

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Automatic generation of unique, custom and stylish clothes for fashion industries

Catarina de Almeida Figueiredo



Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Rui Maranhão

Co-Supervisor: Abubakar Zakari

October 21, 2021

Automatic generation of unique, custom and stylish clothes for fashion industries

Catarina de Almeida Figueiredo

Mestrado Integrado em Engenharia Informática e Computação

October 21, 2021

Abstract

Every year, the fashion industry spends a big part of its revenue creating new clothing collections. Traditionally, the fashion industry produces four clothing collections per year (*i.e.*, Spring/Summer, Fall/Winter, Resort, and Pre-Fall). In recent years, the usual four clothing collections per year were replaced by 52 micro-collections, representing an increased investment in the creative process. The fashion industry is one of the most polluting in the world in an era where sustainability is becoming increasingly important to consumers. A straightforward way to minimize waste is to guarantee that the correct quantity of each piece of clothing is produced and ensure that it meets customer expectations. Furthermore, the integration of sustainability in the industry will come with additional costs. In order to maintain market leadership, the industry can reduce the costs of the design creation stage with the integration of Artificial Intelligence (AI) models.

In this dissertation, we will focus on applying AI in clothes generation to enable fashion organizations to reduce their costs, maximize their sales, and consequently reduce unsold clothing and the environmental impact of the design process. Two separate models are proposed to improve and assist designers during the design creation phase. The first model automatically generates garments based on specific features (*e.g.*, type of garments). This model enables fashion designers to create complete collections of clothing in a matter of minutes. The second proposed model allows designers to convert fashion sketches into real garments. This way, designers can make minor changes in the sketches, alter the fabrics, and see the resultant garment almost in real-time. In other words, the model converts fashion sketches with overlaid texture patches into realistic garments.

The proposed models were both evaluated using performance metrics that assessed the quality of the generated images. The first model was evaluated using the performance metric Fréchet Inception Distance (FID). The model was able to generate suitable images from different types of garments. The second model was evaluated using the performance metric Structural Similarity Index Measure (SSIM). The metric pointed out to high-quality generated images almost identical to the ground truth images. A user survey was conducted to evaluate the realness and marketability of the garments generated by the second proposed model. The user survey results indicate that the generated garments are very similar to real garments and that customers are most likely to buy them.

Keywords: Neural Networks, GANs, Fashion Design, Garment Generation, Garment Translation, Image-to-image Translation, Artificial Intelligence, Deep Learning

Resumo

Anualmente, a Indústria da Moda gasta grande parte das suas receitas na criação de novas coleções de vestuário. Nos últimos anos verificou-se uma mudança de uma produção de 4 coleções para 52 micro coleções de vestuário por ano, o que representa um aumento significativo do investimento no processo criativo das coleções. A Indústria da Moda é uma das mais poluentes do mundo numa era em que a sustentabilidade está a ganhar importância. Grande parte dos consumidores preferem empresas que tenham princípios de sustentabilidade e que tentem reduzir o desperdício ao máximo. Para minimizar o desperdício, uma solução possível é garantir que a quantidade correta de cada peça de vestuário é produzida e assegurar que esta satisfaz as expectativas do cliente. A integração da sustentabilidade na indústria implicará custos adicionais. Com o intuito de manter a liderança do mercado, a indústria pode reduzir os seus custos com a integração de modelos de inteligência artificial (IA) na criação das coleções de vestuário.

Nesta dissertação iremos usar modelos de IA para a geração de peças de roupas, isto irá permitir que as organizações da indústria da moda reduzam os seus custos, minimizem o impacto ambiental do processo de design, maximizem as vendas e consequentemente reduzam a quantidade de roupa não vendidas. Foram implementados dois modelos distintos com o objetivo de auxiliar os designers durante a criação das coleções de roupa. O primeiro modelo gera automaticamente peças de roupa com base em alguma característica (e.g., tipo de peça de roupa). Este modelo permite aos designers de moda criarem coleções de roupa em poucos segundos. O segundo modelo permite aos designers transformarem os seus esboços de moda em peças de vestuário. Desta forma, os designers podem fazer pequenas alterações nos esboços, alterar os tecidos que pretendem usar, e ver a peça de vestuário resultante em tempo real.

Os modelos implementados foram ambos avaliados utilizando métricas de avaliação da qualidade das imagens geradas. O primeiro modelo foi avaliado utilizando a métrica de desempenho Fréchet Inception Distance (FID). O modelo foi capaz de gerar imagens realistas de diferentes tipos de peças de vestuário. O segundo modelo foi avaliado utilizando a métrica Structural Similarity Index Measure (SSIM). O valor da métrica veio reforçar que as imagens geradas pelo modelo são de grande qualidade e muito realistas. Efetuou-se um questionário de forma a avaliar a percepção visual das imagens geradas e de forma a averiguar se seriam ou não vendáveis. Os resultados do questionário indicaram que as peças de vestuário geradas são muito semelhantes às peças de vestuário reais e que é bastante provável existirem clientes a quererem, adquirir as peças de vestuário geradas.

Keywords: Redes Neurais, GANs, Design de Moda, Geração de Roupa, Tradução de imagem a imagem, Inteligência Artificial, Aprendizagem Profunda

Acknowledgements

Gostaria de agradecer a toda a gente que esteve presente e que me apoiou durante estes 5 longos anos do curso. Sem este apoio nunca teria chegado onde cheguei e não teria atingido os objetivos que pretendia.

Gostaria de fazer um especial agradecimento ao orientador Professor Rui Maranhão pelo apoio dado durante a realização da dissertação. Também queria agradecer ao co-orientador Abubakar Zakari por estar sempre disponível e pronto para ajudar. Gostaria também de agradecer ao CEO Orlando Ribas da XNFY Lab pela oportunidade de estar a trabalhar numa área do meu interesse, numa empresa de inovação com imenso potencial e por todo o apoio que deu nestes últimos meses.

Por fim, mas não menos importante, quero agradecer à minha família e amigos pela preocupação e apoio. A vida tem os seus altos e baixos e agradeço do fundo do coração todo o apoio que me deram. Levo comigo momentos de muita frustração, stress e tristeza, mas também momentos bons e de muitas conquistas.

Quero dedicar este trabalho a quem me acompanhou desde o início o meu percurso nesta faculdade, mas infelizmente não o pode ver terminar. A vida nem sempre segue os nossos planos, mas devemos sempre dar o nosso melhor, por nós, por aqueles que estiveram e por aqueles estão e continuarão ao nosso lado.

Obrigada 🐾,
Catarina Figueiredo

*“Art is unquestionably one of the purest and highest elements in human happiness.
It trains the mind through the eye, and the eye through the mind.
As the sun colours flowers, so does art colour life.”*

John Lubbock

Contents

1	Introduction	1
1.1	Background	1
1.2	Problem Statement	3
1.3	Motivation	3
1.4	Research Questions	5
1.5	Research Objectives	5
1.6	Dissertation Structure	6
2	Background	7
2.1	Artificial Intelligence	7
2.2	Machine Learning	8
2.2.1	Overview	8
2.2.2	Type of learning	9
2.2.3	Reinforcement Learning	10
2.2.4	Evaluation	11
2.3	Deep Learning	12
2.4	Artificial Neural Networks	12
2.4.1	ANNs Components	12
2.4.2	ANNs Training	15
2.5	Convolutional Neural Networks	16
2.5.1	Convolution Layer	17
2.5.2	Pooling Layer	17
2.5.3	Fully Connected Layer	17
2.6	Generative Adversarial Networks	18
2.6.1	GANs Training	19
2.6.2	GANs challenges	20
2.6.3	GAN-based models	20
2.6.4	Evaluation	21
3	Literature review	25
3.1	Image Generation	25
3.1.1	Pixel-wise	25
3.1.2	ProgressiveGAN	25
3.1.3	StyleGAN	26
3.1.4	SpatialGAN	27
3.2	Image-to-Image Translation	28
3.2.1	Pix2pix	28
3.2.2	CycleGAN	28

3.3	GANs for Fashion	29
3.3.1	Garment Matching Design	29
3.3.2	Garment Translation	29
3.3.3	Garment Attribute Manipulation	31
3.3.4	Garment Extraction	31
3.3.5	Garment Transfer	31
3.3.6	Garment Generation	32
3.4	Fashion Datasets	34
4	Methodology	37
4.1	Research Process	37
4.2	Proposed Models	38
4.3	Conditional Garment Generation model	39
4.3.1	Generator	40
4.3.2	Discriminator	44
4.3.3	Regularization Methods	45
4.3.4	Sampling Techniques	47
4.3.5	Loss Functions	49
4.4	Pattern Transfer model	49
4.4.1	Generator	50
4.4.2	Discriminator	54
4.4.3	Loss Functions	54
5	Experimentation	57
5.1	Data collection	57
5.2	Libraries Used	58
5.3	Experiments	58
5.3.1	Experiment 1: Conditional garment generation	58
5.3.2	Experiment 2: Pattern Transfer	60
5.4	User Survey	63
6	Results and Discussion	65
6.1	Experiment 1: Conditional garment generation	65
6.2	Experiment 2: Pattern Transfer	70
6.3	User Survey	75
6.3.1	Section I - Visual perception of generated garments	75
6.3.2	Section II - Identify the generated dress	75
6.3.3	Section III - Fashion sketch to dress I	76
6.3.4	Section IV - Fashion sketch to dress II	77
6.3.5	Section V - Evaluating the marketability of the generated garments	79
7	Conclusion and Future Work	81
7.1	Research Findings	81
7.2	Research Contributions	83
7.3	Threats to Validity	84
7.4	Challenges	84
7.5	Future Research	85
7.6	Final Remarks	85

<i>CONTENTS</i>	xi
A User Survey	87
References	97

List of Figures

2.1	Artificial Intelligence sub-fields. Figure adapted from [11]	8
2.2	Traditional Programming approach vs. Machine Learning approach. Figure extracted from [38]	9
2.3	K-Means clustering algorithm. Figure extracted from [13]	11
2.4	Artificial Neural Network	13
2.5	Single neuron in an Artificial Neural Network	14
2.6	Convolution neural network Architecture [48]	16
2.7	Convolution operational with a kernel size of 3x3. Figure extracted from [47]	17
2.8	GAN Architecture	18
3.1	Images generated by StyleGAN [64]	26
3.2	Images generated by StyleGAN2 [44]	26
3.3	Images generated by VOGUE given a target person and a specific top garment [32]	27
3.4	Image generated by DISCO-GAN[60]	29
3.5	Image generated by TextureGAN [68]	30
3.6	Image generated by AMGAN [1] when modifying the sleeve length attribute	31
3.7	Image generated by CascadeGAN [73] and CatGAN [72] when extracting a blazer from a model	32
3.8	Images generated by GarmentGAN [50] and CP-VTON[65]	33
3.9	Images generated by ProgressiveGAN [52]	34
3.10	Garments generated by cGAN [40] on Fashion MNIST dataset. Images extracted from [8]	35
3.11	Garment generated by e-AttnGAN [2] on Fashion-Gen dataset	35
4.1	Condition Garment Generation model	38
4.2	Pattern Transfer model	38
4.3	Overview of the Condition Garment Generation model components	39
4.4	Generator Networks of the CGG model	40
4.5	Mapping network architecture. Diagram partially adapted from [64]	41
4.6	Synthesis network architecture for 128x128 of resolution	42
4.7	Architecture of the Style Block from the Synthesis Network	43
4.8	Architecture of the ToRGB Block from the Synthesis Network	43
4.9	Discriminator network for 128x128 image resolution	44
4.10	Architecture of the Discriminator block from the Discriminator of the CGG model	45
4.11	Discriminator Epilogue architecture	46
4.12	Data augmentation applied to a garment image	47
4.13	Representation of the LAB color space. Figure extracted from [12]	50
4.14	Overview of the Generator 5-channel input	50

4.15	Overview of the Scribbler architecture	51
4.16	Scribbler Residual block architecture	52
4.17	Overview of TextureGAN Generator architecture	52
4.18	Comparison of two convolution operations with different dilation rates.	53
4.19	Overview of Scribbler Dilate 128 Generator architecture	53
4.20	Discriminator network from the Pattern Transfer model	54
4.21	Overview of the Pattern Transfer model architecture	55
5.1	Sample of images from the multi-garment dataset used for the training of the CGG model	60
5.2	Sample of images from the dresses paired dataset used to train the Pattern Transfer model	61
5.3	Images obtained using the pre-trained version of the DexiNed Model	62
6.1	Generated images collected throughout the training from the CGG model.	66
6.2	FID metric per training epoch for the labels Skirts, Trousers and Dresses	67
6.3	Different types of garments generated throughout the training from a common latent vector	68
6.4	FID metric per training epoch for Whole Body, Lower Body and Upper Body garments	69
6.5	Garments generated by Scribbler, TextureGAN and Scribbler dilate 128 generators from the training runs with the highest performance	71
6.6	Pattern Transfer results for the training run 3 model using single colored dress	72
6.7	Pattern Transfer results which failed in applying the color constraints	73
6.8	Pattern Transfer results for the training run 3 model using dresses with patterns	73
6.9	Pattern Transfer results that didn't propagate correctly the textures through the dress	74
6.10	SSIM metric per training epoch for the training run 3 model	74
6.11	Generator Loss per training epoch in training run 3	74
6.12	User survey answers for Section I	75
6.13	User survey answers for multiple-choice questions of Section IV, (a) Question 1, (b) Question 2, (c) Question 3	78
6.14	User survey answers distribution for questions of Section V	79

List of Tables

5.1	Number of images per label	59
5.2	Dataset csv file format	59
5.3	Hyperparameters changed between the training run of the proposed Pattern Transfer model	63
6.1	Hyperparameters changed between the training run of the proposed Pattern Transfer model	71
6.2	Percentage of correct answers for Section II multiple choice questions	76
6.3	Percentage of correct answers for Section III multiple choice questions	76
6.4	Percentage of correct answers for Section IV multiple choice questions	77

Abbreviations

AI	Artificial Intelligence
ANNs	Artificial Neural Networks
cGAN	Condicional GAN
CGG	Conditional Garment Generation
CNNs	Convolutional Neural Networks
DCGAN	Deep Convolutional GAN
DL	Deep Learning
EMA	Exponential Moving Average
FID	Fréchet Inception Distance
GANs	Generative Adversarial Networks
GDP	Gross Domestic Product
IS	Inception Score
KL	Kullback-Leibler
Leaky ReLU	Leaky Rectified Linear Unit
MAE	Mean Absolute Error
ML	Machine Learning
MLP	Multilayer Perceptron Network
ReLU	Rectified Linear Unit
RMSE	Root Mean Square Error
SSIM	Structural Similarity Index Measure
Tanh	Hyperbolic Tangent

Chapter 1

Introduction

Machine learning is constantly evolving. Proof of this is the emergence of new and improved models that are gradually being used to assist humans in fulfilling specific tasks. In the last few years, the acceptance rate of the integration of machine learning models in the industry sector has been steadily growing. Integrating Artificial Intelligence (AI) in the fashion industry, especially in the creative field, can be highly beneficial. The emergence of Fast Fashion has turned the fashion industry into a very competitive and unstable industry. With the integration of AI, the industry could align sustainability, which is becoming extremely important for consumers, with faster speed to market, fewer losses, and more profit.

1.1 Background

The fashion industry is one of the world's largest industries, and it is devoted to the business of designing, making, and selling garments. The arrival of new technologies such as sewing machines and new mass-production methods enabled the term fashion industry to arise [67]. In the late nineteenth century, there was a transition from producing handmade clothing for individuals to the mass-production of clothing with standard sizes for the mass. With the arrival of mass production, garments became cheaper and accessible for the overall population, which led to massive growth in the industry.

Over the past few years, the industry has been adapting and evolving. The integration of new technologies led to a greater manufacturing capacity, shorter production time, and a continuous accentuated growth.

Fast Fashion is a phenomenon in the fashion industry where the entire manufacturing process is accelerated to bring new trends to market faster and at a lower cost to meet customer demand. Clothing collections are inspired by the latest trends, which can change quickly and are very difficult to follow [63]. Fast Fashion companies offer clothes inspired by upcoming trends, cheaper and

similar to expensive designer garments at an impressive speed [63]. These companies allow consumers to wear the garments when they are at the peak of their popularity. Fast fashion garments have a short usage time since the materials used are cheaper and of poorer quality. Therefore, to provide cheaper clothing items, the garments produced by fast fashion companies are less durable and of lower quality which contributes to further waste and consumerism. As the industry grows, so does the impact it has on the environment. The Fashion Industry is responsible for 10% of the global CO₂ emissions, which is more than the global air-shipping traffic combined [21]. Fast Fashion covers the entire lifecycle from design to manufacture and marketing. It is growing very fast and has risen to become a large part of the overall fashion retailing. Fast Fashion opposes the traditional fashion cycle as it captures current fashion trends and puts them on the market as faster as possible. Consumers want to keep on with the current trends, and Fast Fashion enables them to look stylish at an affordable price [19]. The fashion industry comprises four levels: production of raw materials, production of fashion goods (which goes from the design to the manufacturing of collections), retail sales, and advertising. All phases are carried out separately, but they are inter-dependent as their common goal is to satisfy consumer demand in conditions that allow companies to have profit.

