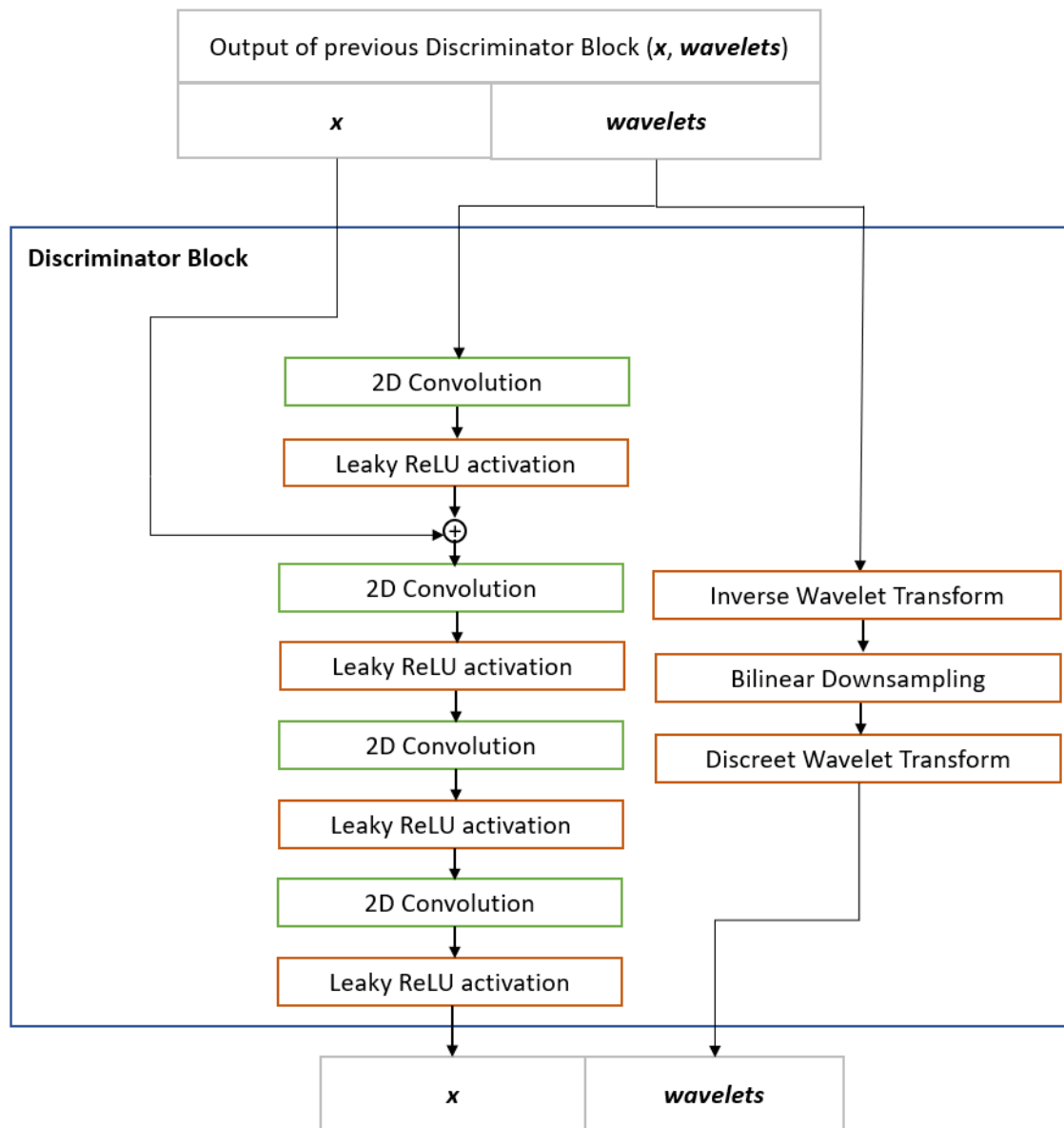


Figure 4.8: Overview of the Discriminator Block component of the XGAN Discriminator.

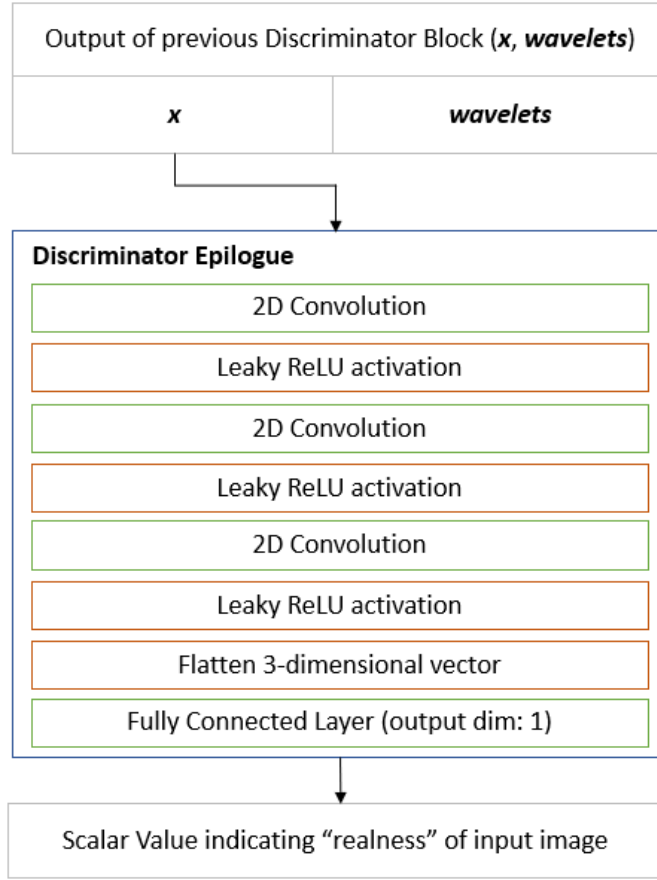


Figure 4.9: Overview of the Discriminator Epilogue component of the XGAN Discriminator.

4.3.3 Regularization

To improve training stability, XGAN employs the same regularization methods used by StyleGAN2 plus the Data Augmentation techniques introduced in [105].

- The *Generator* is constrained by **Path Length Regularization**.
- The *Discriminator* is constrained by **R1 Regularization** and **Data Augmentation**.

These methods of regularization are detailed next.

4.3.3.1 Path Length Regularization

Path Length Regularization encourages the *Generator* to produce fixed-magnitude changes in the output image by each fixed-size step taken in the styles vector $\mathbf{w}$. In other words, constant changes in $\mathbf{w}$ should produce semantic changes of fixed-magnitude in the resulting image. [43] demonstrate that this type of regularization empirically leads to higher quality images.

The Path Length Regularization loss term is calculated by keeping an exponential running average of the average Path Length and subtracting the mean Path Length of the generated fake samples from the running average. The Path Length of each minibatch is obtained by adding noise

drawn from a normal distribution to the generated images and calculating the norm of the gradients of the noisy images with relation to the styles vector $\mathbf{w}$. A hyperparameter $PLR_{strength}$ is multiplied with the loss term to control the strength of the regularization.

4.3.3.2 R1 Regularization

R1 Regularization [57] aims to penalize the *Discriminator* for deviating from the GAN game nash-equilibrium. It has been found that when the *Discriminator* deviates from the nash-equilibrium, the training stability of the *Generator* is compromised because the feedback given to the *Generator* by the *Discriminator* vanishes.

To obtain the R1 Regularization loss term, the gradients of the *Discriminator* output with relation to the samples of each minibatch of real images are first calculated: $\nabla D_{(x)}$. The final loss term is given by subtracting 1 from the norm of the gradients $\nabla D_{(x)}$ and then squaring the resulting value, as presented in Equation 4.1:

$$R_1 = (\|\nabla D_{(x)}\| - 1)^2 \quad (4.1)$$

A hyperparameter $R1_{strength}$ is multiplied with the loss term to control the strength of the regularization.

4.3.3.3 Data Augmentation

A recurring problem in GAN training is the need for vast amounts of training data [105]. When the training dataset is small, the *Discriminator* can memorize noisy details of real images and provide lousy feedback to the *Generator*, compromising training convergence. Data Augmentation addresses this issue by augmenting the input images (both real and fake) to the *Discriminator* with differentiable transformations, such as cutout and translation. Cutout consists of masking out square regions of the image and translation involves shifting all the pixels of the image in the X and Y axes. Performing these transformations encourages the *Discriminator* to learn meaningful details of the images and allows GANs to produce high-quality images on small datasets. A hyperparameter $Augment_{probability}$ controls the probability of augmenting all the images in a minibatch. Note that, unlike R1 regularization, Data Augmentation does not consist of a loss term added to the loss function of the *Discriminator*; instead it consists of augmentation operations that are performed on the input images.

Figure 4.10 illustrates Data Augmentation applied to two images.

4.3.4 Sampling Techniques

XGAN uses two techniques also employed by StyleGAN2 to improve image quality during sampling time (these techniques do not affect training). The techniques, **Truncation Trick** and **Exponential Moving Average Generator**, are detailed next.

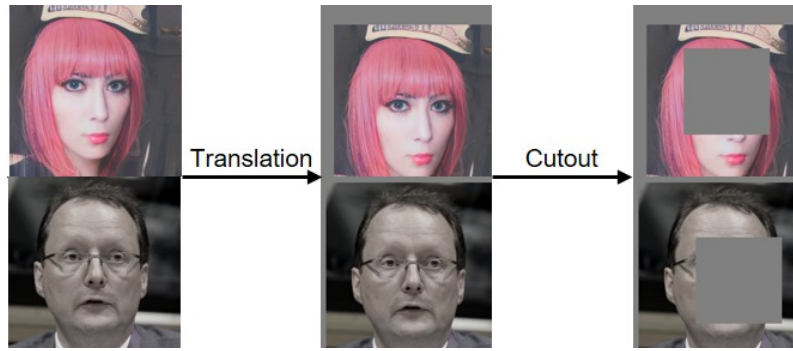


Figure 4.10: Data Augmentation regularization applies random differentiable image transformations to the input of the Discriminator, encouraging it to learn important semantic features and thus improving the feedback provided to the Generator. Figure adapted from [105].