Technology has become a crucial part of the Fashion and apparel industry, and slowly the industry is rising its acceptance rate. It is essential to absorb and integrate current technological innovations to reduce manufacturing costs and, consequently, sell more at a lower price. A key to achieving success in the industry is to meet customer demand. Artificial Intelligence (AI) methods have been incrementally used in the last few years and significantly improved the fashion industry's response to customer demand. AI is being used in the fashion industry in trend forecasting, market performance analysis, and product recommendation. AI techniques allow to achieve great results, so one can say that AI boosts a great competitive advantage for fashion industries. AI has the potential to revolutionize the fashion industry by speeding the production process while reducing operational costs. The advances in computer vision tasks enabled AI to perform tasks that typically require human perception. Computer vision tasks as image recognition or image generation can be successfully addressed through a subset of AI models named Deep Learning models. The arrival of Generative Adversarial Networks (GANs) in 2014 boosted the development of new and improved image generation models. Furthermore, it also potentiated the development of models for other computer vision tasks such as image-to-image translation.

Deep Learning models can be applied in the design creation process. Imitating a designer is now a topic with great potential. Computers algorithms can mimic the design process knowing past designs and future trends. AI-designed garments can be used as inspiration for designers or as the finished design, significantly reducing the time spent on the design process and improving customer satisfaction. On the other hand, Deep Learning models can also automatically transform fashion sketches made by designers into real garments. This way, the designers could visualize the garments before their production.

1.2 Problem Statement

The industry's primary goal is to reduce the time-to-market as much as possible and present the new trends to consumers first-hand. It is a competition between who can accomplish it first. The first to do it gets the market opportunity. Over the years, through the use of automated processes, new mass-production techniques, and the growth of e-commerce, fast fashion time-to-market has been dramatically reduced [63].

The fashion industry is very volatile, and companies must be able to at least keep up with the pace of the industry and adapt. Those who cannot keep up will suffer heavy financial losses. This industry is facing a time of transition where it is very important to keep up with the rhythm of fast fashion while moving towards sustainability. Therefore, companies have to find a middle term between these two antagonist factors. While fast fashion is characterized by huge waste and pollution, sustainability focuses on the recycling of fabrics and on reducing the environmental impact that the fashion industry was on the world. Two important considerations: the fashion industry is one of the most polluting industries in the world; sustainability is becoming increasingly important to consumers and is a crucial factor when it comes to buying. A rising percentage of consumers prefer companies that aim to reduce waste to the minimum. A straightforward way to minimize waste is to guarantee that the right quantity of each piece of clothing is produced and ensure that it meets the customer's expectations. The awareness is growing and changing customer's preferences when buying clothes. Companies will prosper if they manage to combine fast fashion time-to-market with sustainability principles.

Integrating sustainability in the industry will come with additional costs. Most of the companies that are integrating sustainability are increasing the final price of the products. The retail prices increase because the cost of production and materials also increases. In order to address this problem and maintain market leadership, the industry needs to reduce the costs somewhere else. If the costs of other phases are reduced, the final product prices will probably not increase as much. This cost reduction can be achieved in the design creation phase by integrating AI models that can help and assist designers in creating clothing collections. This way, the amount of time spent in the design creation phase will be reduced.

Currently, only a few AI methods are used to assist designers in this phase. Creating AI models capable of automatically generating complete clothing collections and converting fashion sketches into realistic garments would be of great impact to the Fashion Industry.

1.3 Motivation

The fashion industry keeps growing with a market value of 3 trillion dollars and 2% of the world's Gross Domestic Product (GDP) [17]. It is an industry with much space to grow. Every year, the fashion industry spends a big part of its revenue creating new clothing collections. To succeed, keep their competitive position and profit, fashion industries should always keep up with the current trends and produce clothing collections that fulfill customer demands.

In the last years, fashion industries have shifted from 4 seasons to 52 micro-seasons per year. In other words, instead of designing four clothing collections per year (i.e., Spring/Summer, Fall/Winter, Resort, and Pre-Fall [58]), fifty two micro-collections are designed, representing an increased investment in the creative process. The entire process of designing a collection must be as fast as possible to allow consumers to follow the trends when they are at their peak. The exponential growth of e-commerce in the latter year allowed even more cheap clothes and faster time-to markets. The more production capacity a company has, the cheaper the garments and the faster it can put them in the market. By speeding up the whole process, there is less time to detect errors. The errors can refer to the wrong choice of fabrics or the wrong selection of designs. Even though this acceleration allows us to increase sales, mistakes can be very costly and should be avoided. Although there are marketing studies about consumers' tastes and preferences, clothes' accurate sales volume is only known when the collection is presented. Hence, garments that are less in line with customer's expectations may exist, and consequently, those garments' sales volume can be quite low. The biggest challenge is that these problems are detected too late, and nothing can be done to correct them and reduce the organization's losses.

Deep Learning is a subset of Machine Learning methods that utilize Artificial Neural Networks to learn relevant information from data. The advent of Generative Adversarial Networks (GANs) [18] in 2014 enabled computers to achieve impressive results in media synthesis. GANs are Deep Learning models that revolutionized the area of image synthesis, thus opening a door for its application in the most diverse areas. This is a great opportunity for fashion industries to be ahead of time, improve their response to customer demand, and increase their revenue.

Integrating deep learning in clothes design generation has many advantages for fashion industry organizations. Using deep learning models can reduce the time to market of new collections, reduce their costs, maximize sales and, consequently, reduce unsold clothing and reduce design process environmental impact. As the time-to-market is reducing and the prices are becoming cheaper, the quality of the garments is also decreasing. These factors have contributed to more items of clothing ending up in land fields. Applying AI in the creative process, namely in garment design, can accelerate the design creation process. Designers then have more time to focus on other details, such as improving the fabrics' quality. Considering that there is a real representation of the garments without them being manufactured, the companies can show them to customers and analyze their sell-ability. This way, companies can produce a more accurate quantity of garments and ensure that fewer of them will be wasted. This acceleration of the creative process can lead to an improvement in the quality of the materials used and, therefore, extend the lifespan of the garments to minimize waste. It is a priority to reduce the environmental impact of the fashion industry. A small change made by many companies can be a significant improvement as a whole.

Since 2014, a lot of research has been made on how to use GANs for image generation [74] [28] [29] and image to image translation. In addition, applying GANs in the fashion industry has been a studied topic due to its potential and advantages. In this dissertation two models are proposed: a Conditional Garment Generation model and a Pattern Transfer model. The first model is capable of generating realistic clothes designs using Generative Adversarial Networks based on

specific conditions. The model will generate garments of different categories (*e.g.*, t-shirts, pants, dresses, among others). The second model, Pattern Transfer model can transform fashion sketches with overlaid texture patches into real garments.

1.4 Research Questions

The main research question of this research is:

Can AI models be used to assist designers during the design creation stage by generating complete clothing collections and transforming fashion sketches into real images of garments?

To be able to research into this question, sub-questions were developed. The sub-questions are the following:

- **Research Question 1** - How is AI being applied in fashion and in computer vision tasks of image generation and image-to-image translation?
- **Research Question 2** - How can ANNs be used to generate garments according to certain conditions automatically?
- **Research Question 3** - How can ANNs be used to turn fashion sketches into real garment images?
- **Research Question 4** - What is the performance of the Conditional Garment Generation model and the Pattern Transfer models in terms of performance metrics and user surveys?

1.5 Research Objectives

Having into consideration the proposed research questions, the following research objectives were formulated:

- **Research Objective 1** - To research the available AI technology applied to fashion and to the tasks of image generation and image-to-image translation.
- **Research Objective 2** - To implement a conditional generative model capable of generating different types of garments based on certain conditions.
- **Research Objective 3** - To implement an image to image translation model that converts fashion sketches into real garment images.
- **Research Objective 4** - Evaluate the performance of both models based on performance metrics and user surveys.

1.6 Dissertation Structure

The present dissertation is divided into seven chapters. This section provides a brief overview of their content.

Chapter 1 - Introduction

The present chapter gives a brief overview of the fashion world. The problem statement and the motivation are presented. Also, the research questions and research objectives formulated for this research work are described.

Chapter 2 - Background

This chapter addresses the related technical notions associated with this dissertation. It presents a brief introduction to artificial intelligence and machine learning general concepts and components. Furthermore, the chapter reviews Deep Learning and explains how Artificial Neural Networks, Convolutions Neural Networks, and Generative Adversarial Networks are structured and their expected behavior.

Chapter 3 - Literature review

The Literature review chapter carefully addresses related work found for the addressed computer vision tasks. There are outlined several AI models developed to be used in different tasks within fashion. Computer vision models for tasks as image generation and image-to-image translation will be covered as well. The fashion datasets discovered will also be mentioned in this chapter.

Chapter 4 - Methodology

This chapter outlines in detail the methodology followed during this research. The chapter details the proposed models and the performance metrics selected to evaluate them. Both models proposed, the Conditional Garment Generative model and the Pattern Transfer model are presented and explained in detail.

Chapter 5 - Experimentation

The experimentation chapter outlines the followed experiment setups and the implementation and evaluation of the two proposed models. The created datasets and technologies used are presented as well.

Chapter 6 - Results

This chapter presents the results obtained for each model through the experiments that were conducted. An evaluation of the results is presented based on the performance metrics chosen and the user surveys conducted.

Chapter 7 - Conclusion

In the Conclusion chapter, it is discussed all the identified challenges and future research topics. All research findings and achievements are carefully detailed.

Chapter 2

Background

This dissertation is integrated into the fields of deep learning and computer vision. This chapter addresses the related technical knowledge associated with this research work which comprises the fields of artificial intelligence, machine learning, and deep learning. The chapter starts with a brief introduction to Artificial Intelligence and a review of machine learning's most important concepts and components. In addition, in this chapter, deep learning and Artificial Neural Networks are discussed. Lastly, it is explained how Convolutions Neural Networks and Generative Adversarial Networks are structured, their expected behavior, and application fields.

2.1 Artificial Intelligence

Artificial Intelligence (AI) approaches allow computers to create autonomous systems that mimic human behavior by making autonomous decisions based on patterns and knowledge obtained through the analysis of existing data. AI can be described as the ability of machines to think and behave like human beings. The utilization of AI has several goals, such as improving day-to-day tasks, forecasting, or upgrading the performance of certain industries [51]. Figure 2.1 shows a representative diagram of the sub-fields of AI.

Machine Learning (ML) is a subset of AI methodologies that, through data analysis, extract data patterns and, with those discovered patterns, is able to predict future data [42]. Machine learning uses what is generally known as "training data" to train the models in order to predict outcomes from new and unseen situations. It is used in a wide range of situations, for example, to predict if a bank client will be able to pay back a loan.

Deep Learning (DL) is a field within machine learning that is characterized by being able to learn without supervision using unstructured and unlabeled data. DL models make use of Artificial Neural Networks (ANNs). ANNs mathematically imitate the human brain and allow computers to perceive, see, and make human-like decisions. By training ANNs on large datasets of images, text, or audio, features and characteristics can be extracted from the data [43].

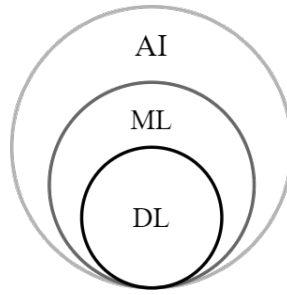


Figure 2.1: Artificial Intelligence sub-fields. Figure adapted from [11]

Deep learning models are mainly used to solve computer vision problems. Computer vision models are becoming increasingly important in the evolution of technology [20]. In the future, it is predicted that computers will replace humans in the majority of tasks that require human perception [61]. Some of the models that have revolutionized computer vision are convolutional neural networks [3] (CNNs), which are normally used for object recognition, and generative adversarial networks [18] (GANs) mainly used in image generation [64] and image-to-image translation [23].

2.2 Machine Learning

ML is comprised of models that learn by experience and are able to make accurate predictions [41]. ML models look for complex patterns in the data in order to know the output of new and unseen situations. ML models allow making better decisions in the future by knowing what worked best in the past. Past examples are analyzed and data patterns are extracted by the models. Through the extracted patterns the models are capable of predicting the outcome of new examples. ML models can learn without any human interaction [41]. Machine learning is essential to data science and is mostly applied in classification, Regression, and clustering problems.

2.2.1 Overview

ML has a different approach compared to traditional programming [11]. It emerged as a solution to rather complex programming problems which were difficult to solve through traditional programming. Problems such as autonomous car driving can involve an extensive amount of rules and conditions that must be satisfied according to traditional programming. Through ML approaches, those conditions would be hidden and inferred by the models, thus arriving at the same result with better performance and simplified algorithms.

In traditional programming, the development of an algorithm requires an initial study of the problem, resulting in the writing of a set of rules. The algorithm is tested with new data, and a result is given. If the result does not correspond with the expected result, the written rules should be revised. The ML approach is different; in this case, there is also a problem analysis and study, but instead of creating rules, past data examples and their expected results are provided to the algorithm. The ML approach is divided into two phases. In the first phase, the algorithm is trained

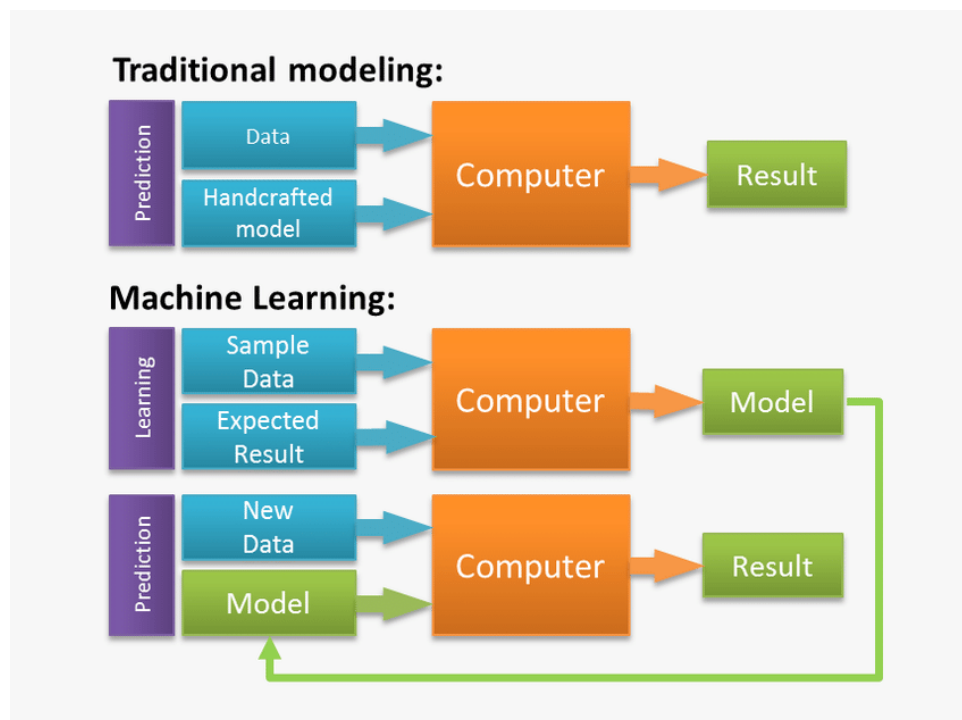


Figure 2.2: Traditional Programming approach vs. Machine Learning approach. Figure extracted from [38]

with past data examples. In the second phase, the model obtained through the training is used to predict new data examples. Figure 2.2 show the differences between both approaches.

2.2.2 Type of learning

ML models can be grouped and classified considering the type of learning process used. There are mainly four types of learning supervision, namely supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning [9] [42]. These four types of learning are described below.

2.2.2.1 Supervised Learning

In supervised learning, the model learns from experience; this means that the models will receive past examples as input and predict with high accuracy the outcome of new and unknown examples. Supervised learning is distinguished by the use of labeled data collected from past examples. The labeled data contains a set of data variables, known as features, and the associated outcomes. The labeled data is fed into the models, and during training, the models assess for correlations and patterns between the data variables and the known outcomes. After training, the model should be able to predict outcomes for new examples [16]. If the problem at hand is *"Will a certain client be able to pay back a given loan?"* the labeled data will include features such as age and income and the outcome variable, whether the person has paid back the loan or not. Based on previous

experience, the model will predict the outcomes of other situations allowing banks to make better decisions regarding which clients they will lend money to [14].

The two most common supervised learning tasks are classification and regression problems. Classification and Regression are described in detail below.

- **Classification problems** goal is to predict with high accuracy a class label for a given example of input data. The labels are known as the dependent variable and, in Classification problems are always nominal variables. The model finds patterns and relations in the training data and is able to classify new and unseen examples [24].
- **Regression problems** goal is to assign and predict new numerical values for unseen examples of data. Regression problems dependent variables are always numerical variables. An example of a regression problem is to predict the sales volume of a given garment. This type of model is used mostly to make estimates and forecasting [25].

2.2.2.2 Unsupervised Learning

Unsupervised machine learning uses unlabelled datasets; the models analyze and group data in the best way possible, with no human intervention. Models can spot complex, hidden patterns in data, learn similarities between data, and cluster them accordingly. The data used only comprises data variables; unlike in supervised learning problems, the dependent variable is not provided. Unsupervised learning can perform more complex tasks than supervised learning. The most common unsupervised learning task is clustering. Clustering is about assigning labels to input examples using unlabeled datasets. When using datasets with no labels, achieving the optimal model becomes more complicated. Clustering splits data into groups according to their similarities. Data points within the same group are more similar to each other and have more similar features. Two of the most widely used cluster algorithms are K-means, a centroid-based algorithm, and DBSCAN, an algorithm whose clusters can have arbitrary shapes. Figure 2.3 shows a graphic representation of the K-mean clustering algorithm.

2.2.2.3 Semi-supervised Learning

Semi-supervised learning is a combination of supervised and unsupervised learning. The datasets used contain both labeled and unlabeled data. In fact, the number of unlabeled examples is usually far higher than the number of labeled ones. The goal of semi-supervised learning is the same as supervised learning. The motto is to use unlabelled data to add more information about the problem and improve the model accuracy.

2.2.3 Reinforcement Learning

Reinforcement learning algorithms learn from trial and error, and unlike supervised learning, they are not trained using training data. A reinforcement agent will determine what steps to do to perform a given task following a reward-based system. When performing a specific behavior, the

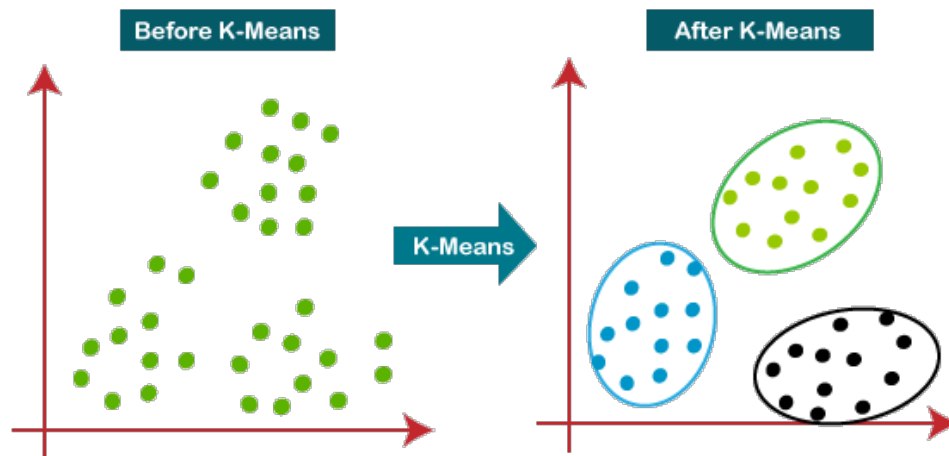


Figure 2.3: K-Means clustering algorithm. Figure extracted from [13]

reinforcement agent will receive a signal concerning the quality of its behavior. If the behavior is the expected one, the agent will be rewarded, and if it is not, it will be penalized. The agent's goal during training is to maximize the sum of the rewards. Throughout the training, the agent will perform better and better at a given task.

2.2.4 Evaluation

The evaluation of Machine learning models is a crucial step. It is important to ensure that the model is not overfitting and has a good performance before deployment. Overfitting is a common machine learning problem, and it happens when the model overfits the training dataset. If overfitting exists, the model will perform excellent with examples from the training data but will perform terribly with unseen examples. If the model's performance is far greater with the training data than with the test data, the model is probably overfitting.

There are different ways to evaluate machine learning models, the most common holdout evaluation. In this technique, the complete dataset is divided into two subsets; the training set used to build the predictive model and the test set. The test set will be used to evaluate the model and assess the model's performance in the future. Using a test set is important to check if the model is overfitting the training data. Overfitting is probably occurring if the model performs much better on the training data than on the test data. A drawback of this method is the fact that the subset division highly influences predictive performance. Other evaluation methods attempt to overcome this problem by creating several partitions from the original data, each partition with a test and training set. The predictive performance is calculated as the average of the predictive performance of all the partitions. This way, a more reliable predictive performance can be achieved. The most commonly used evaluating methods are random subsampling, k-fold cross-validation, leave-one-out, and bootstrap [25].

The metric chosen to evaluate a model's performance is highly dependent on the problem and machine learning task that is being evaluated. The most common metric to evaluate classification problems is accuracy, which is the ratio of all correct predictions. In unbalanced classification problems, a high accuracy value does not mean good performance, so other metrics should be used. One of them is the F1 measure; this measure is more reliable as it combines both precision and recall metrics [24]. In regression problems, the most used metrics are the Mean Absolute Error (MAE) between the model predictions and the ground truth, and the Root Mean Squared Error (RMSE).

2.3 Deep Learning

Deep learning is a subset of machine learning methods that utilize ANNs to learn patterns from data. It is growing very fast and being applied in many fields such as industry, medicine, finance, and others [51].

Unlike ML that uses algorithms to learn information about the data provided and make informed decisions based on that acquired knowledge, deep learning uses the ANNs to learn and make intelligent decisions on its own.

Deep learning models can perform tasks that in the past only could be accomplished with the help of the human's visual and hearing system. Its models are usually applied in areas such as computer vision and natural language processing.

Computer vision's objective is to make computers see the world as humans do, enabling them to do tasks that necessarily need aid from the human's visual system. Some tasks of computer vision include image generation, analysis, and recognition. Those tasks focus on extracting patterns from real data, namely images, and making the computer interpret them correctly.

Over the years, the interest to make a computer perform tasks that only humans could do due to its potential and economic advantages surfed a considerable growth, and a wide range of deep learning methods and models have emerged with it. Convolutions neural networks (CNN) and generative adversarial networks (GAN) are two of the most used computer vision models.

2.4 Artificial Neural Networks

One of the fundamental parts of deep learning are ANNs. ANNs mimic the behaviors of human brains. Their structure is inspired by the structure of the human brain, and their primary goal is to replicate the way humans learn. ANNs allow computers to detect more complex patterns in data and solve complex problems. ANNs are mainly used for Natural language processing and computer vision tasks such as image recognition and image generation.

2.4.1 ANNs Components

ANNs are networks composed of layers of nodes called neurons; the neurons are the core process unit of ANNs. ANNs contains an input layer, one or more hidden layers, and an output layer.

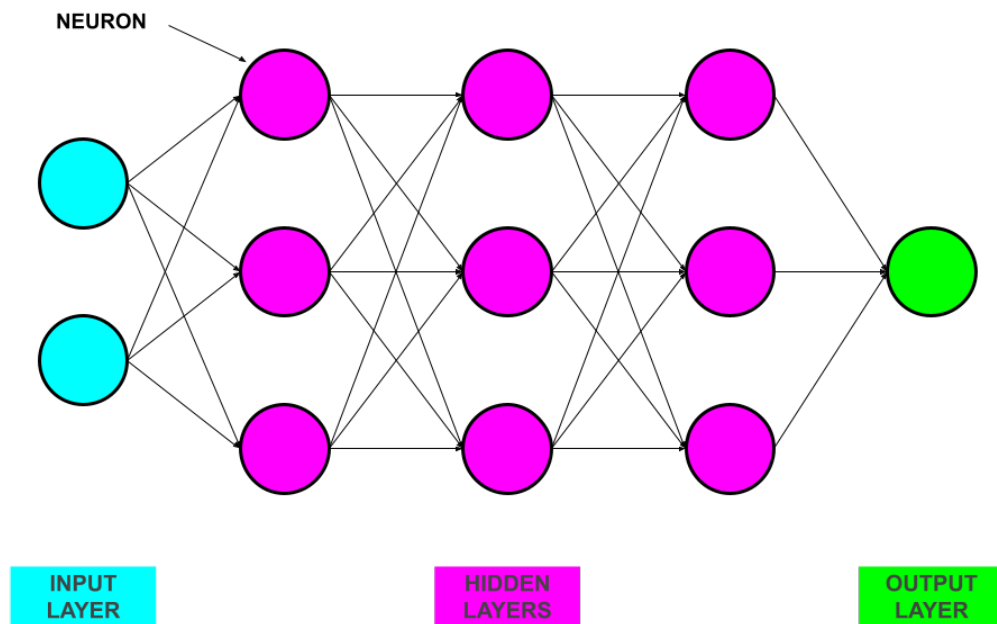


Figure 2.4: Artificial Neural Network

The input layer is responsible for receiving and processing the model's inputs. The output layer predicts the final outputs of the network and can be comprised of a single output or a vector of outputs. In between these two layers are what we call hidden layers. Hidden layers perform most of the computation required by the ANNs and are responsible for the majority of the learning of ANNs. Figure 2.4 provides a visual representation of an ANNs with three hidden layers.

Each layer of neurons is connected to the neurons of the following layer and has an associated weight w and a threshold value. Each neuron is provided with input data and produces an output that will be sent to the neurons of the following layer. This mathematical operation is divided into three separate steps. First the inputs of each neuron x_i are multiplied by a weight w_i . The weights of each neuron x_i are not fixed values and will change during training. Next, all weighted inputs of a given neuron are added together with a bias value b , as illustrated in Figure 2.5. The next step is to pass the sum through an activation function f . The output value of the neuron y will be the value returned by the activation function. The activation function operation is defined in equation 2.1. If the value is bigger than a given threshold value, the neuron will be activated and pass data to the next layer. This is called forward propagation, and this way, the data is propagated throughout the entire network.

Activation functions are used to normalize the output of each neuron from an unbounded value to a nice, predictable value. They can be divided into two different types, linear and non-linear

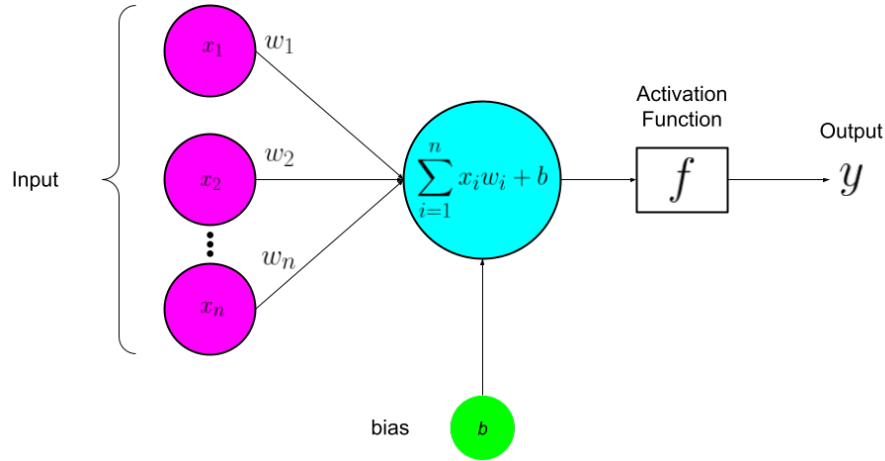


Figure 2.5: Single neuron in an Artificial Neural Network

activation functions. In linear activation functions, the outputs will form a line, so the output values will not be confined within a certain range. On the other hand, non-linear activation functions are confined within a specific range of values and are the most widely used [11] [71].

$$y = f\left(\sum_{i=1}^n x_i w_i + b\right) \quad (2.1)$$

Some of the most used non-linear activation functions are:

- **Sigmoid**

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

- **Rectified Linear Unit (ReLU)**

$$f(x) = \max(0.0, x) \quad (2.3)$$

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

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ 0.01x & \text{otherwise.} \end{cases} \quad (2.4)$$

- **Hyperbolic Tangent (Tanh)**

$$f(x) = \tanh(x) \quad (2.5)$$

The activation functions of the hidden layers and the output layers usually are different from each other. The activation function of the output layer depends on the type of output we want and the problem at hand. The most commonly used activation function in the output layer is the softMax activation function. This function can predict multinomial probability distributions and, because of that, is normally used for multi-class classification problems. The sum of the

probability distributions will always be one. Softmax activation function is defined in equation 2.6.

$$f(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^n \exp(x_j)} \quad (2.6)$$

Knowing that x is the input vector of the softmax activation function in this equation:

- x_i is the i -th element of the input vector x .
- $\exp(x_i)$ is the exponential function applied to x_i .
- $\sum_{j=1}^n \exp(x_j)$ is a normalization term, ensures that the sum of all $f(x)_i$ is 1.

2.4.2 ANNs Training

The Training of ANNs can be divided into three major components. The following sections explain the Loss functions, the Backpropagation algorithm, and the Optimization methods required to train ANNs.

2.4.2.1 Loss functions

To train ANNs, the use of a loss function is always mandatory. The loss function will compare the predicted outputs with the ground truth values, measuring how different they are. The goal in training is to minimize the loss of the neural network, ensuring better predicting performance. The most common loss functions used are:

- **Mean Squared Error Loss** is more appropriate for regression problems. It calculates the average overall squared errors.

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_{true} - y_{pred})^2 \quad (2.7)$$

- **Cross-Entropy Loss** is indicated for classification problems. In binary classification problems, p is the predicted probability observation and y is a binary indicator.

$$Loss = -(y \log(p) + (1 - y) \log(1 - p)) \quad (2.8)$$

2.4.2.2 Backpropagation Algorithm

A crucial step of ANNs training is the backpropagation algorithm [54]. This algorithm was introduced in 1986 and allowed ANNs to fine-tune their weights throughout the training. Backpropagation can be divided into two phases:

- The error calculated with the loss function is feeding backwards into the network.
- An optimization method is used to update the weights to new values that will probably lead to a smaller loss in the next iteration.

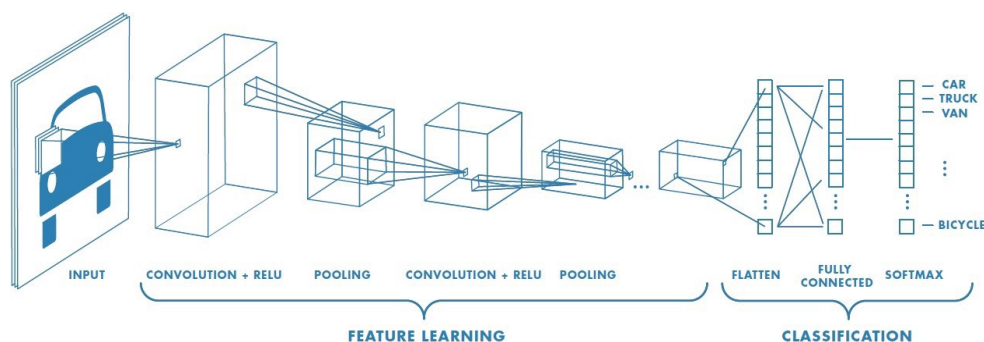


Figure 2.6: Convolution neural network Architecture [48]

2.4.2.3 Optimization Methods

The most used optimization algorithms are gradient descent based. Gradient descent calculates the gradient of the loss function, considering all weights and parameters of the network. The gradient gives us the directions to reach the higher loss values more quickly. Therefore the negative of the gradient is used in order to minimize the losses. Gradient descent updates each parameter weight w_i in the direction that may reduce the total loss.

2.5 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a deep learning model that achieves better results than a neural network when receiving input data with a known grid-type topology. CNNs can be used to classify input images in a particular category. They are mostly used in computer vision tasks such as face recognition, image classification, and object detection.

CNNs can be defined as ANNs that can detect more complex patterns in the input data, make sense of them, and classify the data into specific categories. The use of convolution layers is what differentiates ANNs from CNNs. They receive images as input, and the model interprets them as an array of pixels in the format of $h \times w \times d$. Through the network, the convolutional layers are the ones responsible for extracting patterns from those images.

Overall, CNN uses three types of layers, Convolution Layers, Pooling Layers, and Fully-connected Layers. There are always two types of operation, convolution and pooling. The convolution operation, using multiple filters, can extract relevant patterns from the input data. The pooling operation is used to reduce the dimensionality from the extracted patterns.

CNNs can automatically extract features from the input data without human supervision and learn key features associated with each category. These advantages make CNN a great model in the field of computer vision. In Figure 2.6 [48] it is presented a representation of a CNN architecture.

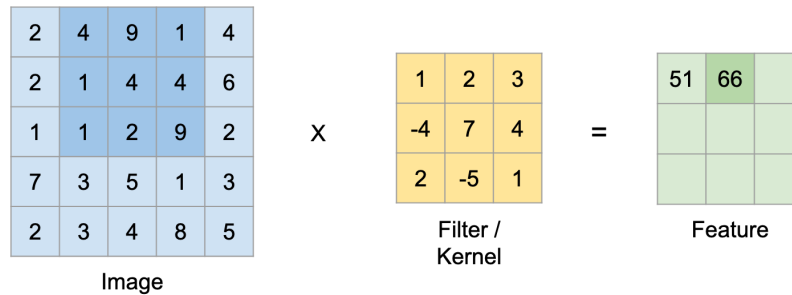


Figure 2.7: Convolution operational with a kernel size of 3x3. Figure extracted from [47]

2.5.1 Convolution Layer

Convolution layers are the most distinct element of a CNN. A convolution layer is made by a group of filters, known as kernels. Those kernels run over the input data, typically a matrix of images, and compute a dot product. The result of this multiplication usually is called Feature Map. By doing this convolution, the layer is able to extract features from small squares of input images, maintaining the relationship between the original pixels. Figure 2.7 presents a convolution operation with a 3x3 filter being applied to a 5x5 input image resulting in a 3x3 feature map.