4.3.4.1 Truncation Trick

The distribution of training data inevitably contains areas of low density that are difficult for the *Generator* to represent properly. As such, sampling images using a style vector $\mathbf{w}$ pointing to an area of low density leads to low-quality images [42]. The Truncation Trick modifies the style vectors $\mathbf{w}$ obtained from the Mapping Network to address this issue. The following steps detail how to apply the Truncation Trick:

1. Estimate the "average" style vector $\bar{\mathbf{w}}$ by sampling a sufficiently big number of styles from the Mapping Network. More precisely, obtain a large number of noise vectors $\mathbf{z}$, pass them through the mapping network, and then proceed to calculate the average vector.
2. Rescale the input style vector $\mathbf{w}$, obtaining the $\mathbf{w}'$ that will be used as input for the Synthesis Network, according to Equation 4.2:

$$\mathbf{w}' = \bar{\mathbf{w}} + \psi(\mathbf{w} - \bar{\mathbf{w}}) \quad (4.2)$$

A hyperparameter named truncation psi ψ is used to control the trade-off between image quality and diversity. Values close to 1 will lead to diverse images with mediocre quality (images corresponding to areas of low density of the training data) whilst values close to 0 will lead to similar images but with great quality.

4.3.4.2 Exponential Moving Average Generator

Training GANs is inherently unstable, and sampling from the *Generator* yields images of inconsistent quality throughout training. In other words, images generated on one epoch of training are not necessarily better visually looking than images generated on the previous epoch. To mitigate this issue, ProGAN [41] introduced the Exponential Moving Average (EMA) *Generator*. This technique consists of creating a clone of the *Generator* named *EMA Generator* at the start of

training. Its parameters are updated after each training step based on the parameters of the original *Generator*. As such, the parameters of the *EMA Generator* have less variance, which in turn increases the consistency of the output quality. Note that only the original *Generator*'s parameters are updated during backpropagation and gradient descent.

The update rule is based on an exponential moving average formula. Equation 4.3 defines the update rule, where P corresponds to the parameters of the original *Generator* and P_{EMA} corresponds to the parameters of the *EMA Generator*.

$$P_{EMA} = P_{EMA} * \alpha + P * (1 - \alpha) \quad (4.3)$$

The Exponential Moving Average decay α hyperparameter controls the degree to which the parameters of the *EMA Generator* are updated during each training step.

4.3.5 Loss Functions

XGAN is trained to optimize the hinge version of the GAN adversarial loss [62, 102, 51]. Nevertheless, other loss functions such as the standard GAN cross-entropy binary loss could instead be used and would arguably present similar results. Recent studies have shown that loss functions do not consistently outperform each other [54].

Formally, the hinge adversarial loss function is defined for the *Generator* in Equation 4.4 and for the *Discriminator* in Equation 4.5, where p_{data} is the real data distribution, p_z defines the prior noise distribution, $D(\mathbf{x})$ is the output of the *Discriminator* given sample $\mathbf{x}$ and $G(\mathbf{z})$ is the output image of the *Generator* given noise vector $\mathbf{z}$. Both networks minimize the loss functions in an alternating fashion.

$$L_G = -\mathbb{E}_{\mathbf{z} \sim p_z} D(G(\mathbf{z})), \quad (4.4)$$

$$L_D = -\mathbb{E}_{\mathbf{x} \sim p_{data}} [\min(0, -1 + D(\mathbf{x}))] - \mathbb{E}_{\mathbf{z} \sim p_z} [\min(0, -1 - D(G(\mathbf{z})))] \quad (4.5)$$

4.4 Garment Try-On

For the Garment Try-On stage of the proposed pipeline, the CP-VTON+ framework [59] was implemented by refactoring the open-source implementation provided by the authors Minar *et al.*². Unlike XGAN, CP-VTON+ is not a Generative Adversarial Network. On the contrary, it is trained in a supervised manner, and thus, a dataset containing pairs of (*garment image*, *person wearing the garment*) is required.

CP-VTON+ receives as input an image of the target person and the desired garment to dress on that person. However, the image of the target person is pre-processed using off-the-shelf libraries

²<https://github.com/minar09/cp-vton-plus>

before being served as input to the framework, as illustrated in Figure 4.11. Namely, the following components are extracted from the image of the target person:

- A pose heatmap indicating the position of the main human body joints.
- A segmentation mask with the pixels of the human body set to 1 and the rest set to 0.
- A segmentation mask with the pixels of the garment with undress set to 1 and the rest set to 0.
- An RGB image indicating the body elements to preserve during the try-on process (face and hair).

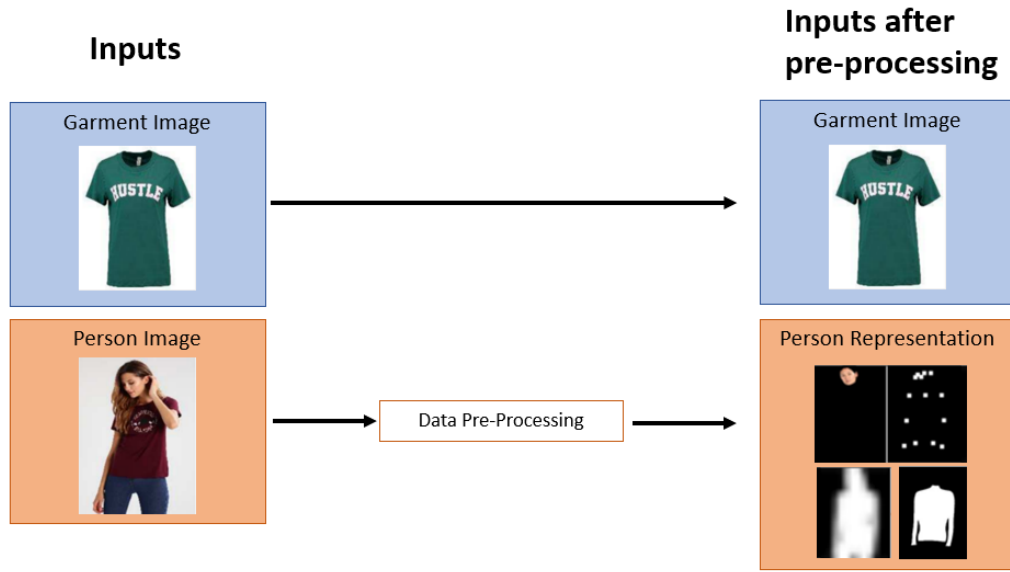


Figure 4.11: Overview of input pre-processing of the CP-VTON+ framework. Garment and Person Images reproduced from [90]. Person Representation images reproduced from [59].

CP-VTON+ is comprised of two ANN modules: a Geometric Matching Module (GMM) and a Try-On Module (TOM), each trained separately. Data is served as input to the GMM first, and the output is used by the TOM to produce the final result. The GMM produces a crude approximation of how the garment should be warped to fit the person, and the TOM refines the approximation and presents the final result. The rest of this section gives an overview of these two modules.

4.4.1 Geometric Matching Module (GMM)

The GMM transforms the garment image to fit the body of the target person. It receives as input the person representation (from the person image preprocessing step) and the garment image. It outputs a set of parameters θ for a thin-plate spline transformation T which is applied to the input garment image c , as defined in Equation 4.6:

$$\text{Warped Garment Image} = T(c; \theta) \quad (4.6)$$

Figure 4.12 presents an overview of the inputs and outputs of the module.

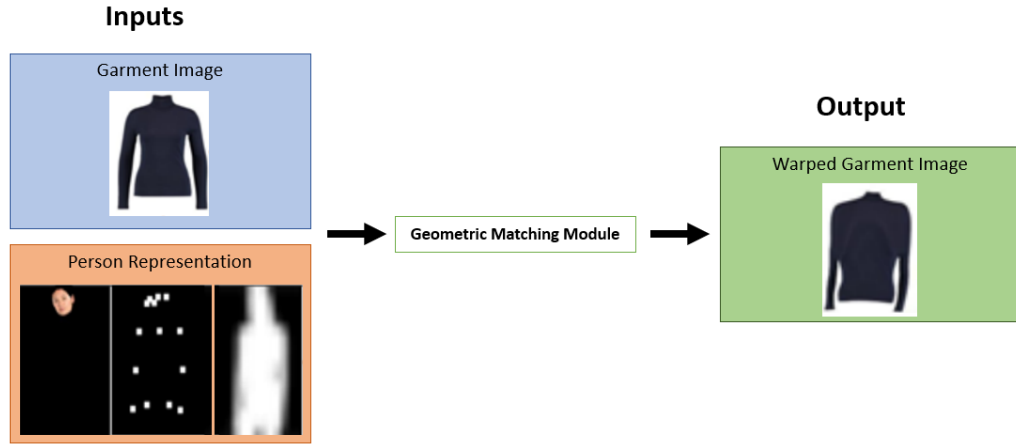


Figure 4.12: Overview of the inputs and outputs of the Geometric Matching Module of the CP-VTON+ framework. Garment and person representation images reproduced from [59].

During training, the GMM optimizes a loss function composed by two loss terms:

- $L1$ difference between a) the segmentation mask of the *output* warped garment image and b) the segmentation mask of the garment on the person *ground truth* image.
- Regularization of the thin-plate spline parameters to encourage minimal warping.

4.4.2 Try-On Module (TOM)

The TOM receives as input the warped garment image produced by the GMM and also the person representation. It produces two intermediary outputs, a composition mask M and the rendered person I . The final result of the CP-VTON+ framework - an image of the person dressing the garment depicted in the garment image - is then calculated based on pixel-wise multiplications involving M , I and the input warped garment image C as defined in equation 4.7:

$$\text{Final Result} = M \otimes C + (1 - M) \otimes I \quad (4.7)$$

Figure 4.13 presents an overview of the inputs and outputs of the Try-On Module.

During training, the module optimizes a loss function composed by three loss terms:

- $L1$ difference between a) the composition mask and b) the segmentation mask of the garment on the person *ground truth* image.
- $L1$ difference between the final result try-on image and the *ground truth* image.
- *Perceptual loss* [39] between the output try-on image and the *ground truth* image using a pre-trained VGG network [81].

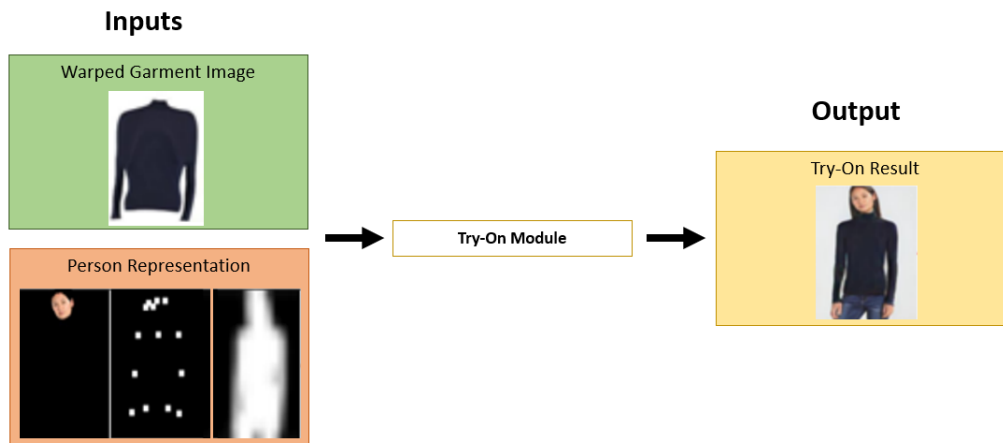


Figure 4.13: Overview of the inputs and outputs of the Try-On Module of the CP-VTON+ framework. Garment and person images reproduced from [59].