At the beginning of the network, simple filters that can perform operations as edge detection, corner detection, and blur are expected. The deeper the network goes, the more sophisticated those filters become. In the end, they should be able to perform more complex operations as object detection.

2.5.2 Pooling Layer

The Pooling Layer is responsible for applying a downsampling operation along the spatial dimensions. The dimensionality of each map is gradually reduced while retaining the essential features of the data. When using these layers over the network, there is a reduction in the number of parameters and computation throughout the network.

2.5.3 Fully Connected Layer

In a fully connected layer, every neuron in one layer is connected to every neuron of another layer. In the later steps of a CNN, the feature map is fed into a fully connected network. To pass the matrix to this network, first, it is flattened into a vector. Then, it is applied an activation function such as softmax to classify the outputs.

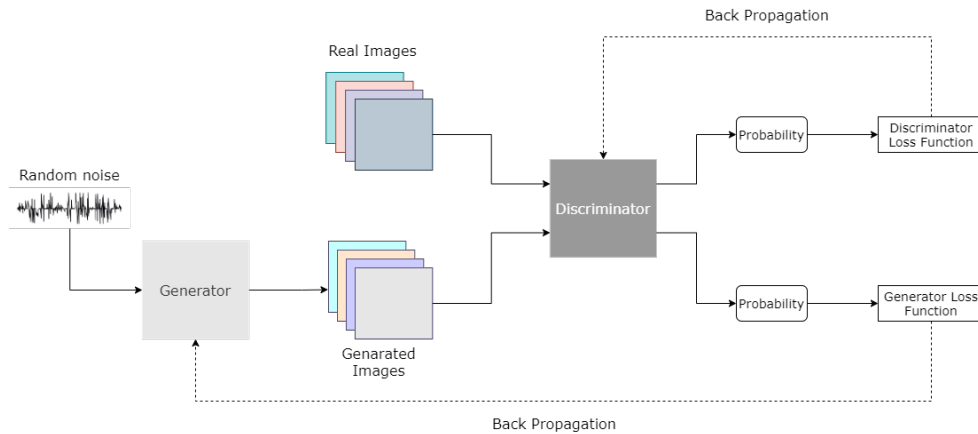


Figure 2.8: GAN Architecture

2.6 Generative Adversarial Networks

Generative Adversarial Networks [18] (GANs) is a DL generative model that introduced an innovative way of generating images via adversarial training. GANs were proposed by Ian J. Goodfellow and his colleagues in 2014 [18]. The ability of GANs to generate realistic examples in several domains and excellent results in image synthesis [27], image-to-image translation [23] [75] and other fields opened a big door in computer vision. Although GANs are not the only model capable of synthesizing images, their remarkable results have turned GANs into one of the most promising models for some computer vision tasks. Since 2014 a vast number of GAN-based models have been developed, achieving impressive results such as in generating photorealistic human faces [27] or coloring black and white images [23].

GANs introduced a new paradigm consisting of two adversarial neural networks, a discriminative and a generative network.

The Generator goal is to capture the data distribution of the training dataset and generate new images according to it, while the Discriminator network goal is to distinguish generated images from real images by learning how the real images should look. The Generator should be capable of fooling the Discriminator, making it incapable of distinguishing generated images from real images.

During training, the generative network learns how to create fake images identical to the real ones through the feedback that is provided by the Discriminator classifications, whether the images looked fake or appear to be real. Both Networks will improve at their task throughout the training. At the end of the training, it is expected for the Generator to produce new and unique images with features and characteristics of the real images from the training dataset.

2.6.1 GANs Training

In Figure 2.8 it is presented the GAN Architecture proposed by Ian J. Goodfellow and colleagues in 2014. The Generator network is fed with a random noise vector z and generates a fake image $G(z)$. The Discriminator receives the generated images $G(z)$ and outputs the probability of them being real $D(G(z))$. The Generator generates those images hoping for them to be considered authentic by the Discriminator network. The Generator's goal is to fool the Discriminator by producing fake images that are indistinguishable from the real images of the training data. In contrast, the Discriminator's objective is to be the more accurate as possible in identifying fake images produced by the Generator and recognizing the real ones [31].

The Networks are trained simultaneously using backpropagation. Each network has an individual cost function based on the Discriminator's loss. The Generator tries to maximize the Discriminator's loss while the Discriminator tries to minimize it. The worse the Discriminator is at distinguishing between real and false images, the higher the Discriminator loss will be. For the Generator, the Discriminator loss being higher indicates that the Generator is going in the right direction and that more fake images were able to fool the Discriminator. The backpropagation works separately for both networks, each network can only tune its parameters, but their cost functions are dependent on both the network's parameters. Hence, the two networks can be seen as two players trying to perform moves to improve their performances, not knowing the opponent's move. The training should end when the Generator and the Discriminator Networks reach *Nash equilibrium* [31]. *Nash equilibrium* is reached when none of the networks can improve their performance no matter what parameters they try to update. In other words, Nash equilibrium is reached when:

- The Generator produces images indistinguishable from the real images.
- The Discriminator can not distinguish real images from fake images.

The GANs training converges when *Nash equilibrium* is reached. Taking into consideration the complexity involved, is very difficult for GANs to reach *Nash equilibrium* and, therefore, to converge.

GANs loss function is the same for both networks. The Generator tries to minimize its value while the Discriminator tries to maximize it. The loss function can be described as presented in Equation 2.9.

$$\min_G \max_D L(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log (1 - D(G(z)))]. \quad (2.9)$$

Knowing that x is an instance of the real data from the training dataset and z an instance of random noise, in Equation 2.9:

- $D(x)$ is the probability estimated by the Discriminator of x being real.
- $\mathbb{E}_x$ is the expected value over all real data instances.

- $G(z)$ is the Generator's output given noise z .
- $D(G(z))$ is the probability estimated by the Discriminator of a generated instance being real.
- $\mathbb{E}_z$ is the expected value over all generated instances.

To further explain how the training of GANs is executed, the GANs training Algorithm is presented in Algorithm 1.

Algorithm 1 GANs Training Algorithm

for each training iteration **do**

1. **Train the Discriminator:**

- Take n samples from real examples x
- Take n samples of random noise vectors z and generate n fake examples $G(z)$
- Compute the Discriminator's predictions for $D(x)$ and $D(G(z))$, and backpropagate the total error.

2. **Train the Generator:**

- Take n samples of random noise vectors z and generate n fake examples $G(z)$
- Compute the Discriminator's prediction $D(G(z))$ and backpropagate the loss.

end for

2.6.2 GANs challenges

Despite achieving good results, researchers considered training GANs a very challenging task and have found some problems that can happen during the training that compromise the model performance.

Generating high-resolution images is one of the most challenging problems. When dealing with high-resolution images is easier for the Discriminator to tell apart the generated images from the training images, which makes the Discriminator much better than the Generator. This disparity can provoke a failure in the Generator training due to **vanishing gradients**. It prevents the Generator from making progress; hence, the Generator cannot learn the data distribution.

Mode collapse is also a very recurrent problem. It happens when the Generator network generates the same output image multiple times and, therefore, fails to produce a wide variety of output images.

2.6.3 GAN-based models

GANs can perform well in many computer vision tasks as image retrieval, image generation, image-to-image translation, and text-to-image translation. Every year new GAN-based models arrive and get better and better results in performing those tasks. As mentioned above, GANs have some problems and limitations. Since they were introduced in 2014, new models have been proposed to surpass those problems and limitations. All models presented below are just a small

sample of all the existing methods and only demonstrate that GAN's have a huge potential in computer vision.

- **Deep Convolutional GAN (DCGAN)** [49] attempts to use CNNs instead of fully connected networks on both the Generator and Discriminator. DCGAN is more suitable for receiving input images or videos, and it presents better results, while GAN is more suitable for wider domains and ranges.
- **Conditional GAN (cGAN)** [40] have more control over the data being generated by conditioning the model with class labels. cGAN allows the user to choose the category of the images that will be generated. This way, cGAN achieves more control over the generated images than GANs.
- **WGAN** [5] and **WGAN-GP** [45] proposed a change in the objective function using Wasserstein distance and Wasserstein distance with gradient penalty in order to improve training stability and robustness of GAN models.
- **CycleGAN** [75] is a GAN model which is not applied to image generation. Instead, it is a model applied in the field of image-to-image translation. CycleGAN proposes a method to transform unpaired images from one domain into another, having achieved good results.
- **StarGAN** [10] uses a similar approach as CycleGAN but can transform images from one domain to multiple domains. StarGAN demonstrates, through facial attribute transfer and facial expression transfer, that it is able to successfully learn multi-domain image translation and generate realistic images.
- **ProgressiveGAN** [26] tackles one of the biggest limitations of GANs, which is the low resolution of the generated images. ProgressiveGAN proposes a solution for this problem and can generate high-resolution images. The solution is based on the idea of growing both Generator and Discriminator progressively throughout the training.
- **StyleGAN** [28] is currently the state of the art of image generation GANs. StyleGAN has so far undergone some refinements [29] [44] and is the best GAN model when it comes to the quality and diversity of images generated at high resolution.

2.6.4 Evaluation

Evaluating a GAN can be challenging. Since 2014, the number of available measures has been growing, and some empirical studies [4] [7] have evaluated their performance. The best measures that capture the strengths and limitations of the models remain uncertain. Therefore the models should be evaluated based on the quality of the generated images. The evaluation measures can be quantitative or qualitative. Quantitative measures are numeric scores that evaluate the quality of the generated images.

2.6.4.1 Qualitative measures

Qualitative measures refer to non-numeric measures that require human subjective evaluation. Due to their subjective nature, whenever possible, they should be accompanied by quantitative measures. Some of the most commonly used qualitative metrics are presented below.

- **Preference Judgment** - Rating and preference judgment is a subjective measure that requires human analysis. The human judges have to compare and rank examples from authentic and generated images. This measure should be complemented with quantitative measures.
- **Rapid Scene Categorization** - Rapid Scene Categorization requires the collaboration of human judgment. It consists of presenting images for a limited time and asking judges to classify them as real or fake. Images are presented in pairs, and the judges should select the image they consider as the more realistic. One downside of using this measure is that distinguishing fake images from real ones can improve over time.

2.6.4.2 Inception Score

Inception Score [55] (IS) is one of the most used measures to assess GANs image quality. It uses both the quality and diversity of the generated images to measure its performance. It uses the Inception-v3 pre-trained Network trained on the ImageNet data to classify the generated images. The model receives as input a large number of generated images and measures the average KL divergence between the conditional label distribution $p(y|x)$ and the marginal distribution $p(y)$ from all the samples. Kullback-leibler (KL) divergence or relative entropy measure the differences between two different probability distributions. The Inception Score function is described in Equation 2.10.

$$IS = \exp(E_x[D_{KL}(p(y|x) || p(y))]). \quad (2.10)$$

- $p(y|x)$ is the conditional label distribution for an image x .
- $p(y)$ is the marginal distribution obtained from all samples.
- D_{KL} is the KL divergence between the probability distributions of $p(y|x)$ and $p(y)$.

Higher inception scores correspond to models with larger KL-divergence values and better quality in the generated images. It has limitations such as favor models that generate good objects instead of realistic images.

2.6.4.3 Frechet Inception Distance

Frechet Inception Distance (FID) [4] compares the statistics of the generated images to the real images, instead of classifying the generated images. So, it can compare the similarity between

them. The metric FID uses the Inception-v3 network trained on the ImageNet Dataset. The FID extracts the activations of the last pooling layer of the Inception-v3 network. The FID measures the distance between the activations of the real and generated images.

The Frechet Inception Distance between the real images and generated images can be described as:

$$FID(r, g) = \|\mu_r - \mu_g\|^2 + Tr \left(\sum_r + \sum_g - 2 \left(\sum_r \sum_g \right)^{\frac{1}{2}} \right) \quad (2.11)$$

Knowing that r is the activation of real images and g is the activation of the generated images:

- μ_r is the mean of activation r .
- μ_g is the mean of activation g .
- $\sum_r$ is the covariance of activation r .
- $\sum_g$ is the covariance of activation g .

A lower FID indicates smaller differences between the data distribution of real and generated data. In other words, it points to a higher similarity between them.

Chapter 3

Literature review

In this chapter, several GAN-based models for image generation and image-to-image translation are presented. Furthermore, a global perspective of different GAN models applied in distinct tasks within the fashion field will also be provided.

3.1 Image Generation

As mentioned above, the application of GANs in the fashion context is vast. In this section, it will be referred a few image generation GANs. Most image generation models are applied to broader domains other than fashion.

3.1.1 Pixel-wise

Ruffino and his colleagues [53] presented a gan-based structure that applies pixel constraints in the generated images though enforcing pixel-wise conditioning. It aims to match the distribution of the real images conditioned on a highly scarce constraint map. It uses PacGAN [34] technique to ensure diversity in the generated images. The PacGAN technique modifies the Discriminator network to mitigate mode collapse. Instead of mapping a single image real or generated to a specific label, the Discriminator will map n samples from the real or generated data to a specific label.

3.1.2 ProgressiveGAN

One of the biggest challenges for GAN models is generating high-resolution images. ProgressiveGan [26] or progressive growing generative adversarial network can improve image quality and resolution by growing both Generator and Discriminator progressively. It presents a different training methodology that starts with low-resolution images and gradually increases the resolution by adding new layers progressively to the networks. Instead of finding all kinds of structures

and details simultaneously, the network first discovers large-scale structures and then focuses on small-scale details. By incrementally adding layers to both Generator and Discriminator, the generated images' spatial resolution is increased. ProgressiveGAN makes use of WGAN-GP [45] loss function. Compared to other models, training is faster and more stabilized.

3.1.3 StyleGAN

While in progressiveGAN there is almost no control in the generated images, StyleGAN [28] presents a new approach with higher control over the generated images. With StyleGAN, it is possible to control small characteristics of the generated data. The presence of some hyperparameters that can be tuned by the user allows this control. It permits a small factor of variability in the generated images due to the addition of an adaptive instance normalization, AdaIN [22] in the Generator at each convolution layer. StyleGAN uses the same training principle used by progressiveGAN. The resolution of the generated images is gradually increased throughout the training. The major modification is in the Generator architecture that greatly diverges from the typical GAN Generator architecture.

Even though it achieved unprecedented results in high-resolution image generation, StyleGAN still generates some images with less quality and the presence of unwanted artifacts. An improved version, called StyleGAN2 [29] revised StyleGAN architecture, could achieve better quality, more realistic, and defined results than the initial version. StyleGAN2 pinpointed the unwanted artifacts to the Generator architecture and eliminated the problem by redesigning the normalization applied in the Generator. StyleGAN2 also altered the progressive growing strategy allowing even better results. Figures 3.1 and 3.2 show images generated respectively by StyleGAN and StyleGAN2. Training StyleGAN2 with smaller datasets typically causes the Discriminator to overfit, and as a consequence, the training diverges. In 2020 StyleGAN2-ADA [27] has proposed as an improvement over StyleGAN2. StyleGAN2-ADA uses several data augmentation techniques and achieved impressive results with much smaller datasets than StyleGAN and StyleGAN2.



Figure 3.1: Images generated by StyleGAN [64]



Figure 3.2: Images generated by StyleGAN2 [44]

In a paper presented by Kathleen M. Lewis and colleagues, StyleGAN2 is used as the base of a new model VOGUE [32]. VOGUE is a controllable image generation model that takes two



Figure 3.3: Images generated by VOGUE given a target person and a specific top garment [32]

images as input, the first one with a target person and the second one with a person wearing a target garment. The algorithm generates the target person dressed in the target garment, top or bottom. In Figure 3.3 are noticeable the excellent results both in the quality and realism of the generated images. VOGUE surpasses significantly in both quality and realism the existing try-on methods.

3.1.4 SpatialGAN

SpatialGAN [74] proposes a progressive spatial recursive generation model capable of producing high-quality images. It generates the whole image gradually from the center, which is considered the attention part of the image or the most important part of the image, to the image space. This way, it transforms high-dimensional problems into several low-dimensional problems.

SpatialGAN consists of basic attention adversarial networks as DCGAN [49], BEGAN [6] or WGAN [5] and several spatial adversarial Expansion Networks. It divides each image from the center into several image blocks, called patches. It starts by generating the center and then recursively expands the surrounding image space, using the previously generated content as a constraint for the image that is being generated.

In the paper [74], SpatialGAN results are compared to the progressiveGAN results. Both models achieved very similar inception scores, SpatialGAN presented an FID (Fréchet Inception

distance) slightly lower than progressiveGAN. Analyzing the results, SpatialGAN points to similar or slightly better results than progressiveGAN.

3.2 Image-to-Image Translation

Image-to-image translation can be defined as learning how to map an image from one domain to another domain. The main goal is to transform images from one domain into images with a similar style or characteristics into images from the other domain.