4.5 Summary

This chapter presented the general methodology utilized in conducting this research work. The proposed ANN pipeline and its two components were described in detail. Furthermore, the research process employed in this research study was outlined. The next chapter details the experiments carried out regarding the implementation and validation of the components comprising the proposed pipeline.

Chapter 5

Experimentation

The present chapter describes in detail the experimental setup for both the Unconditional Garment Design component and Garment Try-On component. The data collection process and the software libraries employed for both experiments are also described. Lastly, this chapter details the user survey used to evaluate the proposed pipeline.

5.1 Data Collection

Running the experiments proved to be a compute-intensive task requiring powerful graphics cards with a large amount of video RAM. Using personal computers with typical consumer-grade hardware is infeasible because training the pipeline components can not be completed in a reasonable timeframe. To overcome this obstacle, all experiments were carried out in Microsoft Azure Machine Learning¹ using hourly rented compute resources provided by XNFY Lab.

Data comprising generated images and the values of the evaluation metrics were collected throughout training using the Microsoft Azure SDK. For the first experiment, the data was logged at the end of each training epoch, and the Fréchet Inception Distance (FID) metric [33] was calculated using 12288 generated images and the entire training dataset. For the second experiment, generated images were collected every 10k training steps, but the SSIM (Structural Similarity Index Measure) [93] was only calculated at the end of training.

5.2 Libraries Used

Both experiments were carried out using Python 3.8. Hence, the following software libraries were used:

- **Jupyter Notebook:** this library is used to setup Microsoft Azure compute resources for training and configuration of hyperparameters.

¹<https://azure.microsoft.com/en-us/services/machine-learning/>

- **PyTorch**: this library is used to implementation of ANNs and error metrics.
- **TorchVision**: this library is used to setup PyTorch utility functions for data pre-processing of images.
- **Microsoft Azure SDK**: this library is used for logging of error metrics during training.

5.3 Experiments

In this section, the experiments regarding the Unconditional Garment Design component and Garment Try-On components are described in detail. For each experiment, the training details, datasets used, and evaluation metrics are presented.

5.3.1 Experiment 1: Unconditional Garment Design

This experiment consists of training the Unconditional Garment Design component and evaluate its performance. The methodology of this experiment can be found in Section 4.3.

5.3.1.1 Training Details

Training of the component is divided into two stages. The first stage consists of training the Unconditional Garment Design component for 100 epochs on a big image dataset comprised of dresses. Next, in the second stage, the component is re-trained for 200 epochs on a smaller dataset containing images of women upper apparel. Training on a big dataset and then re-training on a small dataset increases data efficiency [92]. This improves training stability and the quality of the generated images, by mitigating the risk of the GAN *Discriminator* overfitting.

The Adam method [45] was used for optimization with learning rate = 0.002, $\beta_1 = 0$ and $\beta_2 = 0.99$. The batch size (number of samples per training step) was set to 32. R1 regularization was applied every 20 training steps with strenght $R1_{strenght} = 10$. Path length regularization was applied every 32 training steps with strength $PLR_{strenght} = 1$. The data augmentation probability $Augment_{probability}$ was set to 0 in epochs [1, 100], set to 0.2 in epochs [101, 200] and set to 0.33 in epochs [201, 300]; it was increased over training to prevent overfitting on the smaller dataset.

5.3.1.2 Datasets Used

Two datasets were used during training: one comprised of images of dresses and one comprised of images of women upper apparel. The images of both datasets have been resized to 256x256, which corresponds to the input size of the Unconditional Garment Design component.

The first dataset is comprised of around 55.000 images of dresses and was obtained by scraping e-commerce stores of fashion retailers. It is a non-public dataset provided by XNFY Lab. Each image corresponds to a frontal photograph of a dress on a white background. Figure 5.1 shows the sample of images from the dataset.



Figure 5.1: Sample of images from the dresses dataset used during the first stage of training of the Unconditional Garment Design component.

The second dataset contains around 16,000 pairs of images of women upper apparel and fashion models wearing the garments. However, only the garment images were used for this experiment. Similar to the dresses dataset, this dataset was also obtained by scrapping e-commerce stores of fashion retailers. It is a public dataset introduced by Han *et al.* [30]. Each garment image corresponds to a frontal photograph of women upper apparel on a white background. Figure 5.2 depicts a collection of images from this dataset.

5.3.1.3 Evaluation Metric

The Fréchet Inception Distance (FID) metric [33] was used to evaluate the performance of the Unconditional Garment Design component. The FID metric measures the similarity between the training dataset and the generated images using the features of a hidden layer of an Inception-v3 network [85] trained on ImageNet [76]. A low FID value correlates with high image quality and high diversity.

The FID is calculated by estimating the mean and covariance of the activations of the last pooling layer of the Inception-v3 network, obtaining $\mu_r, \mu_g, \Sigma_r, \Sigma_g$. Then the FID value is obtained according to Equation 5.1, where $Tr(m)$ denotes the trace of matrix m :

$$FID = \|\mu_r - \mu_g\|_2^2 + Tr\left(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{\frac{1}{2}}\right), \quad (5.1)$$



Figure 5.2: Sample of garment images from the women upper apparel dataset [30] used during the second stage of training of the Unconditional Garment Design component.

Thus, the FID gives the similarity of the statistics of the activations of the last pooling layer between real and generated images. More details regarding the metric are explained in Subsection 2.3.4.

5.3.2 Experiment 2: Garment Try-On

This experiment consists of training the Garment Try-On component and evaluating its performance. The methodology of this experiment can be found in Section 3.2.1.

5.3.2.1 Training Details

The Garment Try-On component is comprised of two modules: a Geometric Matching Module (GMM) and a Try-On Module (TOM). First, the GMM was trained, and its outputs, the warped images of garments, were used to guide the training of the TOM.

Three distinct training runs were carried out with different hyperparameters to find the best-performing models. The number of training steps, batch size, and learning rate hyperparameters were changed between training runs as depicted in table 5.1. The rest of the hyperparameters and training settings are the same as reported by the research work of CP-VTON+ [59]. Namely, the Adam optimization method was used with $\beta_1 = 0.5$, $\beta_2 = 0.999$ and the learning rate was decayed linearly to 0 after 50% of the training steps were completed.

Training Run	Number of Training Steps	Batch Size	Learning Rate
1	200000	4	1e-4
2	200000	16	1e-4
3	300000	16	2e-4

Table 5.1: Hyperparameters that were changed between training runs of the Garment Try-On component.

5.3.2.2 Datasets Used

The women upper apparel dataset introduced by Han *et al.* [30] was used during training of the Garment Try-On component. This dataset is composed of image pairs (garments and fashion models wearing the garments) which were used to train the GMM and TOM modules in a supervised learning way. Figure 5.3 presents a sample of image pairs from the dataset.

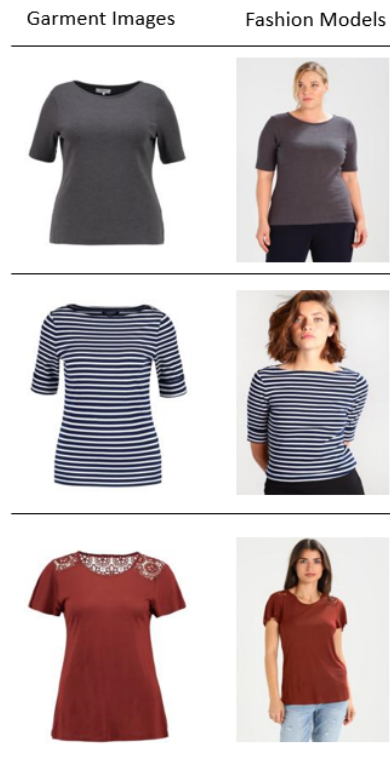


Figure 5.3: Sample of image pairs from the women upper apparel dataset [30] used to train the Garment Try-On component.

5.3.2.3 Evaluation Metric

The SSIM (Structural Similarity Index Measure) [93] metric was used to evaluate the performance of the Garment Try-On component. The SSIM measures the similarity between two images. It can be viewed as a quality measure of one image with relation to another image. In this experiment, the SSIM is used to compare the similarity between the output try-on results with relation to the ground truth images of the fashion models wearing the garments. Figure 5.4 illustrates the evaluation process.

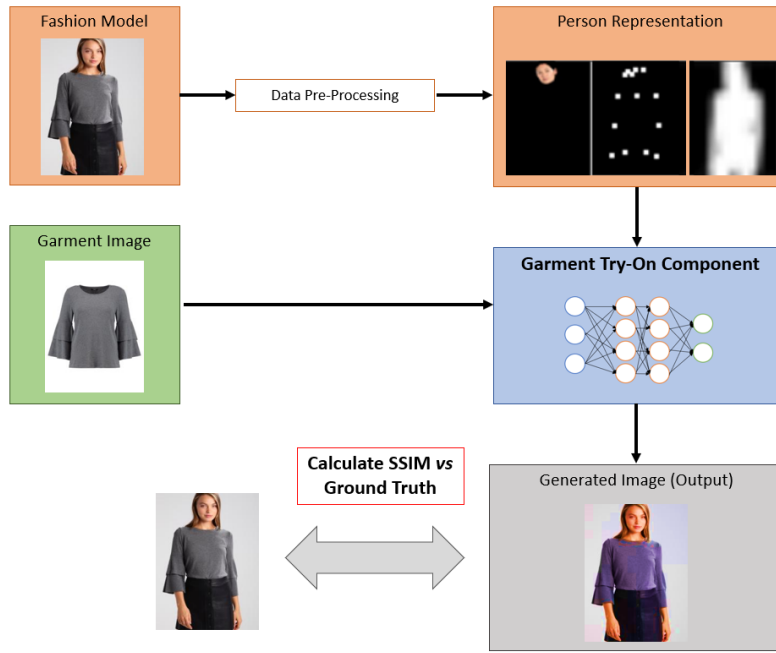


Figure 5.4: Overview of the evaluation process for the Garment Try-On Component. Garment images reproduced from [30].