3.2.1 Pix2pix

Pix2pix [23] is a paired image-to-image translation model. Pix2pix uses a paired dataset, each pair of images is made up of an image from one domain and the corresponding image in another domain. It is a conditional GAN, every output of the model is conditioned on a target input. It can be used for problems such as coloring greyscale images or turning sketches into real images. Pix2pix learns how to map an image from one domain to another.

Pix2pix Generator uses U-Net based architecture and the PatchGAN Discriminator. While typical Discriminators will output only one classification per image, the PatchGAN Discriminator outputs a matrix of classifications. Instead of judging the authenticity of an image as a whole, it will judge the authenticity of all the patches in the image. Pix2pix Generator follows an encoder-decoder structure that takes in an input image, encodes the image, and then decodes it. The goal is for the decoded image to have a different style conditioned by the input image. This Generator uses skip connections which helps the decoder to learn more details directly from the encoder. Skip connections mean skipping some layers in the network, passing the outputs of a layer as input to other layers instead of only to the following layer.

3.2.2 CycleGAN

CycleGAN [75] is an unpaired image-to-image translation GAN. Finding pairs of images to create paired datasets is very complicated, expensive, and time-consuming. Unpaired image-to-image translation lies in identifying the differences and similarities between two separate sets of images without the need for paired data. The goal of these models is to identify each set's style and global characteristics to convert images from one set to images with the style of another set, maintaining the content of the images.

CycleGAN proposed a method to transform unpaired images from one domain to another using two GANs. Two U-net based Generators and two patchGAN Discriminators are used. The network $N_{x \rightarrow y}$ is responsible for receiving images from domain x and transform them into images with the same style and characteristics as the images from domain y . Similarly, the network $n_{y \rightarrow x}$ receives as input images from domain y and has to transform them into images that can fit in the domain x . CycleGAN loss function is comprised of multiple losses. Each network loss function has a Discriminator loss, a cycle consistency loss, and an identity loss.



Figure 3.4: Image generated by DISCO-GAN[60]

3.3 GANs for Fashion

3.3.1 Garment Matching Design

Garment Matching can be defined as the generation of a clothing piece that pairs nicely with another clothing piece so that the overall outfit can be considered a fashionable outfit. Being well dressed is not an easy task for many people, and so, some generative models as Attribute-GAN[36] and MGCM [35] were developed to learn how to generate a bottom garment that matches with a given top garment. Both models make use of textual descriptions to characterize the images of the used datasets.

3.3.2 Garment Translation

Another interesting topic is how to generate clothes gathering characteristics from two different domains. The paper [60] by Singh, and his colleagues prove that by using DISCO-GAN [30] it is possible to generate clothes with characteristics from a trendy clothes domain and a vintage clothes domain. DISCO-GAN discovers cross-domain relations given unpaired datasets. It can transfer style from one domain to another while preserving some features and characteristics. In Figure 3.4 a trendy generated garment can be observed, the model preserved the garment tones while changing its shape.

In the design process finding the right colors and textures for garments is a very time-consuming process. GD-StarGAN [57] propose a multi-domain image-to-image translation in garment design



Figure 3.5: Image generated by TextureGAN [68]

capable of applying a given texture to a garment of choice, a dress, a t-shirt, or any other. To generate a new image is necessary to give as input the texture image and a label defining the type of garment to be generated. Despite the model managing to preserve the texture quite well, there is no control over the garment's shape that is being generated.

A different approach to help designers during the creative process is to turn fashion sketches into realistic clothing items. TextureGAN [68] and FashionGAN [70] are two different models that address this problem.

TextureGAN [68] is a conditional generative network that turns fashion sketches into realistic images. The TextureGAN model takes a sketch with an overlaid texture patch placed in a position chosen by the user and learns how to synthesize realistic images consistent with the sketch shape and texture provided. The network generates images in the Lab color space and enforces a separate color loss on the ab channels. The training is divided into two steps, pre-training and fine-tuning. In the pre-training step, the model is trained to produce ground-truth images. The loss function comprises several loss functions: feature loss, adversarial loss, pixel loss, color loss, and style loss. In the fine-tuning step, an additional texture dataset is used to help TextureGAN propagate better in unseen textures, and a local texture loss is proposed. The local texture loss is formed by a local adversarial loss, a local style loss and a local pixel loss. In total, this network has a Generator, a global Discriminator, and a local Discriminator. Figure 3.5 shows some results obtained through the TextureGAN model.

FashionGAN [70] proposes a virtual garment display based on conditional generative adversarial networks. The model takes two inputs, the fashion sketch chosen and a specified fabric image, and it is able to generate images with the texture of the selected fabric and the shape of the fashion sketch. The FashionGAN model can perform three different image-to-image translation tasks, including fashion sketch to garment image, one garment image into another garment image, and contour image to garment image. The model is inspired by BicycleGAN [76] which combines

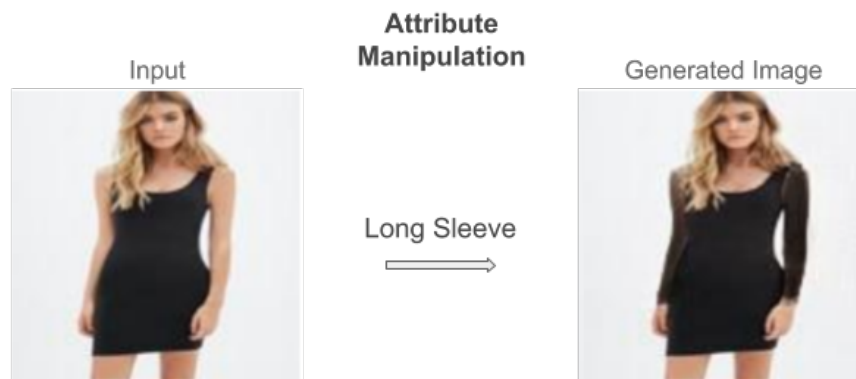


Figure 3.6: Image generated by AMGAN [1] when modifying the sleeve length attribute

two different types of networks, cVAE-GAN, and cLR-GAN.

3.3.3 Garment Attribute Manipulation

Garment Attribute manipulation consists of changing a specific attribute or characteristic of a chosen garment. The attributes to be modified can vary from the type of neckline to the garment's color and pattern. During the designer's creative process, it is frequently necessary to make minor adjustments to the designs after they are finished.

AMGAN [1] enables computers to perform these adjustments while preserving the remaining characteristics of the garment. This way, designers can define the attributes they want to change and automatically perform this change faster and more cleverly. Figure 3.6 presents an image generated by the AMGAN model with the attribute sleeve length being manipulated.

3.3.4 Garment Extraction

Garment extraction is the task of extracting a piece of clothing from a human model. Zhang *et al.* [72] proposed a new framework named CatGAN able to generate tiled clothing images automatically. The framework utilizes GANs to learn transformation rules between clothes on wearers and clothes that are tiled. CascadeGAN [73] proposes a two-stage cascade GAN capable of extracting a given garment from a human model. CascadeGAN showed better results and performance than the CatGAN model. In Figure 3.7 a comparison between CascadeGAN and CatGAN extracted garments is made.

3.3.5 Garment Transfer

Another aspect that has also been widely studied is garment transfer. Garment transfer consists in placing a particular piece of clothing on a human model. The task is not trivial as it depends on the model's pose.

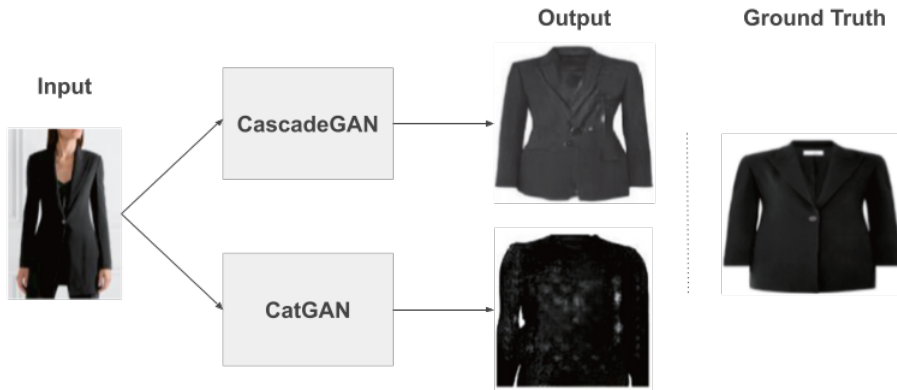


Figure 3.7: Image generated by CascadeGAN [73] and CatGAN [72] when extracting a blazer from a model

Poly-GAN [46] is a conditional GAN capable of placing a piece of clothing on a model having into account the RGB skeleton of the model at an arbitrary pose. CP-VTON [65] and CP-VTON+ [39] are also two models proposed in the field of garment transfer. Both models can place a target garment on a human model. GarmentGAN [50] can transfer a target clothing item onto a reference body. During training, GarmentGAN uses additional information comprising segmentation maps and body key-points information. During the try-on task, the model is able to preserve the reference person and the clothing items that are not being changed. Figure 3.8 shows a comparison between the generated images by GarmentGAN and CP-VTON models.

3.3.6 Garment Generation

Building a system capable of designing new pieces of clothing can be very useful for the fashion industry because it can assist designers in the creative process of clothing collections and accelerate the creation process. Listed below are three different ways to generate clothing items automatically.

- Unconditional Garment generation
- Conditional Garment generation based on labels
- Conditional Garment generation based on textual descriptions

3.3.6.1 Unconditional Garment generation

Unconditional Garment generation consists in generating images without any kind of conditions. The main goal is from a latent code belonging to a latent space to generate different, realistic, and diverse pieces of clothing.

In FashionGen [52], some experiments were carried out with a fashion dataset containing slightly less than 300 000 dressed models. The ProgressiveGAN model was used and showed quite

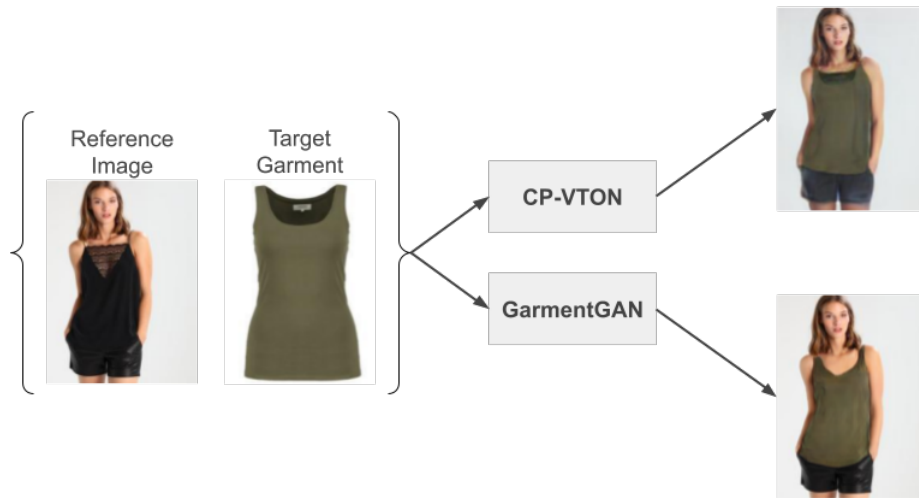


Figure 3.8: Images generated by GarmentGAN [50] and CP-VTON[65]

good results. To evaluate the image quality, the metric Inception Score was calculated for both generated and real images. The inception score of the images generated by the model was very close to the used dataset. Figure 3.9 shows the garment images generated using ProgressiveGAN.

3.3.6.2 Conditional Garment generation based on labels

Conditional Garment generation based on labels consists in generating images that are conditioned on a particular label. The main goal is to generate clothes from a random vector of the latent space and a specific label. The generated images should be realistic, diversified, and fit within the chosen label. Conditional GAN is the most common model used for this task. In figure 3.10 a couple of examples of cGAN generated images applied to a fashion dataset are presented.

3.3.6.3 Conditional garment generation based on textual descriptions

Conditional garment generation based on textual descriptions is included within the field of text-to-image synthesis. Text-to-image synthesis consists in generating realistic images semantically consistent with a detailed textual description. These models aim to generate an image that matches a textual description provided by users. In general, the datasets used to train such models require the images and their textual descriptions.

In this field and applied to garment generation, e-AttnGAN [2] proposed a GAN-based model specially designed for fashion image synthesis. Given a textual description, it can generate the corresponding garment. In Figure 3.11 a sample of the results obtained through this model is presented.



Figure 3.9: Images generated by ProgressiveGAN [52]

3.4 Fashion Datasets

The data used in the training of deep learning models is essential. Data makes the difference when it comes to the results obtained and the performance of the models. Some of the most relevant fashion datasets found are presented below.

- **Fashion MNIST** [69] is a dataset with approximately 70 000 images of fashion items. Each image is a 28x28 grayscale image and has an associated label. The dataset contains images from 10 different classes, including different types of clothing and different types of shoes.
- **Deep Fashion** [37] is a large-scale clothes database with approximately 800,000 fashion images. Each image contains information regarding the category the image belongs to and its descriptive attributes, bounding box, and clothing landmarks. The deep fashion dataset is made up of 50 different categories and 1,000 different descriptive attributes.
- **Deep Fashion 2** [15] is a clothing item dataset that contains around 491000 clothing images of 13 different clothing categories from both consumers and shopping stores. This dataset also contains 801k clothing items with detailed annotations. Each item contains information about its scale, occlusion, zoom-in, viewpoint, category, style, bounding box, dense landmarks, and per-pixel mask. DeepFashion 2 dataset is also made up of 873000 pairs of commercial-consumer clothes.
- **Fashion-Gen** [52] is a dataset with nearly 300000 sets of high definition fashion images. Each set is made up of an image photographed from different angles and a detailed textual description.



Figure 3.10: Garments generated by cGAN [40] on Fashion MNIST dataset. Images extracted from [8]

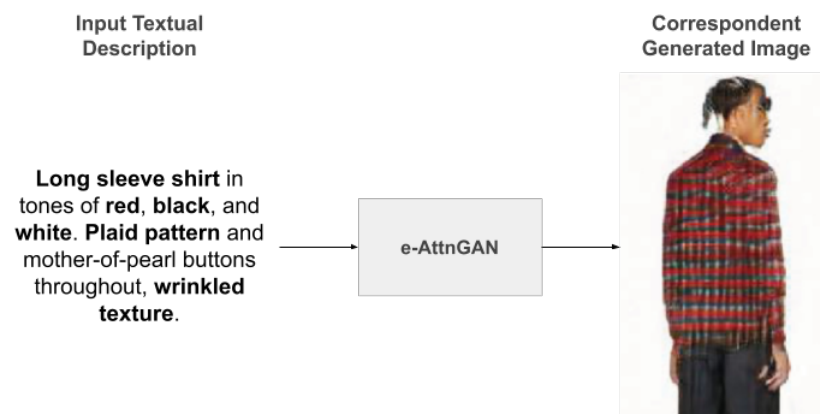


Figure 3.11: Garment generated by e-AttnGAN [2] on Fashion-Gen dataset

Chapter 4

Methodology

In the present chapter, the methodology employed to conduct this research is outlined. The research process had four different stages, all of which are described below. This chapter also details each component of the proposed solution.

4.1 Research Process

The first stage of the present research is the **literature review stage**. The main goal in this stage was to identify AI models applied to fashion in order to identify their limitations and contributions. Models to solve problems such as garment attribute manipulation, garment generation, garment translation, garment extraction, and garment transfer were discussed in detail. Furthermore, deep learning models for image to image translation and image generation tasks were also outlined. At last, well-known fashion datasets were studied and explained.

The second stage is the **problem analysis stage**. In this stage, the research problem and research gap were defined. No research work that aimed for the conditional generation of clothing garments was found. At this stage, the problem statement as well as the research questions and research objectives were formulated.

The third stage is called the **research approach stage**. This stage focused on developing the proposed models to solve the research problem and fulfill the proposed research objectives. The first proposed model addresses conditional garment generation, producing garments according to a criterion defined by the user. The second proposed model can be placed within the clothing translation problem. This model takes fashion sketches with small fabric patches as input and transforms them into real representations of garments.

The last stage is the **evaluation stage**. The evaluation stage was focused on the evaluation of the implemented models. The evaluation was conducted using performance metrics, namely FID and SSIM, calculated during the models' training. Additionally, a user survey was carried out combining generated images with real images to validate the generated images' credibility and marketability.

4.2 Proposed Models

This dissertation implements two separate models that can assist designers in the design creation phase. A brief description of the proposed models is given below:

- **Conditional Garment Generation:** The Conditional Garment Generation model consists in generating garments conditioned on class labels as illustrated in Figure 4.1.

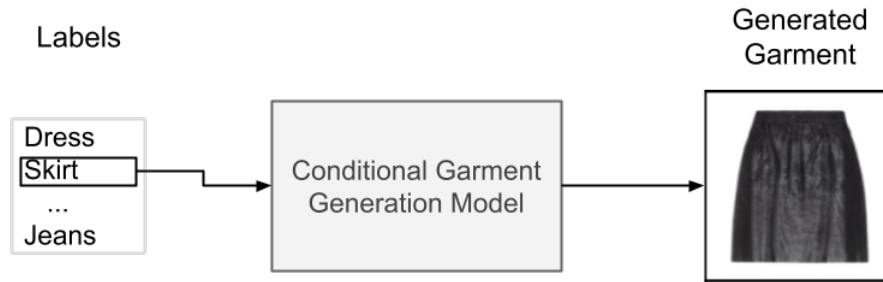


Figure 4.1: Condition Garment Generation model

- **Pattern Transfer:** The Pattern Transfer model consists in converting fashion sketches with overlaid texture patches into realistic garment images as illustrated in Figure 4.2.

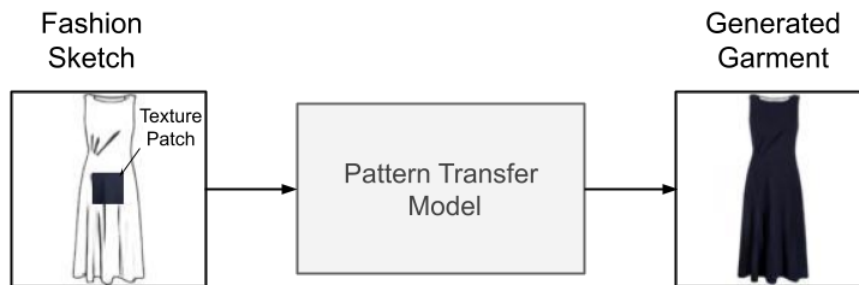


Figure 4.2: Pattern Transfer model

In the following sections the models implemented in this research work are described in full detail. In Section 4.3 the components of the Conditional Garment Generation model are presented. The Pattern Transfer model and its components are described in Section 4.4.

4.3 Conditional Garment Generation model

A component of the proposed solution is the implementation of a Conditional Garment Generation (CGG) model. This model is designed to generate garments that satisfy certain conditions. The conditions are defined by class labels which, in this model, correspond to different types of garments. Figure 4.3 illustrates all the components of the CGG model. The model is fed with a random noise vector known as latent vector z which is then concatenated with a specific label y (e.g., dress) and should generate a garment with features that fit within the label that was fed into the model. One major advantage is allowing users to select the type of garment they want to generate. Conditional generation gives more control over the generated images than unconditional generation. The proposed model is named Conditional Garment Generation model and was implemented in partnership with XNFY lab.

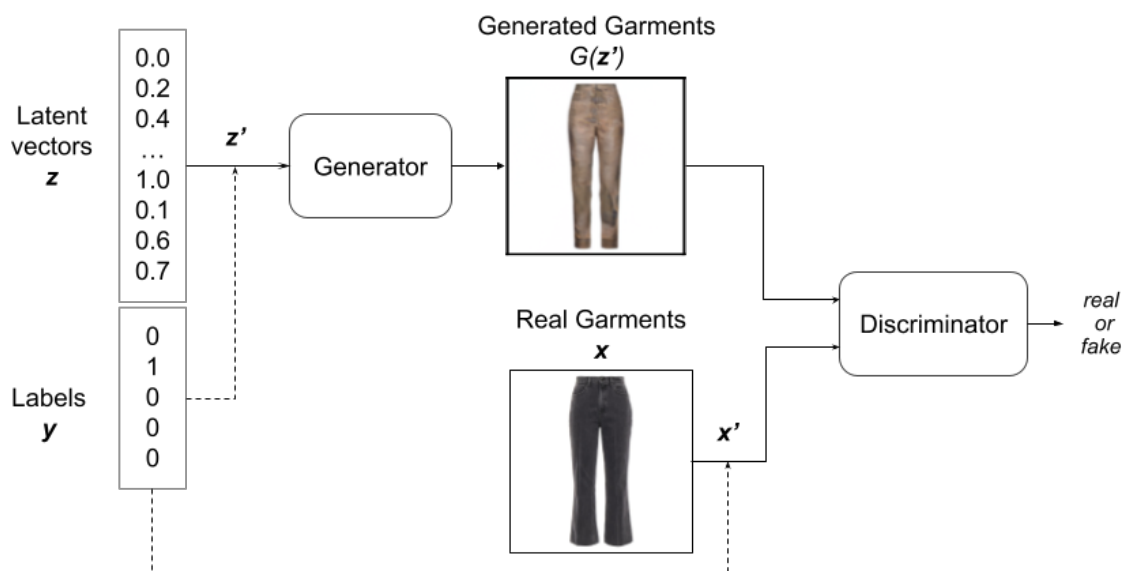


Figure 4.3: Overview of the Condition Garment Generation model components

The **Conditional Garment Generation** model is based on the architecture of StyleGAN2-ada [27], which is the state-of-the-art model for image generation. Data augmentation methods were also implemented to ensure good results using smaller datasets. Unlike in the StyleGAN2 model, the Generator and the Discriminator receive information regarding image labels as illustrated in Figure 4.3. This way, the model associates the images with their labels and is able to generate the images correctly according to the associated labels.

The entire architecture of this model will be explained in detail below. Subsection 4.3.1 and subsection 4.3.2 carefully details the *Generator* and *Discriminator* architecture, respectively. The Regularization methods used to ensure better training stability are described in subsection 4.3.3. The sampling techniques used to improve image quality during sampling are detailed in subsection 4.3.4. Finally, the loss functions used are explained in subsection 4.3.5.

4.3.1 Generator

The Generator is formed by two distinct networks, a mapping network, and a synthesis network as illustrated in Figure 4.4. Their architecture is described in detail below.

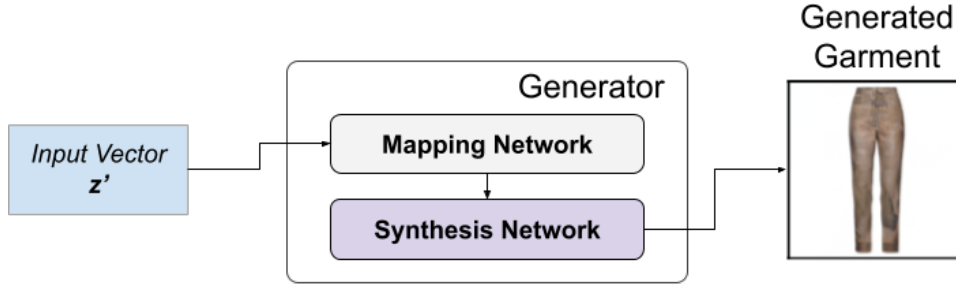


Figure 4.4: Generator Networks of the CGG model

4.3.1.1 Mapping Network

The Mapping Network is a Multilayer Perceptron Network (MLP) made up of 8 fully connected layers. Figure 4.5 illustrates the Mapping Network architecture. The Network takes as input a vector z' and maps it to a latent space W . The input vector z' is obtained from the concatenation of two vectors. The first vector is a random noise vector z also known as latent vector from latent space Z with dimension 512, while the second is a vector representing a particular garment label. Each label has a unique label vector that consists of 0's in all cells except in one. That cell is filled by the value one and is used to identify the specific label. The dimension of the label vector is equal to the number of different labels in the dataset used. The total dimension of the input vector z' is the latent space dimension plus the number of labels available. The input vector z' is normalized using pixel normalization, and then it is used as input to the Mapping Network. The Mapping Network is formed by 8 Fully Connected layers followed by Leaky ReLU activations. The Mapping Network outputs a Style vector w from latent space W which is used as input for the Synthesis Network.

The latent space W is less entangled than the latent space Z . The latent space being less entangled means that there are less chances of 2 different features of the generated images being modified (*e.g.*, sleeve length and color), when a single digit in a latent vector w is changed. In other words, images features are less entangled in latent space W than in latent space Z . The StyleGAN2 authors prove that applying a sequence of affine transformations to z' could result in less entangled features in w . Less entangled features lead to more control and smoother interpolation between generated images.

4.3.1.2 Synthesis Network

The Synthesis Network is formed by several Style Blocks. Each block is fed with a noise vector n , a style vector w , and the output of the previous block. The first one receives a learnable constant

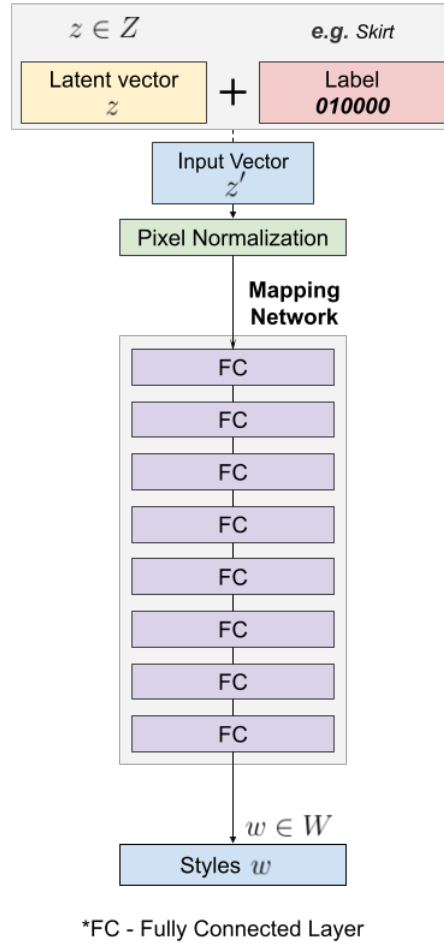


Figure 4.5: Mapping network architecture. Diagram partially adapted from [64]

vector x instead of the output of the previous Style Block. They all behave as a feature map upsampler, doubling the resolution of the input feature maps. Therefore, the image resolution will progressively increase until it reaches the desired resolution of 128×128 for the three RGB channels. A high-level overview of the Synthesis Network architecture for 128×128 output image resolution is presented in Figure 4.6.

Each style block produces two outputs rgb and x . Those outputs are three-dimensional vectors modified in each style block and then fed to the following block. x contains information about each image, such as its structure (e.g., shape) and certain details (e.g., color) that should be provided to the following blocks to guide the entire image generation process. rgb contains a representation of the generated image in lower resolution. Thus, it directly controls the final image generated by the Synthesis network. The resolution will increase along with the style blocks, starting at 4×4 in the first block and ending at 128×128 in the last block. Each style block's input x is modified by styled convolutions, affine transformations, and Leaky ReLU activation functions. It is then summed to the input rgb in the ToRGB Block. In the last Style Block, x is discarded and the final generated

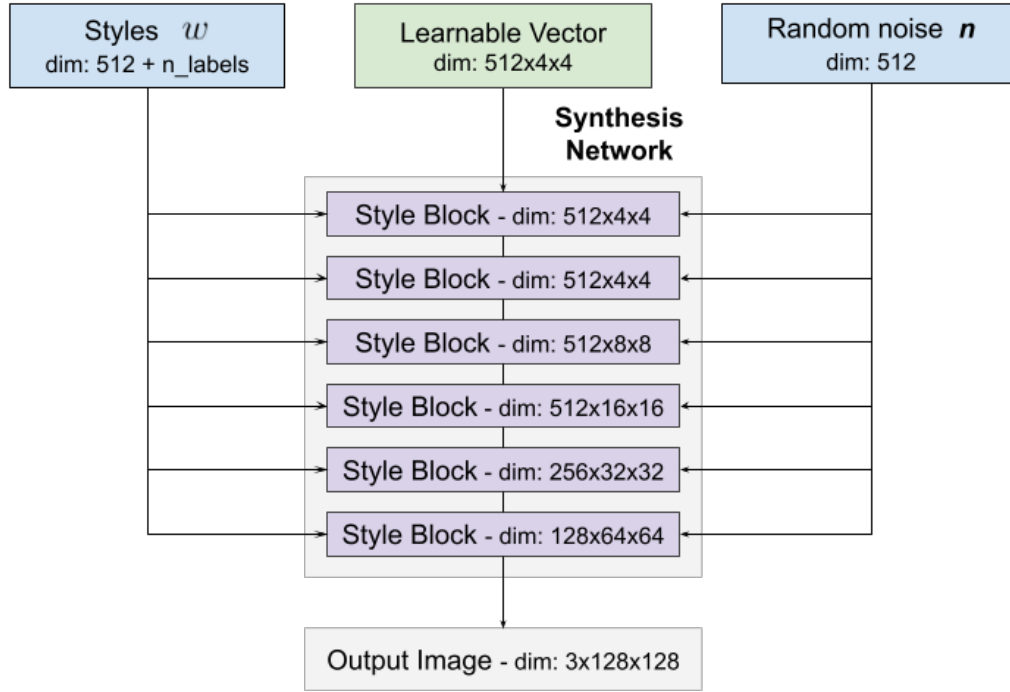


Figure 4.6: Synthesis network architecture for 128x128 of resolution

images are provided by rgb . Figure 4.7 presents an overview of the style block architecture and its components.

The style blocks are formed by several Styled Convolutions. Style Convolutions were introduced by [44] in order to eliminate unwanted artifacts produced by the StyleGAN Generator. The use of Style Convolutions enables more control over the Generator synthesized images. Each Style convolution performs 2D convolutions based on learnable parameters and weights calculated using style vector w . The 2D convolution is then summed with an affine transformation of a random noise n followed by a Leaky ReLU activation. Each style block has two sets of styled convolutions, affine transformations, and Leaky ReLU activations. The output x of each style block is given as input to the ToRGB Block. This block takes two inputs, rgb and x and outputs a modified version of rgb . Each ToRGB Block applies a styled convolution to x sums it to the input rgb and then applies a Bilinear Upsampling to the result. The output of the block has twice the resolution of the input rgb . Figure 4.8 illustrates the ToRGB block architecture.

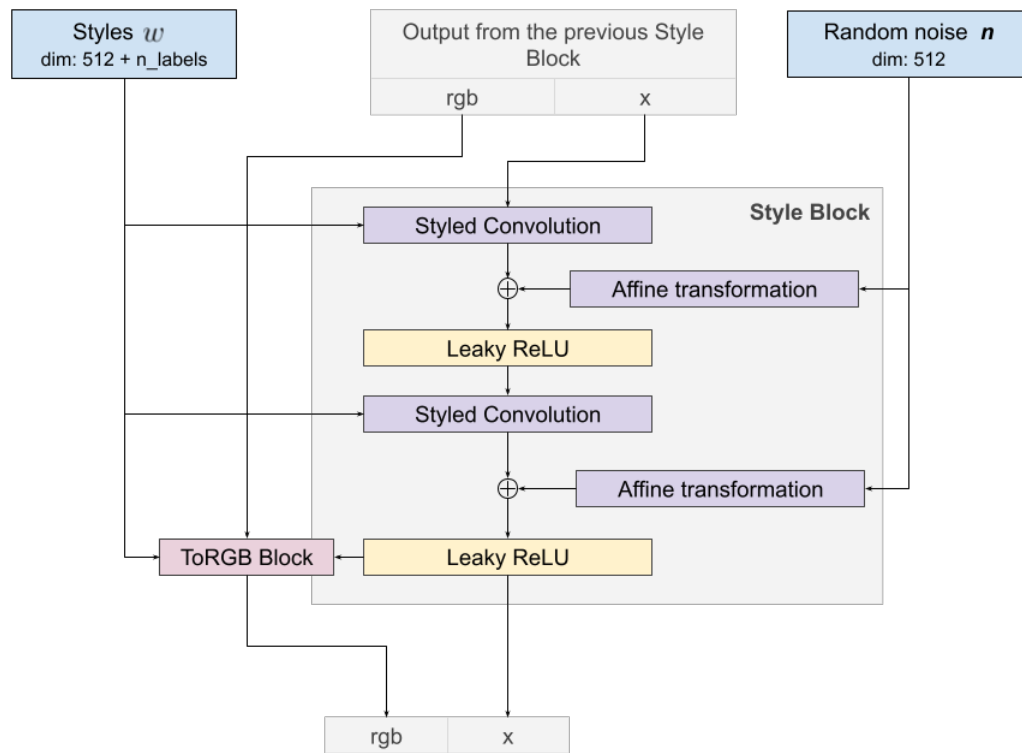


Figure 4.7: Architecture of the Style Block from the Synthesis Network

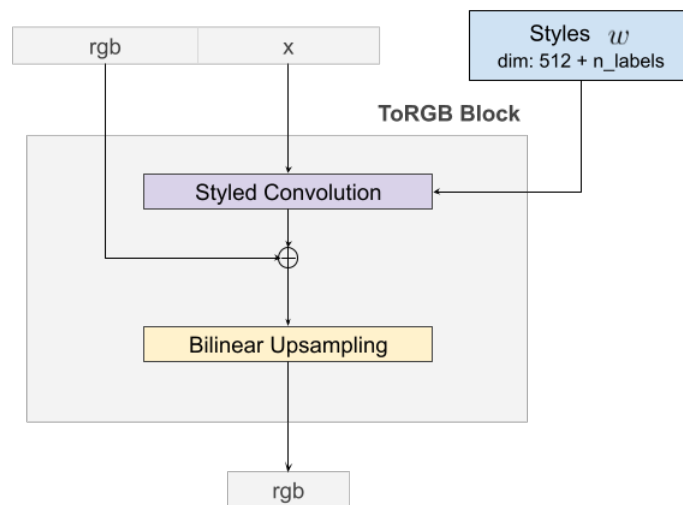


Figure 4.8: Architecture of the ToRGB Block from the Synthesis Network

4.3.2 Discriminator

The Discriminator network is constituted by several blocks, named Discriminator blocks. Those blocks progressively reduce the resolution of the input feature maps. The last block is entitled Discriminator Epilogue and outputs a scalar value with the probability of the input image being authentic or not. Figure 4.9 presents an overview of the CGG model Discriminator architecture for 128x128 image resolution.