The SSIM metric is calculated on the nonoverlapping square regions of the two images, and the final metric value is given by the average SSIM for each of the regions. The SSIM between two square regions x and y of width N is defined by equation 5.2, where μ_x and μ_y are the average pixel values for x and y , σ_x and σ_y are the variance of the pixel values for x and y , σ_{xy} is the covariance between the pixel values of x and y , $C_1 = 1e-4$ and $C_2 = 9e-4$ are two constants that provide numerical stability when the fraction denominator is close to zero.

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1) + (2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (5.2)$$

5.4 User Survey

A user survey was conducted to evaluate the performance of the proposed ANN pipeline.

First, garment images were generated by the Unconditional Garment Design component. Those images were then passed as input to the Garment Try-On component to generate images of fashion models wearing the garments. Both generated images and real images from the women's apparel dataset were used to design the user survey questions.

The conducted user survey aimed to assess the a) realness and b) marketability of the generated images. Hence, the survey questions were divided into two groups:

- **Group 1:** comprised of 5 single selection questions and 5 open-ended questions. Participants were asked to distinguish between generated and real images (single selection questions) and then explain what errors and artifacts in the generated images influenced their decision (open-ended questions).
- **Group 2:** comprised of two single selection questions. Participants were asked their opinion regarding the marketability of garments depicted on images. The questions were formulated as "Customers are likely to buy the depicted garment", with the possible answers being those from the Likert Scale [50]: "Strongly Agree", "Agree", "Neutral", "Disagree", "Strongly Agree".

The user survey questions can be found in Appendix A.

Workers of Microsoft Portugal and students of the Integrated Master's in Computing Engineering at the Faculty of Engineering of the University of Porto were asked to participate in the user survey. A total of 38 volunteers participated in the user survey.

5.5 Summary

This chapter discussed the experimental setups for the two conducted experiments, including training details, datasets used and performance metrics used. The data collection process and the software libraries used for both experiments were also described. Furthermore, the devised user survey used to evaluate the proposed ANN pipeline was also detailed. The next chapter contains the results and discussion of each experiment of this research and an analysis of the user survey results.

Chapter 6

Results and Discussion

This chapter presents the results and discussions of the conducted experiments regarding the components of the proposed pipeline: the Unconditional Garment Design component and the Garment Try-On component. Furthermore, the results of the user survey are presented and analyzed.

6.1 Experiment 1: Unconditional Garment Design

This section discusses the experimental results of the Unconditional Garment Design component. The component was trained in two stages. The first stage consisted of training for 100 epochs on a big dataset comprised of dresses. The second stage involved fine-tuning the components on a dataset of women upper apparel for 200 epochs. The Fréchet Inception Distance (FID) evaluation metric, which inversely correlates with image quality, was collected during training to evaluate the performance of the component. Furthermore, images generated by the component throughout training were also collected.

Figure 6.1 presents the FID of the component per training epoch for both training stages. Figures 6.2 and 6.3 depicts generated images during the first training stage and the second training stage, respectively.

6.1.1 First Training Stage

Regarding the first training stage, the FID metric showed high variance for the first 20 epochs and low variance for the remaining 80 epochs, as presented in Figure 6.1a. The initial value for the FID was 324, and, at the end of training it was 11.5. It kept decreasing continuously and did not seem to be reaching a plateau. Thus, arguably, if training had continued, the FID would have reached even lower values, indicating higher image quality. However, the aim of this training stage was not to obtain a model capable of generating dresses but instead to increase data efficiency, preventing overfitting on the smaller dataset used during the second training stage.

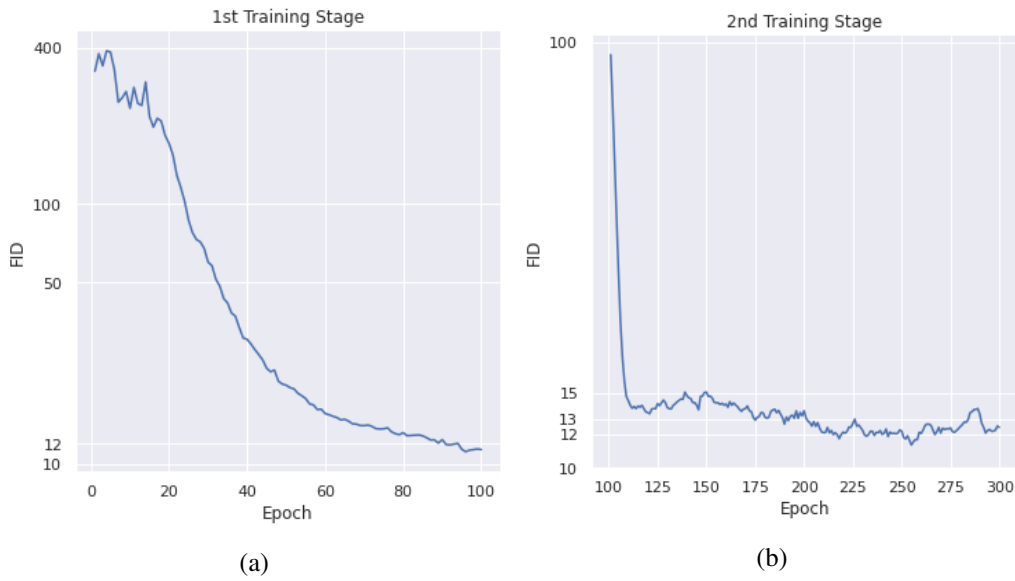


Figure 6.1: FID metric per training epoch for the first (a) and second (b) stages of training of the Unconditional Garment Design component. The Y axis of both plots is depicted with a logarithmic scale.

The increase in the quality of the generated images of the component, depicted in Figure 6.2, is well explained by the values of the FID metric. Throughout training, the component produces images of increasingly definition: the shape structure and colorization of the images become progressively better. At the end of the training stage, the generated dresses have complex texture details and resemble real dresses at first glance. Nevertheless, some decolorization artifacts are still present in the garments because the component struggles to learn global-level details, making the images lack color consistency.

6.1.2 Second Training Stage

As for the second training stage, the FID started at 93 and quickly dropped to below 15 after ten epochs, as can be seen in Figure 6.1b. The component had already learned some clothing details, such as shape and color on the first training stage. Thus, learning the specific details for the second dataset took significantly less time. This phenomenon is often referred to as transfer learning [92].

The FID significantly decreased during the first ten epochs and remained stable throughout the rest of training. Nevertheless, it slowly converged to a final value of 12.7, and it seems that it had hit a plateau at that point. The lowest value FID reached was 11.3 on epoch 255, but it slowly increased over the following 30 epochs before decreasing again. Thus, it seems unlikely that more training time would present any improvement, possibly even worsening performance through overfitting.

The quality of the generated images increases during training, as shown in Figure 6.3. This increase in image quality is validated by the decrease of the FID. During the first 10 epochs, the

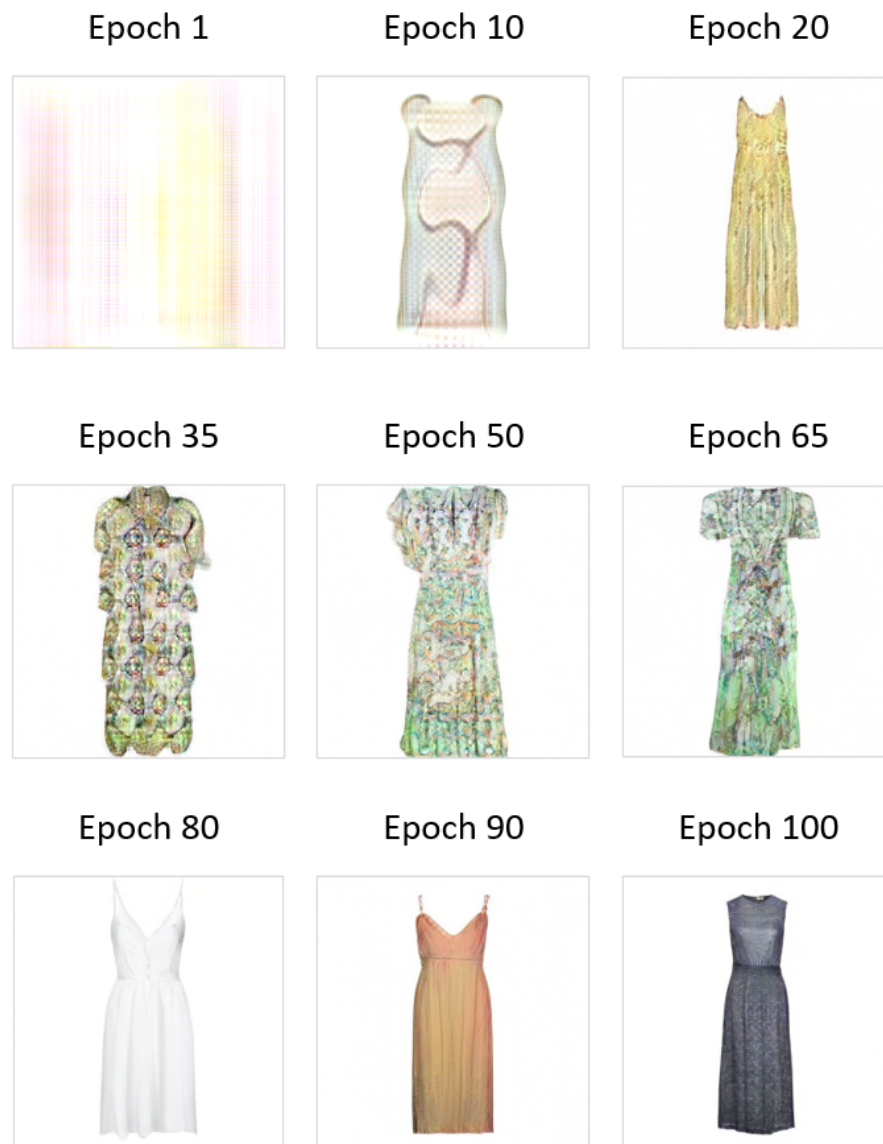


Figure 6.2: Generated images collected from the component throughout the first stage of training of the Unconditional Garment Design component.

image quality improved significantly but did not vary considerably during the rest of training. Clothing shape was quickly learned during the first 5 epochs and remained stable during the rest of training. However, it can be seen that the images contain the same type of artifacts as those of the first training stage. A lack of consistent coloring can also be observed on most images, even on later epochs. Furthermore, texture details are absent, and most generated garments are single-colored without any extra detail such as stripes or text.

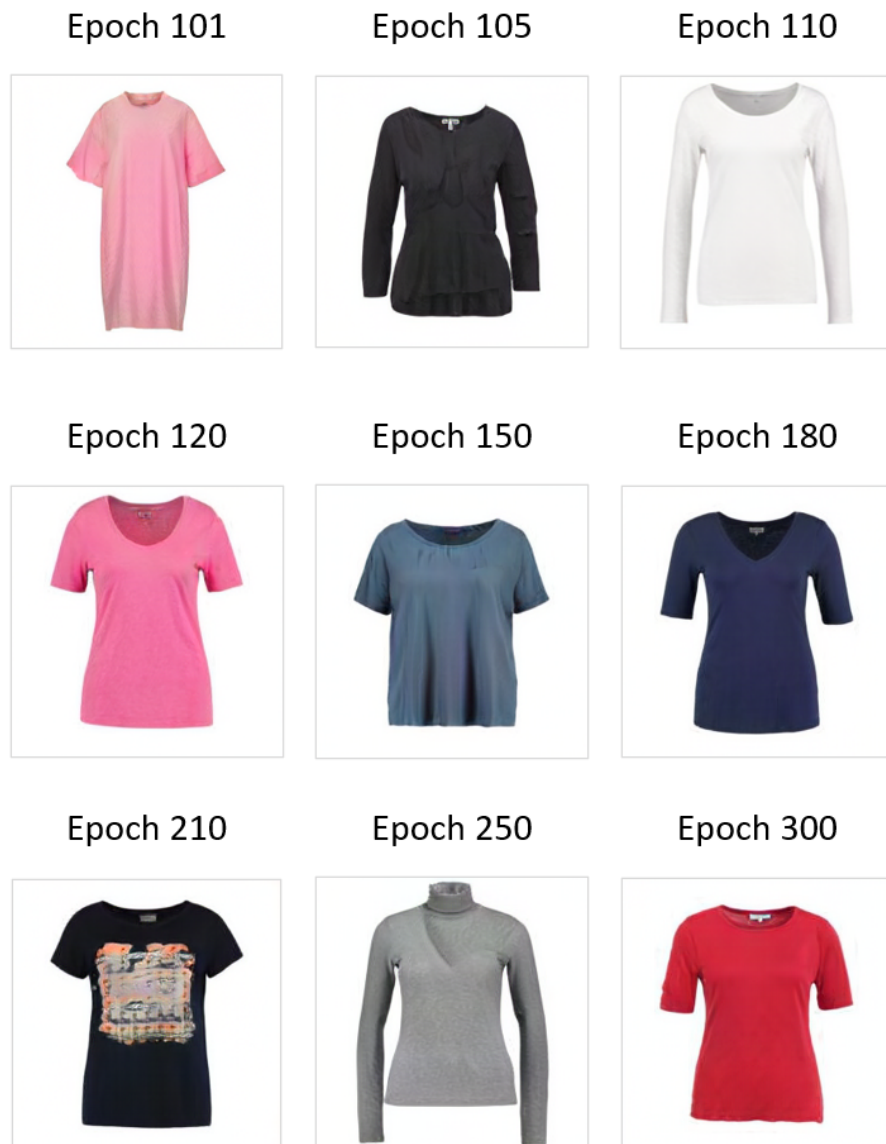


Figure 6.3: Generated images collected from the component throughout the second stage of training of the Unconditional Garment Design component.

6.2 Experiment 2: Garment Try-On

This section discusses the experimental results of the Garment Try-On component. The evaluation metric used to assess the quality of the produced images was the SSIM (Structural Similarity Index Measure) [93], which produces a value in the range of 0 to 1, where 1 indicates perfect image quality and 0 the opposite. The SSIM was calculated in the training set and a separate test set for the three training runs after training was completed. Three training runs were performed with different hyperparameters values for the number of training steps, learning rate, and batch size. The results are shown in Table 6.1.

The SSIM values for all training runs are similar. Thus, changing the number of training steps,

Training Run	Training Set SSIM	Test Set SSIM
1	0.778	0.654
2	0.772	0.646
3	0.780	0.647

Table 6.1: SSIM results measured after training for the three training runs. Bold indicates the best scores for the training and test sets.

the batch size, and the learning rate does not lead to better-performing models. Moreover, the test set SSIM was significantly lower than the training set SSIM for all training runs, meaning that all models overfitted to the training data. Training run 3 achieved the best SSIM score in the training set but not in the test set, meaning it does not generalize well to unseen data.

Training runs 2 and 3 were trained on a higher number of total images than training run 1 due to increased values for the batch size and training steps hyperparameters. Nevertheless, they obtained worse test set SSIM scores than training run 1, indicating that longer training does not lead to models with better performance for the particular case of the Garment Try-On component. Therefore, training for 200k steps with a batch size of 4 and a learning rate of $1e-4$ (the hyperparameters of training run 1) is the recommended approach.

Training run 1 attained the highest test set SSIM score. Hence, it was chosen to compose the Garment Try-On component of this thesis' proposed ANN pipeline. Figure 6.4 shows outputs of the training run 1 model and the corresponding ground truth images. Visually inspecting the outputs, it can be stated that decolorization is present on all try-on generated images as the model fails to retain the same color on all regions of the garment. Furthermore, fabric folds do not exist on the generated images but occur commonly on the ground truth images.

Text details on the chest area of the garments are not correctly preserved by the model when generating the try-on result. The model also fails to preserve some details related to the person's identity, such as skin color. One other failure case occurs when the person's hands are placed in front of the garments, producing significant color artifacts in the hands and the surrounding area.

Moreover, the generated try-on images are blurrier than their ground truth counterparts, and checkerboard artifacts are present on the collar and hip areas of the garments. Given the large number of artifacts and other errors present in the generated images, it can be concluded that distinguishing between the generated and ground truth images is simple.



Figure 6.4: Try-On results for the training run 1 model. The input garment images and the ground truth images are also depicted.

6.3 User Survey

This section discusses the results of the user survey detailed in Section 5.4. The user survey questions can be found in Appendix A. The aim of the user survey was to evaluate the performance of the proposed ANN pipeline in terms of i) the realness and ii) the marketability of the generated garments. Hence, the user survey comprises two groups of questions. The first question group contains questions regarding the realness of the generated garments, while the second question group contains questions regarding the marketability of the generated garments.

A total of 38 volunteers participated in the user survey and all responded to both question groups. The rest of this section discusses the results for each question group.

6.3.1 Question Group 1

The first question group contains questions regarding the realness of the images. Participants were asked to distinguish between generated and real images (5 single selection questions) and then explain what errors and artifacts in the generated images influenced their decision (5 open-ended questions).

Each single selection question contains two image pairs: *A* and *B*. Each image pair comprises i) an image of a garment and ii) an image of a fashion model wearing the garment. The participants were asked to identify the generated image pairs with four possible answers: "*A was generated by AI*", "*B was generated by AI*", "*A and B were generated by AI*" and "*None were generated by AI*". The single selection questions were proceeded by open-ended questions asking the participants to indicate what made them perceive the identified generated image pairs as non-real.

The participants answers for the single selection questions are summarized in Figure 6.5 and the percentage of participants who answered correctly each question is presented in Table 6.2.

Single Selection Question	Participants who answered correctly (%)
1	39%
2	63%
3	32%
4	53%
5	39%

Table 6.2: Percentage of correct answers for the single selection questions.

Most participants failed to identify the generated image on single selection questions 1, 3, and 5. However, for the single selection questions 2 and 4, the opposite was true; most participants answered correctly.

Single selection question 3 has the lowest correct answer rate, with only 32% participants identifying the correct image. Single selection question 2 has the highest correct answer rate, with 63% participants choosing the correct answer.

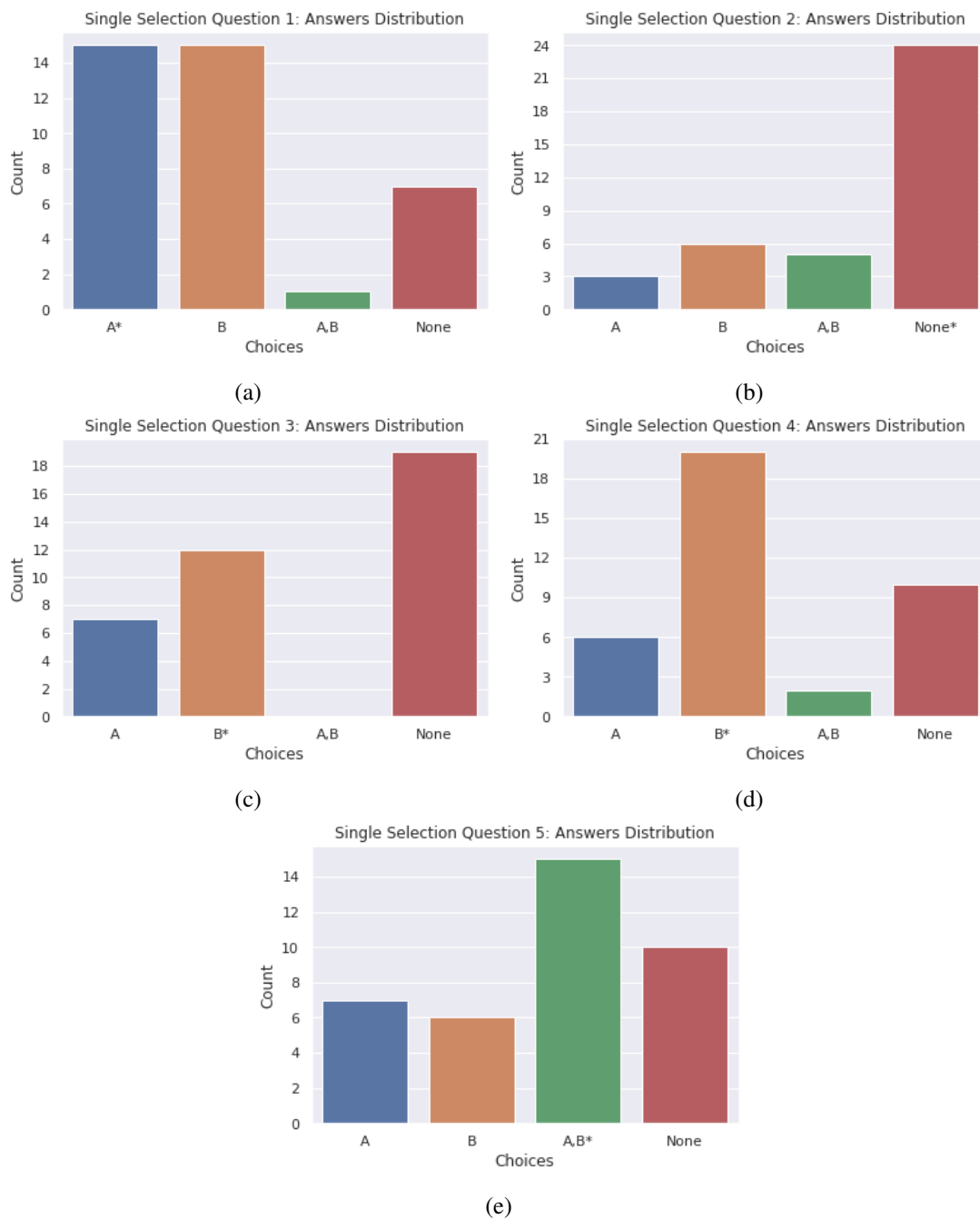


Figure 6.5: User survey answers for the single selection questions. The * next to an answer choice denotes the correct answer for the question.