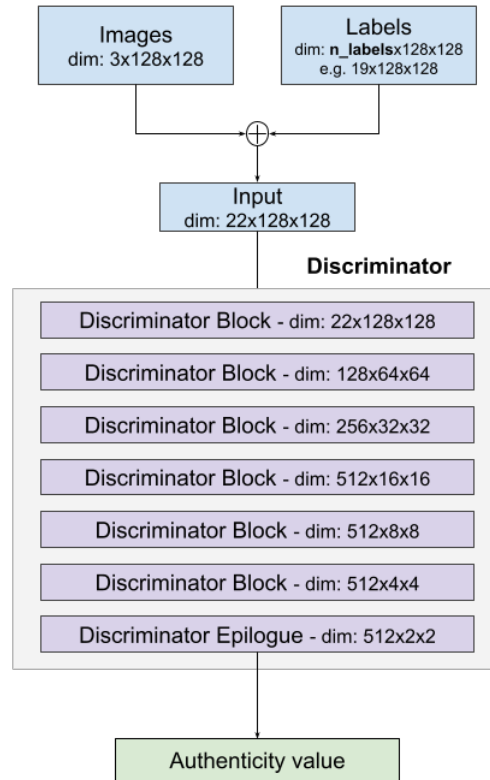


Figure 4.9: Discriminator network for 128x128 image resolution

The first Discriminator block is fed with a batch of labeled images and outputs a 3-dimensional vector x . The other blocks receive x from the previous Discriminator block and return a modified version of x . Each Discriminator Block consists of several 2D Convolutions followed by Leaky ReLU activation functions. In each Discriminator block except the last one, a downsample operation is applied as well. The last Discriminator block outputs a variable x , given as input to the Discriminator Epilogue Block. Figure 4.10 presents the architecture and operations that make part of each Discriminator Block.

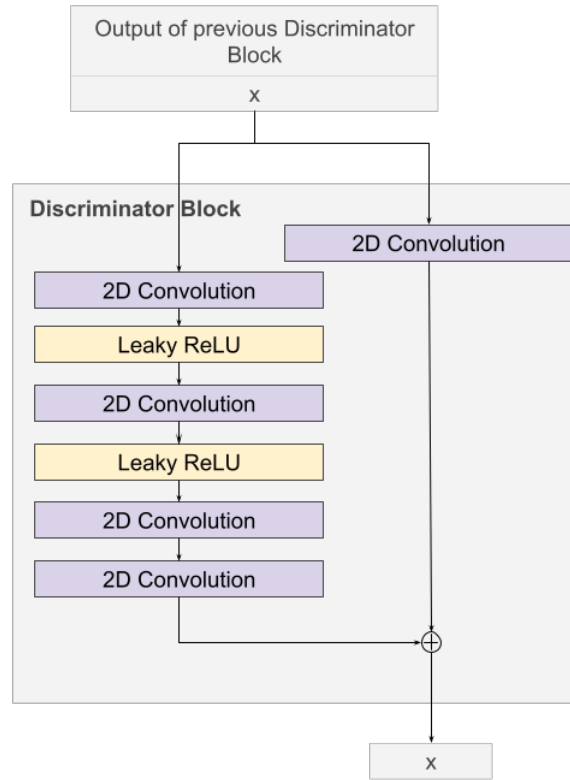


Figure 4.10: Architecture of the Discriminator block from the Discriminator of the CGG model

The Discriminator Epilogue block outputs the Authenticity value. This value is the Discriminator's prediction over whether the images it received as input are authentic or not. This block is formed by three series of 2D Convolutions followed by a Leaky ReLU activation function, a flattening of a 3D vector, and finally, a Fully Connected Layer. Figure 4.11 presents the architecture of the Discriminator Epilogue Block.

4.3.3 Regularization Methods

Regularization methods are used to ensure stability during training. Different methods are applied in the Generator and Discriminator networks. Path Length Regularization and R1 Regularization are applied in the Generator and the Discriminator networks, respectively. Both methods were also applied in the StyleGAN2 model. The Data augmentation techniques used in StyleGAN2-ADA are also implemented. The regularization methods are explained in detail below.

4.3.3.1 Path Length Regularization

Path length Regularization is used to encourage the Generator to make fixed-magnitude alterations in the generated images. The Generator for each fixed-size step taken in the styles vector w should produce fixed-magnitude changes in the generated images. Path length Regularization ensures

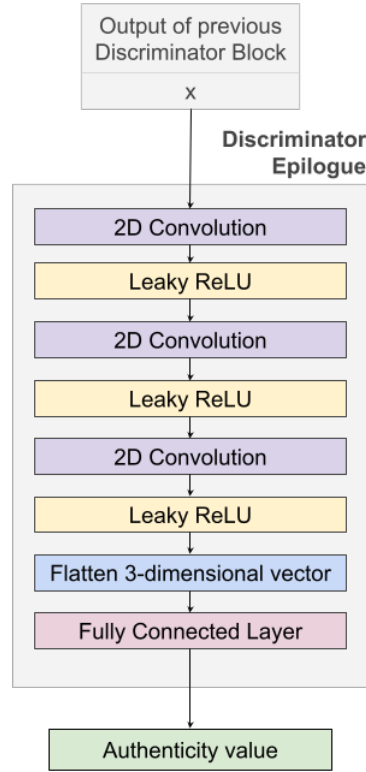


Figure 4.11: Discriminator Epilogue architecture

that small changes in the style vector w produce small changes in the generated images, and larger changes in w produce larger changes in the generated images.

The Path Length Regularization loss term is estimated by keeping an exponential running average of the Average Path Length and subtracting the mean Path Length of the generated samples from the running average. The Path Length of every mini-batch is achieved by combining noise extracted from a normal distribution with the generated images and calculating the norm of the gradients of the resultant images regarding the style vector w . The loss term is then multiplied by a hyperparameter w_{PLR} that controls the weight of the path length regularization throughout the model's training.

4.3.3.2 R1 Regularization

R1 regularization is implemented to penalize the *Discriminator* for deviating from the GAN game nash-equilibrium. The gradients of the Discriminator output with relation to the samples of every mini-batch of real images $\nabla D_{(x)}$ is used to calculate the R1 regularization loss term. The loss term can be defined as the squared value of the norm of the gradient $\nabla D_{(x)}$ minus 1, as presented in Equation 4.1:

$$R1 = (\|\nabla D_{(x)}\| - 1)^2 \quad (4.1)$$

The loss term is then multiplied by a hyperparameter w_{R1} that controls the weight of R1 regularization throughout the model's training.

4.3.3.3 Data augmentation techniques

The need for a large quantity of training data is one of the main requirements to achieve good results. Collecting enough data is not always easy and sometimes even impossible. When training GANs with low amounts of data, the Discriminator tends to memorize irrelevant details of the training data, and as a result, the feedback passed to the Generator will not be helpful and will compromise the training convergence.

The authors of [27] proposed a new way to deal with smaller datasets and avoid those problems using data augmentation techniques. The augmentation is applied to the Discriminator input images, both fake and real images. This way, the Discriminator is fed with several augmented versions of real and fake images. The data augmentation techniques implemented are described above:

- **Translation** consists in shifting all pixels of the image in the X and Y axes. Moving the image will end up cutting out small parts of the image borders.
- **Cutout** consists of eliminating small square patches of the image at random locations. The implemented cutout operation is always applied after a translation operation.

These transformations are random, differentiable, and applied to the training data and the generated images. Figure 4.12 illustrates the implemented Data Augmentation techniques applied to a garment image.



Figure 4.12: Data augmentation applied to a garment image

4.3.4 Sampling Techniques

To improve quality during sampling, the CGG model uses two different techniques from the StyleGAN2 model. The techniques, Truncation Trick and Exponential Moving Average Generator are discussed below.

4.3.4.1 Truncation Trick

Often in the training data distribution, there are low-density areas. Those areas are poorly represented in the training data. There are not enough examples for the Generator to properly learn the data distribution in those low-density areas. That being said, style vectors w pointing to low-density areas in latent space W lead to poor quality generated images.

The truncation trick consists in generating new images from a smaller sampling space in which low-density areas are excluded. Practically, the truncation trick modifies the style vectors w in order to truncate the latent space W . The truncation trick follows the steps presented below:

1. The center of mass $\bar{w}$ of latent space W is calculated. $\bar{w}$ consists of the "average" style vector that is obtained by sampling a large number of styles from the Mapping network and calculating their average vector. To obtain a large number of styles first, it is necessary to pass through the Mapping network a large number of random noise vectors z' .
2. After $\bar{w}$ is calculated, the input style vector w is rescaled into a new vector w' what will be used as input for the synthesis network. This operation is described in equation 4.2

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

This truncation is a trade-off between quality and diversity. A hyperparameter named truncation psi ψ is used to control the quality and diversity trade-off. ψ can vary between 0 and 1, a ψ value close to 1 will lead to diverse images with good or poor quality. A ψ value close to 0 will lead to high-quality images but very similar to each other.

4.3.4.2 Exponential Moving Average Generator

Exponential Moving Average (EMA) Generator was first introduced in progressiveGAN [26] in order to generate samples with consistent quality throughout the training. It consists in creating a clone of the original Generator at the start of the training. The EMA Generator parameters are updated after every ten training steps, leading to less variance and higher quality in the generated images.

The EMA Generator is updated based on the exponential moving average formula, as presented in equation 4.3. P_{EMA} corresponds to the EMA Generator parameters and P to the original Generator parameters. The α variable is a hyperparameter known as Exponential Moving Average decay that controls the degree to which the parameters of the EMA Generator are updated during each training step.

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

4.3.5 Loss Functions

The Loss function used to optimize the training of the CGG model is the Hinge adversarial loss function [33]. The cross-entropy binary loss, which is the standard loss function used in GANs, was also implemented. The hinge adversarial loss for the Generator is defined in Equation 4.4 and for the Discriminator in Equation 4.5.

$$L_G = -E_{z \sim p_z} D(G(z)) \quad (4.4)$$

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

Knowing that x is an instance of the real data and z is an instance of random noise, in this equations:

- $D(x)$ is the probability estimated by the Discriminator of x being real.
- E_z is the expected value over all generated instances.
- E_x is the expected value over all real data instances.
- $G(z)$ is the Generator's output given noise z .
- $D(G(z))$ is the probability estimated by the Discriminator of a generated instance being real.

4.4 Pattern Transfer model

The second model implemented consists of turning fashion sketches with an overlaid texture patch into real garments images as illustrated in Figure 4.2. The Pattern Transfer model is based on TextureGAN model architecture. The majority of computer vision models use training images in RGB color space. The proposed model, similarly to TextureGAN, uses the LAB color space instead. The reasoning behind the use of **LAB** color space is that color constraints in RGB need to fight with the Network's understanding about semantics, e.g., bags are mostly black. LAB color space is comprised of a lightness channel L and two color channels, channel a and channel b . A representation of the LAB color space is presented in Figure 4.13.

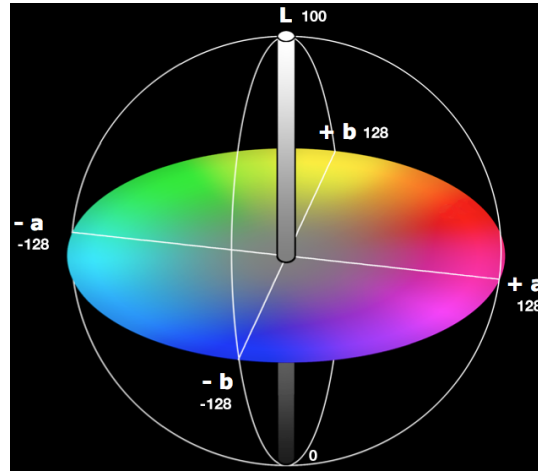


Figure 4.13: Representation of the LAB color space. Figure extracted from [12]

The Pattern Transfer model was tested with three different Generator architectures. Those architectures are presented in Subsection 4.4.1. The Discriminator architecture is similar to the one used in the CGG model and is detailed in Subsection 4.4.2. The adversarial loss function comprises several different losses detailed in Subsection 4.4.3.

4.4.1 Generator

The Generator learns how to generate realistic garments from input sketches with overlaid textures. A 5-channel image is used as input for the Generator. The channels comprise one channel for a greyscale fashion sketch, two channels for the texture, and two channels for color, as illustrated in Figure 4.14.

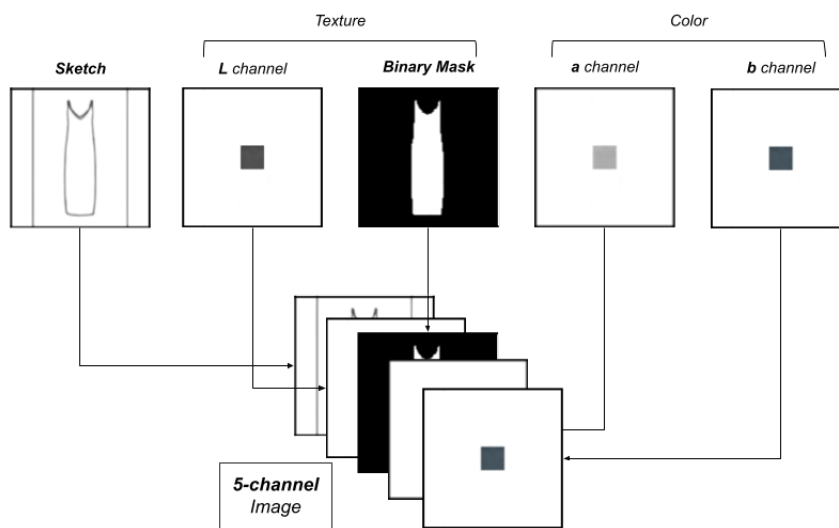


Figure 4.14: Overview of the Generator 5-channel input

The Pattern Transfer model was tested with three different Generators: Scribbler, TextureGAN, and Scribbler dilate 128. The TextureGAN and Scribbler dilate 128 Generators are modified versions of the Scribbler Generator. The description of each Generator is presented below.

The **Scribbler** Generator was proposed by Sangklkoy and his colleagues in 2016 [56]. The Generator architecture consists of several convolutions blocks, Residual blocks, and Upsampling Blocks. The model's inputs are first downsampled and then upsampled to the initial resolution. The output of the model is a 3-channel image (L,a,b) in Lab color space. The Convolutions blocks are comprised of 2D convolutions, always followed by batch normalization and ReLU activation functions. Figure 4.15 presents an overview of the scribbler Generator architecture. The Residual Blocks are formed by a convolution block followed by a convolution and batch normalization. The input of the residual block is then added to the result of the convolutions, and, at last, a ReLU activation function is applied. Figure 4.16 presents a diagram with the operations of each Residual block.