Figure 6.6 illustrates the number of participants who answered a given number of questions correctly. On average, the participants answered 2.2 questions correctly out of 5, with a standard deviation of 1.6. The median for the number of correct answers of each participant was 2. Interestingly, 16% of the participants answered all questions wrongly, and only 11% answered all questions correctly. About 52% of the participants failed to distinguish between generated and real

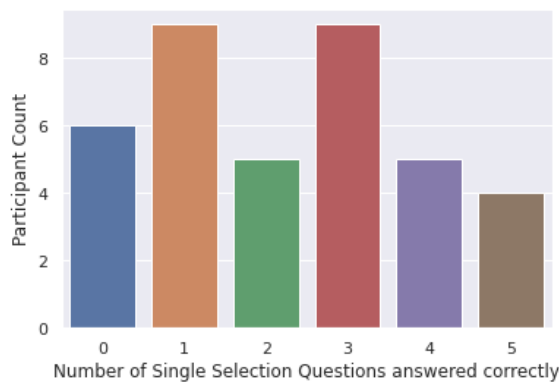


Figure 6.6: Number of single selection questions answered correctly by the participants.

garments on more than half of the survey questions. Thus, it can be concluded that the generated garments are very similar to real ones.

Participants were asked to justify each of their answers for the single selection question, indicating what errors and artifacts were present in the images. The justifications were then analyzed and placed into one or multiple categories to facilitate analysis. The justifications of participants with relation to single selection questions answered incorrectly were ignored since the participants failed to identify the generated images, and, thus, their justification is not helpful to understand the quality problems of the generated images. A total of 52 justifications were obtained.

The categories and the number of corresponding justifications are presented next:

- **Image artifacts around the collar:** 17 justifications (32%). The participants identified image artifacts around the collar area of the generated garments.
- **Strange colors and patterns:** 17 justifications (32%). The participants identified strange coloring or strange patterns on the generated garments.
- **Lack of realistic texture details:** 10 justifications (19%). The participants found that the generated garments lack realistic texture details such as rugs and folds.
- **Global Image Artifacts:** 10 justifications (19%). The participants identified global image artifacts in the generated garments, such as overlapping artifacts (caused by dressing the fashion models with a different garment) and blurry image regions.
- **Strange shape:** 5 justifications (10%). The participants identified problems regarding the shape definition of the generated garments.
- **Image artifacts in the chest area:** 5 justifications (10%). The participants identified image artifacts in the chest area of the generated garments.
- **Simple Design:** 3 justifications (6%). The participants justified their answer on the basis that the generated garments have a simpler design compared to the real garments.

The participants justified their decision based on local errors of the garment images (artifacts on the collar and the chest area) and on global errors that span more than a single image region

(strange colors, lack of realistic texture details, global image artifacts, strange shape). Furthermore, some participants found the generated garments to be simpler in design when compared to the real garments.

6.3.2 Question Group 2

The second question group contains two questions regarding the marketability of garments. The first question depicts a real garment and the second question depicts a generated garment. Participants were asked if they think it is plausible for the garments depicted on the questions' images to be purchased by customers. The questions were formulated as "*Customers are likely to buy the depicted garment*", with the possible answers being those from the Likert Scale [50]: "*Strongly Disagree*", "*Disagree*", "*Neutral*", "*Agree*", "*Strongly Agree*". The answer distributions for the two questions are shown in Figure 6.7.

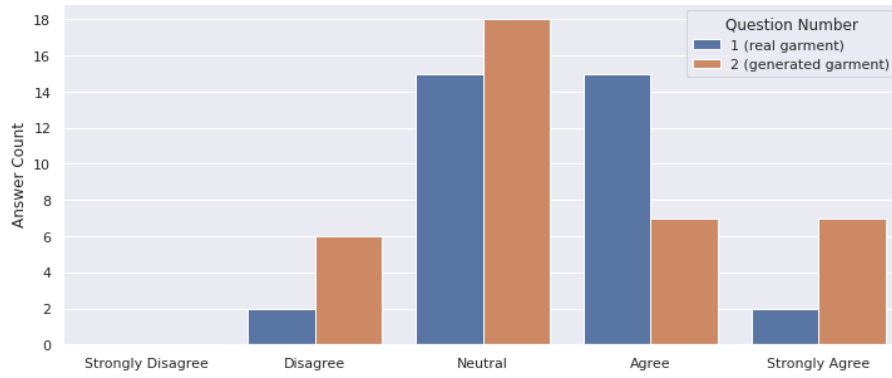


Figure 6.7: Answers distribution for the second question group of the user survey.

The following analysis question was then formulated:

Is the marketability of the generated and real garments as reported by the participant the same?

The answers from the participants were converted to an ordinal 1 to 5 scale, in which 1 denotes "*Strongly Disagree*" and 5 denotes "*Strongly Agree*". The following hypotheses were then formulated:

- H_0 (null hypothesis): The marketability of the generated and real garments as reported by the participants is the same.
- H_1 (alternative hypothesis): The marketability of the generated and real garments as reported by the participants is *not* the same.

Given that the data is not independent since the same participants answered both questions, the non-parametric Wilcoxon signed-rank test [94] is used.

The p-value obtained is 0.071, and, as such, it is not possible to reject the null hypothesis - *The marketability of the generated and real garments as reported by the participants is the same* - at a 95% confidence level.

6.3.3 Results Summary

Regarding the results of the user survey, the following observations were made:

- Most participants failed to distinguish between real and generated garments. The average participant answered correctly 2.2 out of 5 times when asked to identify the generated garments. Furthermore, about 52% of the participants failed to distinguish between generated and real garments on more than half of the survey questions.
- The participants who correctly identified the generated garments justify their decisions based on i) image artifacts around the collar and chest area of the garments, ii) strange colors and patterns, iii) lack of realistic details on the garments, iv) global image artifacts such as blurriness, v) shape issues on the outlines of the garments, and, vi) simple designs of the garments.
- The marketability of the real and generated garments as reported by the participants is not different in a statistically significant way. A hypothesis test was conducted, and the null hypothesis "*The marketability of the generated and real garments as reported by the participants is the same*" could not be rejected at a 95% confidence level.

6.4 Summary

This chapter has presented the results and discussions for the conducted experiments regarding the two components of the proposed pipeline: the Unconditional Garment Design component and the Garment Try-On component. Moreover, the results of the user survey were also discussed and analyzed. The next chapter concludes this research work by summarizing the research findings, describing limitations of the conducted research, presenting the research contributions, and outlining recommendations for future research directions.

Chapter 7

Conclusion

This chapter concludes this research work by summarizing the research findings, describing the limitations of the conducted research, presenting the core research contributions, detailing challenges faced, and giving recommendations for future research directions.

7.1 Summary of Research Findings

The present dissertation proposed an ANN-based pipeline for the tasks of; i) automatic generation of clothing items and ii) dressing fashion models with the generated clothing. Three research objectives were formulated to guide the research methodology of the proposed pipeline:

- **Research Objective 1** - To investigate the existing studies on fashion datasets and the applications of AI to fashion.
- **Research Objective 2** - To propose and implement an ANN-based pipeline that sequentially performs i) automatic generation of clothing items and ii) dressing fashion models with the generated clothing.
- **Research Objective 3** - To evaluate the performance of the proposed ANN-based pipeline in terms of performance metrics and user surveys.

This dissertation aims at fulfilling these research objectives. The following points justify that each research objective was fulfilled:

- To investigate the existing studies on fashion datasets and the application of AI to fashion.

This research objective was achieved by reviewing research works on AI for fashion and existing fashion datasets as discussed in Chapter 3. Applications of AI for fashion such as pose transfer, garment translation, garment design, garment retrieval, and garment manipulation were

studied and presented in detail. Studies and works on these applications were also reviewed. Furthermore, it was observed that the works of AI applications for fashion are based on ANNs and GANs. Therefore, a background review on ML, ANNs, and GANs was conducted to aid the development of the methodology of this study, as presented in Chapter 2.

- To propose and implement an ANN-based pipeline that sequentially performs i) automatic generation of clothing items and ii) dressing of fashion models with the generated clothing.

This objective was fulfilled by providing an ANN-based pipeline that generates fashion garments and dresses them on fashion models. The proposed pipeline is comprised of two components: an Unconditional Garment Design component and a Garment Try-On component. The Unconditional Garment Design component generates images of garments unconditionally. The Garment Try-On component takes the generated garment images and produces images of fashion models dressing the garments. The methodology of the proposed pipeline and the two components can be found in Section 4.2, Section 4.3 and Section 4.4.

- To evaluate the performance of the proposed ANN-based pipeline in terms of performance metrics and user surveys.

This objective was achieved by collecting performance metrics and by conducting a user survey regarding the performance of the pipeline. The performance metrics obtained (FID and SSIM) report that the produced images of the pipeline are of high quality. The aim of the user survey was to assess the performance of the proposed ANN pipeline in terms of i) the realness and ii) the marketability of the generated garments. It was observed that most participants failed to distinguish between real and generated garments. Furthermore, the participants' answers suggest that the marketability of the generated and real garments is the same.

7.2 Research Contributions

The core contributions of this dissertation are as follows:

1. A literature review on AI for fashion and a background review on ML, ANNs, and GANs. Applications of AI for fashion were studied, along with studies and datasets of these applications. Furthermore, a study on ML, ANNs, and GANs was conducted since most AI fashion works are based on these research areas.
2. A novel ANN-based pipeline that generates fashion garments and dresses fashion models with the generated garments. The pipeline is comprised of two components: an Unconditional Garment Design component and a Garment Try-On component. First, the Unconditional Garment Design component generates images of novel garments. Then, these garments are served as input to the Garment Try-On component, which produces images of fashion models dressing the garments.

3. An evaluation of the proposed pipeline in terms of performance metrics and user surveys. The collected performance metrics (FID and SSIM) reveal that the generated images are of high quality. The user survey evaluated the realness and marketability of the generated garments. It was observed that the average survey participant answered correctly 2.2 out of 5 times when asked to distinguish between generated and real garments. Furthermore, a hypothesis test on the participant's answers concluded that the marketability of the real and generated garments is not different in a statistically significant way.

7.3 Threats to Validity

One of the limitations of the present study is the usage of datasets comprised only of women's upper clothing. As a result, the garments generated by the pipeline are only composed of images of women's upper clothing. However, diverse clothing is required to aid the work of fashion designers since most fashion brands target both genders at varied age ranges. As it currently stands, it remains unclear if the proposed pipeline is capable of generalizing to male apparel or fashion accessories such as handbags, scarves, and alike. Moreover, the fashion models dressing the generated garments have similar bodies with typical characteristics such as medium height and average weight. Nonetheless, the general population is highly diverse, and most people's bodies drastically differ from those of fashion models. It is uncertain if using fashion models with more diverse bodies will lead to results similar to those obtained by this thesis's experiments.

Furthermore, the collected answers for the conducted user survey do not comprise a representative sample of all fashion customers, given that the survey participants are workers of Microsoft Portugal and informatics engineering students. Although no personal details were asked of the participants, it is reasonable to expect that most survey participants are male since most workers from Microsoft, a technological company, and most students of informatics engineering are also male. Thus, the conclusions drawn from the participant's answers cannot be extrapolated to the general population. Moreover, all the survey questions depict women's clothing, casting doubts on the validity of the survey results since the male participants might have underestimated or overestimated the clothing's realness and marketability.

7.4 Challenges Faced

The ANN architectures devised for this thesis methodology are very computational intensive since they require millions of floating-point operations to produce images. As such, running the pipeline experiments on traditional consumer-grade hardware is infeasible due to significant training times. To overcome this obstacle, the experiments were carried out on the Microsoft Azure cloud environment using monetary funds provided by XNFY Lab. Powerful computers with numerous GPU cards aided the research of this thesis, allowing experiments that would otherwise take weeks to be completed in few days.

However, running the experiments on a cloud platform posed considerable challenges. For once, performing code changes and waiting for the modified code to run took up to 10 minutes, instead of a few seconds as is the case when the code runs locally on a personal computer. As a result, debugging became a long and frustrating process. Moreover, conducting experiments on Microsoft Azure entailed many code modifications originating from the need to use the Azure SDK library to perform typical ML tasks such as logging performance metrics and saving trained models. As such, the initial codebase, which was implemented to run in a Jupyter Notebook, had to be completely reworked.

Despite the powerful hardware used, the experiments still took a significant amount of time to be completed. For instance, training the pipeline to generate 256x256 images took days, which hindered hyperparameter tuning and the development of new features. Furthermore, improving the ANNs architectures with new techniques from the literature was a time-consuming operation since implementation errors only became apparent after hours of training. Therefore, a substantial part of research time consisted of looking at the collected performance metrics of ongoing training runs in order to figure out the existence of bugs quickly and thus avoid unnecessary computational costs resulting from failed experiments.

7.5 Future Research Directions

The present work has focussed on generating images of novel garments to facilitate the workflow of fashion designers. However, the shapes and color patterns of the generated garments still have to be recreated in Computer-Aided Design (CAD) software by the fashion designers in order for the garments to be manufactured. Instead, a more direct approach would be generating 3D models of garments, allowing the generated garments to be directly imported into CAD software and thus reduce the design time of new fashion apparel further.

Moreover, the proposed pipeline can only unconditionally generate new garments, i.e., not subject to user-defined criteria. Nevertheless, in some scenarios, it is more helpful to control the generation process towards garments with particular characteristics such as specific colors, patterns, and shapes. Generating garments conditionally can help fashion designers iterate on new ideas faster and thus bring tremendous value to the fashion industry. Thus, future work on Conditional Garment Design is a promising research direction.

Additional work on improving the ANN architectures of the proposed pipeline is also required. The devised neural architectures are very computationally demanding, making them incapable of being used on consumer-grade computers. As a result, embedding the proposed pipeline in a software application targeting fashion designers is difficult since powerful computing hardware is required. It is also necessary to devise new techniques to scale the neural architectures to higher image resolutions. The experiments of the present thesis focused on the generation of 256x256 resolution images, and no hindrances were found during training. Some unreported research experiments were conducted regarding 512x512 image resolution, but training failed to converge due to high instability regarding the shapes and colors of the generated garments. Nonetheless, to

aid the workflow of fashion designers, images with resolution equal to or above 1024x1024 are necessary. Otherwise, the generated low-resolution garment images lack fine-grained level details and realistic textures.

7.6 Final Words

Fashion is one of the most essential arts of civilization. However, fashion trends are affected by many unpredictable factors, making fashion the most unpredictable art of all. As a result, fashion brands struggle with providing consumers with clothing aligned with ongoing market trends in a timely manner. Moreover, designing and shipping clothing to consumer markets remains a lengthy process. The present work introduces an AI-based solution that complements the workflow of fashion designers, shortening the design process of new products. First, a literature review on studies of AI for fashion was conducted. Based on the studied works, a novel pipeline based on ANNs was then proposed for the tasks of generating fashion garments and dressing fashion models in the generated garments. Lastly, to assess the performance of the proposed pipeline, performance metrics were collected, and a user survey was conducted. The values of the performance metrics (FID and SSIM) indicate that images produced by the pipeline are of high quality. The survey results suggest that the generated garments resemble real ones. Furthermore, the marketability of generated and real garments as reported by the survey participants is suggested to be the same.

Appendix A

User Survey

The user survey conducted to evaluate the proposed ANN pipeline is presented next verbatim. The following images shown in the survey's questions were generated by the implemented ANN pipeline:

- Question 1: A
- Question 3: None
- Question 5: B
- Question 7: B
- Question 9: A and B
- Question 11: None
- Question 12: Depicted Image (no letter)

Survey Questions

This survey aims to evaluate the performance of an AI (Artificial Intelligence) method that i) designs new fashion garments and ii) dresses them on fashion models.

Participants will be asked questions regarding the realness and marketability of women upper apparel shown in images.

Group 1

This group is comprised of 10 questions: 5 single selection questions and 5 open-ended questions.

Please identify the AI-generated image pairs and indicate what artifacts and errors present in those images influenced your decisions.

Question 1



Please identify which pairs of images were AI-generated.

- ☐ A was generated by AI
- ☐ B was generated by AI
- ☐ A and B were generated by AI
- ☐ None were generated by AI

Question 2

Justify your decision (don't answer if no AI-generated image was identified in the previous question).

Question 3

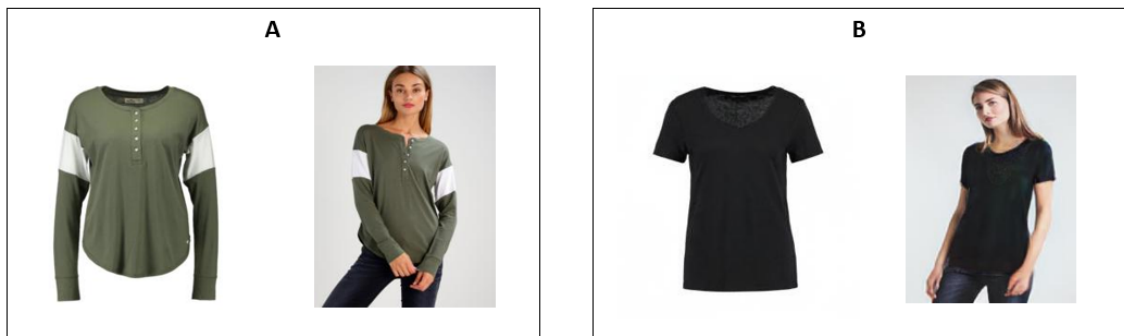


Please identify which pairs of images were AI-generated.

- ☐ A was generated by AI
- ☐ B was generated by AI
- ☐ A and B were generated by AI
- ☐ None were generated by AI

Question 4

Justify your decision (don't answer if no AI-generated image was identified in the previous question).

Question 5

Please identify which pairs of images were AI-generated.

- ☐ A was generated by AI
- ☐ B was generated by AI
- ☐ A and B were generated by AI
- ☐ None were generated by AI

Question 6

Justify your decision (don't answer if no AI-generated image was identified in the previous question).

Question 7

Please identify which pairs of images were AI-generated.

- ☐ A was generated by AI
- ☐ B was generated by AI
- ☐ A and B were generated by AI
- ☐ None were generated by AI

Question 8

Justify your decision (don't answer if no AI-generated image was identified in the previous question).

Question 9

Please identify which pairs of images were AI-generated.

- ☐ A was generated by AI
- ☐ B was generated by AI
- ☐ A and B were generated by AI
- ☐ None were generated by AI

Question 10

Justify your decision (don't answer if no AI-generated image was identified in the previous question).

Group 2

This group is comprised of 2 questions regarding the marketability of fashion garments.

Question 11



Customers are likely to buy the depicted garment.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neutral
- ☐ Disagree
- ☐ Strongly Disagree

Question 12



Customers are likely to buy the depicted garment.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neutral

- ☐ Disagree
- ☐ Strongly Disagree

References

- [1] Kenan E Ak, Joo Hwee Lim, Jo Yew Tham, and Ashraf A Kassim. Attribute manipulation generative adversarial networks for fashion images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10541–10550, 2019.
- [2] Kenan E. Ak, Joo Hwee Lim, Jo Yew Tham, and Ashraf A. Kassim. Semantically consistent text to fashion image synthesis with an enhanced attentional generative adversarial network. *Pattern Recognition Letters*, 135:22 – 29, 2020.
- [3] Kenan Emir Ak, Joo Hwee Lim, Jo Yew Tham, and Ashraf Kassim. Semantically consistent hierarchical text to fashion image synthesis with an enhanced-attentional generative adversarial network. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3121–3124. IEEE, 2019.
- [4] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.
- [5] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223. PMLR, 2017.
- [6] Nobukazu Azuma and John Fernie. Fashion in the globalized world and the role of virtual networks in intrinsic fashion design. *Journal of Fashion Marketing and Management: An International Journal*, 2003.
- [7] Shane Barratt and Rishi Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.
- [8] Vertica Bhardwaj and Ann Fairhurst. Fast fashion: response to changes in the fashion industry. *The international review of retail, distribution and consumer research*, 20(1):165–173, 2010.
- [9] Edward K Blum and Leong Kwan Li. Approximation theory and feedforward networks. *Neural networks*, 4(4):511–515, 1991.
- [10] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G Willcocks. Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. *arXiv preprint arXiv:2103.04922*, 2021.
- [11] Ali Borji. Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2019.
- [12] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

- [13] Otto von Busch. *Fashion-able. Hactivism and engaged fashion design*. School of Design and Crafts; Höskolan för design och konsthantverk, 2008.
- [14] Becca Caddy. Could your next smartwatch replace your phone? <https://www.techradar.com/news/could-a-smartwatch-replace-your-phone>. Accessed: 08-01-2021.
- [15] Xi Chen, Yan Duan, Rein Houthoofd, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: interpretable representation learning by information maximizing generative adversarial nets. In *Neural Information Processing Systems (NIPS)*, 2016.
- [16] Jaewon Choi and Seongcheol Kim. Is the smartwatch an it product or a fashion product? a study on factors affecting the intention to use smartwatches. *Computers in Human Behavior*, 63:777–786, 2016.
- [17] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8789–8797, 2018.
- [18] Martin Christopher, Robert Lowson, and Helen Peck. Creating agile supply chains in the fashion industry. *International Journal of Retail & Distribution Management*, 2004.
- [19] Alan Cramer. Artificial intelligence: The fourth industrial revolution. <https://www.information-age.com/artificial-intelligence-fourth-industrial-revolution-123475170/>. Accessed: 12-01-2021.
- [20] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.
- [21] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.
- [22] Haoye Dong, Xiaodan Liang, Xiaohui Shen, Bochao Wang, Hanjiang Lai, Jia Zhu, Zhiting Hu, and Jian Yin. Towards multi-pose guided virtual try-on network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9026–9035, 2019.
- [23] Rinon Gal, Dana Cohen, Amit Bermano, and Daniel Cohen-Or. Swagan: A style-based wavelet-driven generative model. *arXiv preprint arXiv:2102.06108*, 2021.
- [24] Yuying Ge, Ruimao Zhang, Lingyun Wu, Xiaogang Wang, Xiaoou Tang, and Ping Luo. A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images. *CVPR*, 2019.
- [25] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O’Reilly Media, 2019.
- [26] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.
- [27] 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.

- [28] Simone Guercini. Relation between branding and growth of the firm in new quick fashion formulas: analysis of an italian case. *Journal of Fashion Marketing and Management*, 5(1):69–79, 2001.
- [29] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5769–5779, 2017.
- [30] Xintong Han, Zuxuan Wu, Zhe Wu, Ruichi Yu, and Larry S Davis. Viton: An image-based virtual try-on network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7543–7552, 2018.
- [31] Karen Hao. The ai technique that could imbue machines with the ability to reason. <https://www.technologyreview.com/2019/07/12/65579/the-next-ai-revolution-will-come-from-machine-learnings-most-underrated-form/>. Accessed: 12-01-2021.
- [32] 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.
- [33] 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 *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6629–6640, 2017.
- [34] Junshi Huang, Rogerio S Feris, Qiang Chen, and Shuicheng Yan. Cross-domain image retrieval with a dual attribute-aware ranking network. In *Proceedings of the IEEE international conference on computer vision*, pages 1062–1070, 2015.
- [35] 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.
- [36] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- [37] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.
- [38] Yamini Jhanji. 11 - computer-aided design—garment designing and patternmaking. In Rajkishore Nayak and Rajiv Padhye, editors, *Automation in Garment Manufacturing*, The Textile Institute Book Series, pages 253 – 290. Woodhead Publishing, 2018.
- [39] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.
- [40] Sue Jenkyn Jones. *Fashion design*. Laurence King Publishing, 2005.
- [41] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

- [42] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410, 2019.
- [43] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020.
- [44] Hee-Su Kim and Sung-Bae Cho. Application of interactive genetic algorithm to fashion design. *Engineering applications of artificial intelligence*, 13(6):635–644, 2000.
- [45] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [46] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.
- [47] Karol Kurach, Mario Lucic, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. The GAN landscape: Losses, architectures, regularization, and normalization, 2019.
- [48] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
- [49] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.
- [50] Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.
- [51] Jae Hyun Lim and Jong Chul Ye. Geometric gan. *arXiv preprint arXiv:1705.02894*, 2017.
- [52] Linlin Liu, Haijun Zhang, Xiaofei Xu, Zhao Zhang, and Shuicheng Yan. Collocating clothes with generative adversarial networks cosupervised by categories and attributes: a multi-discriminator framework. *IEEE transactions on neural networks and learning systems*, 31(9):3540–3554, 2019.
- [53] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [54] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. *arXiv preprint arXiv:1711.10337*, 2017.
- [55] Liqian Ma, Xu Jia, Qianru Sun, Bernt Schiele, Tinne Tuytelaars, and Luc Van Gool. Pose guided person image generation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 406–416. Curran Associates, Inc., 2017.
- [56] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017.

- [57] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR, 2018.
- [58] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.
- [59] Matur Rahman Minar, Thai Thanh Tuan, Heejune Ahn, Paul Rosin, and Yu-Kun Lai. Cpvton+: Clothing shape and texture preserving image-based virtual try-on. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.
- [60] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.
- [61] Tom Mitchell. Introduction to machine learning. *Machine Learning*, 7:2–5, 1997.
- [62] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.
- [63] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.
- [64] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.
- [65] Mustafid, Sarrah Ade Karimariza, and Ferry Jie. Supply chain agility information systems with key factors for fashion industry competitiveness. *International Journal of Agile Systems and Management*, 11(1):1–22, 2018.
- [66] Christopher Olah. Neural networks, manifolds, and topology. *Blog post*, 2014.
- [67] Albert Pumarola, Antonio Agudo, Alberto Sanfeliu, and Francesc Moreno-Noguer. Unsupervised person image synthesis in arbitrary poses. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8620–8628, 2018.
- [68] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.
- [69] Amit Raj, Patsorn Sangkloy, Huiwen Chang, James Hays, Duygu Ceylan, and Jingwan Lu. Swapnet: Image based garment transfer. In *European Conference on Computer Vision*, pages 679–695. Springer, 2018.
- [70] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017.
- [71] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.
- [72] Mihaela Rosca, Balaji Lakshminarayanan, David Warde-Farley, and Shakir Mohamed. Variational approaches for auto-encoding generative adversarial networks. *arXiv preprint arXiv:1706.04987*, 2017.

- [73] Negar Rostamzadeh, Seyedarian Hosseini, Thomas Boquet, Wojciech Stokowiec, Ying Zhang, Christian Jauvin, and Chris Pal. Fashion-gen: The generative fashion dataset and challenge. *arXiv preprint arXiv:1806.08317*, 2018.
- [74] Ruth Rubinstein. *Dress codes: Meanings and messages in American culture*. Routledge, 2018.
- [75] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.
- [76] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.
- [77] Stuart Russel, Peter Norvig, et al. *Artificial intelligence: a modern approach*. Pearson Education Limited, 2013.
- [78] Tim Salimans, Ian J Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, 2016.
- [79] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.
- [80] Sabine Seymour. *Fashionable technology: The intersection of design, fashion, science, and technology*. Springer, 2008.
- [81] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [82] Montek Singh, Utkarsh Bajpai, V Vijayarajan, and Surya Prasath. Generation of fashionable clothes using generative adversarial networks. *International Journal of Clothing Science and Technology*, 2019.
- [83] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.
- [84] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [85] 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.
- [86] Md Tabraz. Importance of fashion cad (computer aided design) study for garment industry in bangladesh. *International Journal Of Scientific & Technology Research*, 6, 2017.
- [87] Dana Thomas. *Deluxe: How luxury lost its luster*. Penguin, 2007.
- [88] Rob Thomas. How ai is driving the new industrial revolution. <https://www.forbes.com/sites/ibm/2020/03/04/how-ai-is-driving-the-new-industrial-revolution/>. Accessed: 12-01-2021.

- [89] Rob Thomas. A simple explanation of the inception score. <https://medium.com/octavian-ai/a-simple-explanation-of-the-inception-score-372dff6a8c7a/>. Accessed: 27-01-2021.
- [90] Bochao Wang, Huabin Zheng, Xiaodan Liang, Yimin Chen, Liang Lin, and Meng Yang. Toward characteristic-preserving image-based virtual try-on network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 589–604, 2018.
- [91] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.
- [92] Yaxing Wang, Chenshen Wu, Luis Herranz, Joost van de Weijer, Abel Gonzalez-Garcia, and Bogdan Raducanu. Transferring gans: generating images from limited data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 218–234, 2018.
- [93] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.
- [94] Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer, 1992.
- [95] Zhonghua Wu, Guosheng Lin, Qingyi Tao, and Jianfei Cai. M2e-try on net: Fashion from model to everyone. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 293–301, 2019.
- [96] Yuanbo Xiangli, Yubin Deng, Bo Dai, Chen Change Loy, and Dahua Lin. Real or not real, that is the question. In *International Conference on Learning Representations*, 2019.
- [97] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.
- [98] Gokhan Yildirim, Nikolay Jetchev, Roland Vollgraf, and Urs Bergmann. Generating high-resolution fashion model images wearing custom outfits. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.
- [99] Donggeun Yoo, Namil Kim, Sunggyun Park, Anthony S Paek, and In So Kweon. Pixel-level domain transfer. In *European Conference on Computer Vision*, pages 517–532. Springer, 2016.
- [100] Egor Zakharov, Aliaksandra Shysheya, Egor Burkov, and Victor Lempitsky. Few-shot adversarial learning of realistic neural talking head models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9459–9468, 2019.
- [101] Haijun Zhang, Yanfang Sun, Linlin Liu, Xinghao Wang, Liuwu Li, and Wenyin Liu. Clothingout: a category-supervised gan model for clothing segmentation and retrieval. *Neural computing and applications*, 32(9):4519–4530, 2020.
- [102] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019.

- [103] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiao lei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5907–5915, 2017.
- [104] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.
- [105] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. *arXiv preprint arXiv:2006.10738*, 2020.
- [106] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.
- [107] Shizhan Zhu, Raquel Urtasun, Sanja Fidler, Dahua Lin, and Chen Change Loy. Be your own prada: Fashion synthesis with structural coherence. In *Proceedings of the IEEE international conference on computer vision*, pages 1680–1688, 2017.