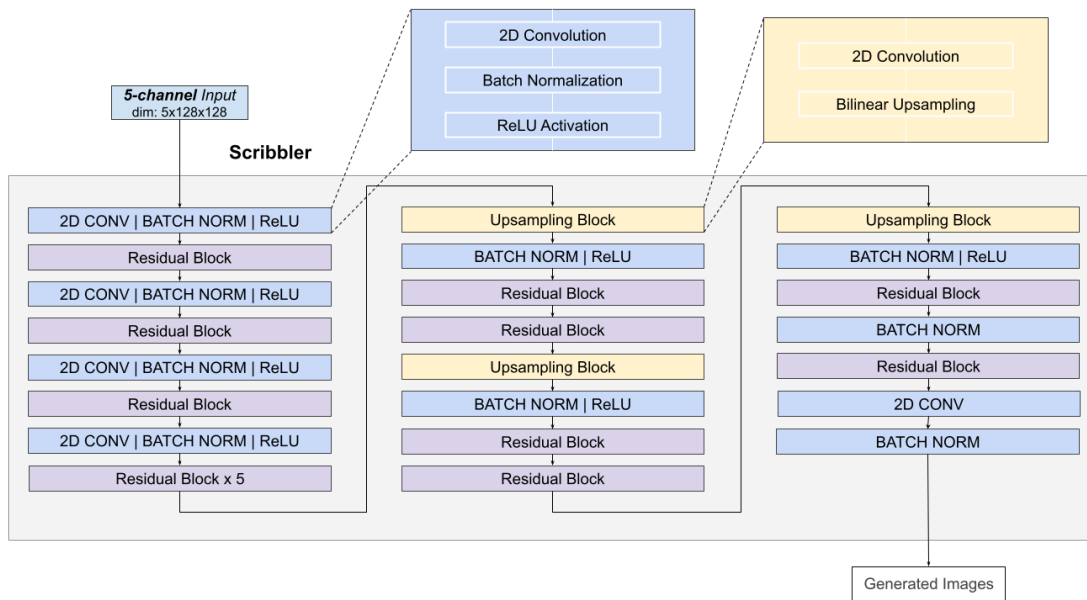


Figure 4.15: Overview of the Scribbler architecture

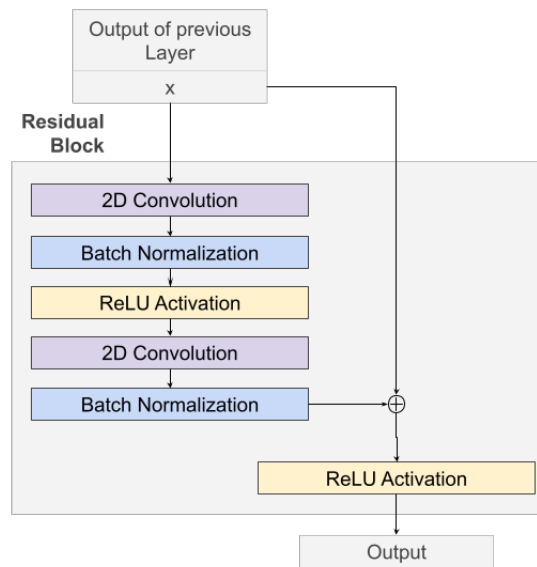
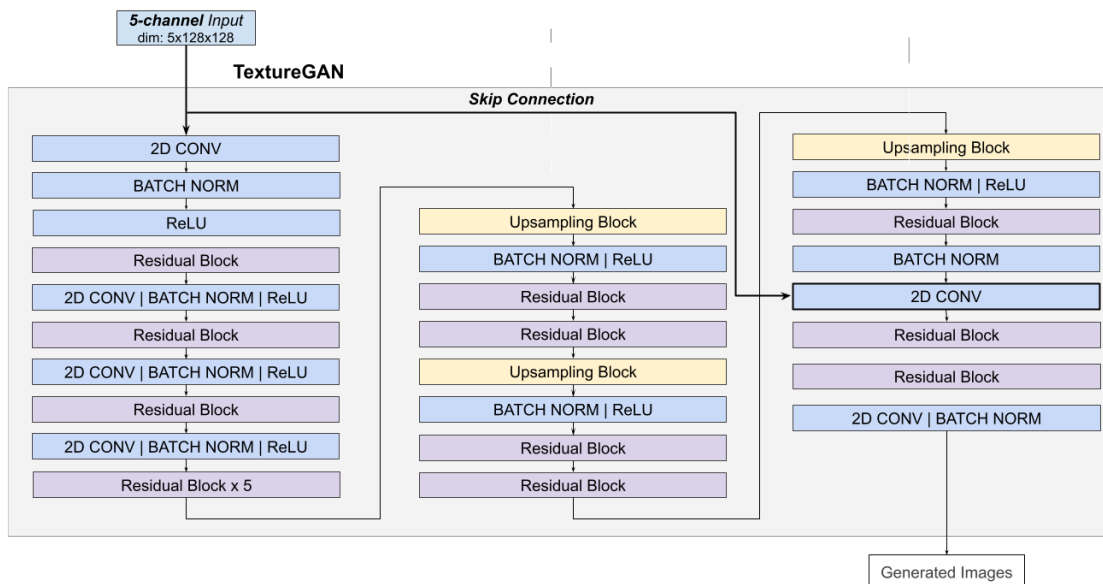


Figure 4.16: Scribbler Residual block architecture

The **TextureGAN** Generator was proposed by Xian and Sangkloy *et al.* [68]. The Network is based on Scribbler architecture with additional skip connections. One single skip connection is used directly from the input to a convolution operation. The architecture of TextureGAN is detailed in Figure 4.17.

Figure 4.17: Overview of **TextureGAN** Generator architecture

The **Scribbler Dilate 128** Generator, as the name suggests, is based on the Scribbler Generator. The first block is named Dilation Block and consists of a series of convolutions with a dilation of 5 pixels. A dilation of 5 indicates that there will be a spacing of 4 pixels between each kernel element in the convolutions operations, as illustrated in Figure 4.18. The architecture of Scribbler dilate Generator is presented in Figure 4.19



Figure 4.18: Comparison of two convolution operations with different dilation rates.

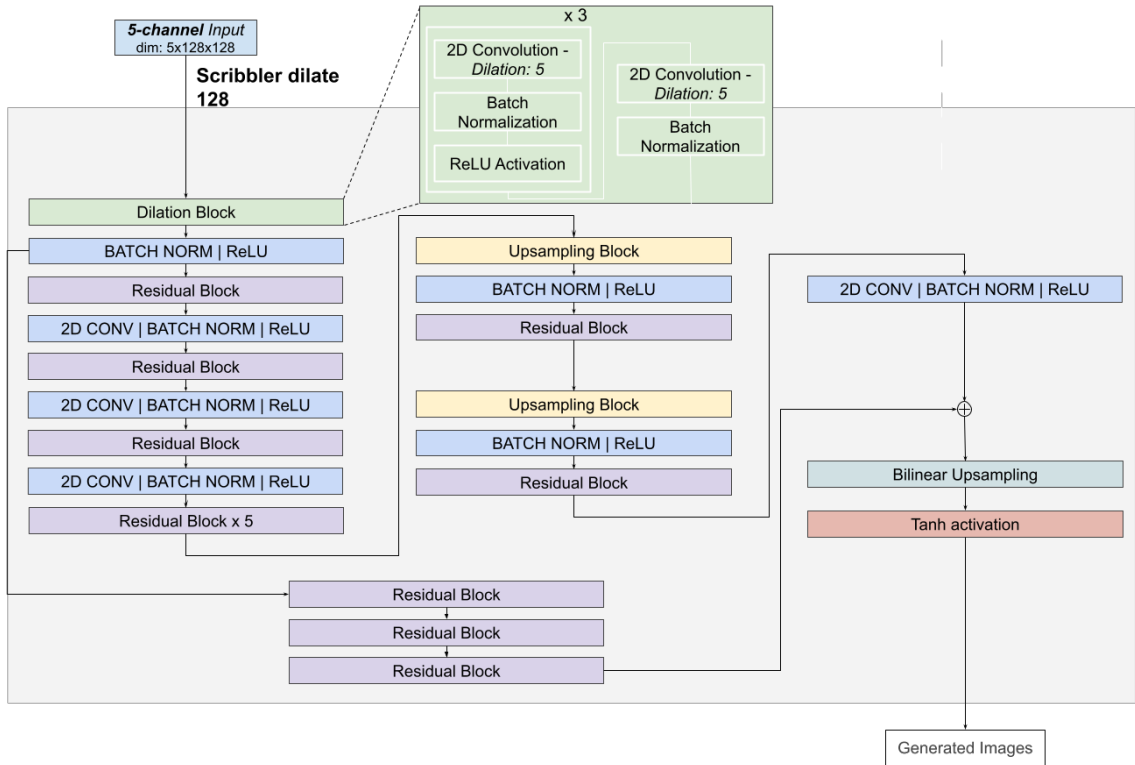


Figure 4.19: Overview of **Scribbler Dilate 128** Generator architecture

4.4.2 Discriminator

The Discriminator receives as input 3-channel images in Lab color space and outputs their prediction on the image's authenticity. The Architecture used is the same as the Discriminator used for the CGG model. Figure 4.20 shows a diagram of the Discriminator architecture. The blocks named "Discriminator Block" and "Discriminator Epilogue" were already explained above in Sub-section 4.3.2.

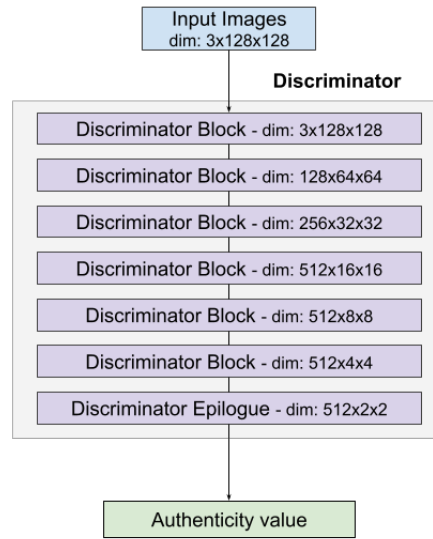


Figure 4.20: Discriminator network from the Pattern Transfer model

4.4.3 Loss Functions

The most important component of the Pattern Transfer model is the loss function. The loss function is based on the components of TextureGAN loss function. Five distinct losses are used to encourage the Generator to produce realistic and consistent garment images. The loss function components are feature loss, adversarial loss, style loss, pixel loss, and color loss. In Figure 4.21 an overview of the proposed model architecture and loss functions is presented. Each loss function component is carefully detailed above.

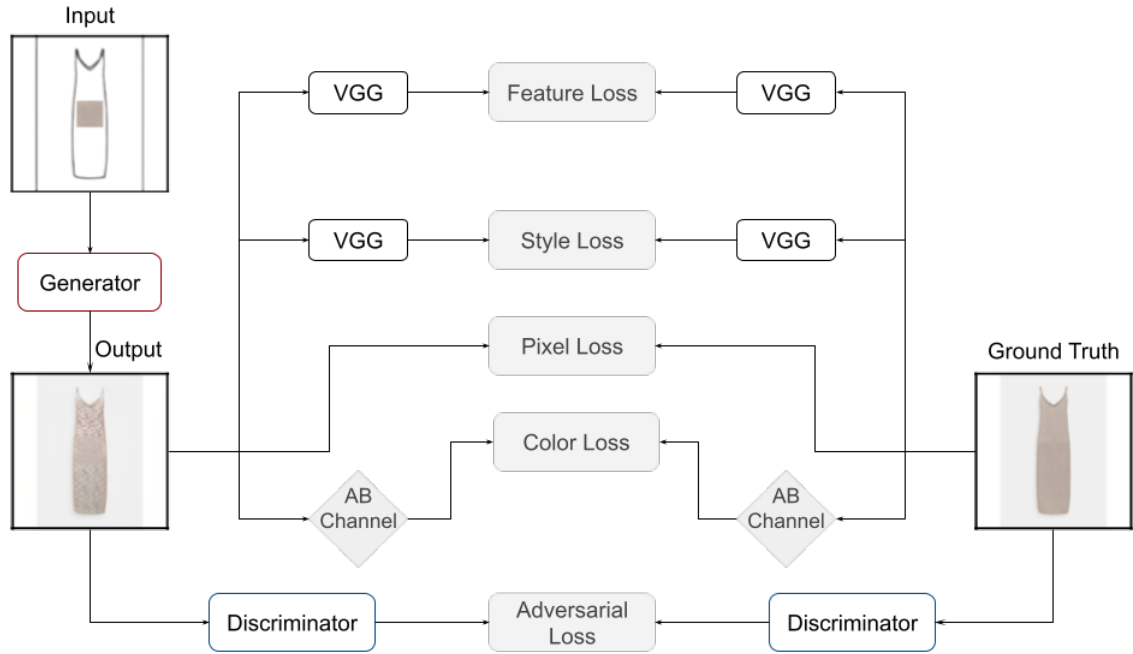


Figure 4.21: Overview of the Pattern Transfer model architecture

4.4.3.1 Feature Loss

The Feature Loss component L_F uses a pre-trained neural network known as VGG-19 [59]. It has been proved that the features extracted by VGG-19 middle layers represent high-level semantic information about the input images. The features extracted from VGG-19 middle layer "relu 4_2" were used to ensure a similar garment structure between generated and ground-truth images. Therefore, if the structures of the images are very different from each other, the Generator is penalized.

To calculate the Feature loss term, the ground-truth garments and the generated garments are converted from RGB to Lab color space, and for both images, a grayscale image is obtained using the Lab luminosity channel. The grayscale images are then fed into VGG-19 Network to extract features from real and generated images. At last, the Mean squared error (MSE) loss function is used to measure the differences between the real and generated images features extracted by VGG-19.

4.4.3.2 Adversarial Loss

The adversarial loss L_{adv} is used to ensure that the generated images are within the training image manifold. This way, the Generator is encouraged to synthesize realistic and sharp images. However, the Generator is limited by the diversity of the training data. The Generator will be incited to choose the most common colors in the training data for the generated images. Hence, the Discriminator understanding about color can conflict with the input images color constraints, e.g., if a dress sketch with a red color constrain is provided, and the training data does not have any

red dress, the Discriminator will think that a red dress looks fake and discourages the Generator from producing red dresses. The color loss function component was implemented as an attempt to mitigate this problem.

The adversarial loss uses the Hinge loss function described in subsection 4.3.5 to calculate the differences between the ground-truth images and the generated images in Lab color space.

4.4.3.3 Style Loss

The Style Loss component L_S is used to encourage the Generator to produce texture details. Unlike feature loss, it aims to capture small-scale details rather than the overall structure of the images. The style Loss is applied only to the L channel and consists in matching the Gram matrices of the features extracted from specific layers, namely relu3_2 and relu4_2, of the VGG-19 pre-trained Network. The Gram matrix $G_{ij}^l \in \mathbb{R}^{N_l \times N_l}$ is defined as:

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l \quad (4.6)$$

Knowing that N_l is the number of features maps in layer l in this equation:

- F_{ik}^l is the activation of the i th filter at position k of layer l .
- F_{jk}^l is the activation of the j th filter at position k of layer l .

4.4.3.4 Pixel Loss

The pixel loss component L_P consists in using the L channel of both ground truth and generated images to calculate the pixel loss. It leads to a more faithful generation of texture details. MSE loss function is used to measure the differences between the pixels of the ground truth and generated images.

4.4.3.5 Color Loss

To ensure that color constraints are correctly applied, a color loss component L_C was implemented. The color loss component makes use of the MSE loss function to calculate the differences in the *ab* color channels between the ground truth images and the generated images.

The objective function L of the proposed model is defined as:

$$L = w_F L_F + w_{ADV} L_{ADV} + w_S L_S + w_P L_P + w_C L_C \quad (4.7)$$

Where w_F , w_{ADV} , w_S , w_P and w_C are hyperparameters that control the strength of each loss function component.

Chapter 5

Experimentation

The present chapter outlines the experimental setup followed by the Conditional Garment Generation and pattern transfer models. The data collection process and the software libraries used in both experiments are outlined. Lastly, a comprehensive description of the user survey used to evaluate the performance of the Pattern Transfer model is given.

5.1 Data collection

The first experiments were initially carried out in Google Collaboratory ¹, since running the experiments in personal computers was not feasible given the computation requirements that the experiments needed. Powerful graphic cards and high amounts of video RAM are needed. Most personal computers cannot fulfill those requirements and conduct the experiments in a reasonable timeframe. In Google Colab, the code is freely executed in Google's cloud servers with a couple of limitations. The major limitation was that running the models with images over 64x64 of resolutions exceeded the amount of memory and RAM available, so the amount of computation needed to run the proposed models was higher than the memory Google Colab could provide.

To overcome this problem, the experiments were carried out in Microsoft Azure Machine Learning ². Azure Machine Learning is a cloud platform in which a wide range of virtual machines can be rented to conduct the experiments. XNFY Lab ³ kindly provided resources to conduct the experiments through the renting of Microsoft Azure virtual machines.

Microsoft Azure SDK was used for both experiments to obtain the generated images and the evaluation metric values. In the Conditional Garment Generation experiment, generated images and evaluation metrics were collected for each training epoch. In this case, the metric FID was calculated individually for each label using 12888 generated images and each label's total amount of images. For the Pattern Transfer experiment, the evaluation metric SSIM was calculated at each

¹<https://colab.research.google.com/>

²<https://azure.microsoft.com/pt-pt/services/machine-learning/>

³<https://xnfy.com/>

training epoch using 1600 generated and the correspondent ground-truth images. Furthermore, the generated images, input images, and ground truth images were obtained for each training epoch.

5.2 Libraries Used

The proposed models were implemented using the Visual Studio Code editor and Azure Repos ¹. The CGG and the Pattern Transfer model are both implemented in Python 3.8. and depend on the software libraries presented below:

- **Jupyter Notebook:** To set up the model's hyperparameters in order to run the experiments in Microsoft Azure.
- **Microsoft Azure SDK:** To display the generated images and evaluations metrics throughout training.
- **Pandas:** To process and prepare the .csv files of the training datasets.
- **PyTorch:** Open source machine learning library used to implement and train the proposed deep learning models.
- **Torch Vision:** To visualize the generated images and apply image transformations in the training data. In the Pattern Transfer model, Torch Vision is also used to load a pre-trained version of VGG-19 [59] network.

5.3 Experiments

In this section, the conducted experiments are described in detail. For each experiment, the training dataset, the training parameters, and the evaluation metrics used are presented.

5.3.1 Experiment 1: Conditional garment generation

The first experiment carried out consists of training the proposed CGG model and evaluating the obtained results. The goal of this experiment was to successfully train the CGG model to generate different types of garments according to the type of garment requested by the user.

5.3.1.1 Datasets Used

The dataset used to train the conditional image generation model contains over 42,000 images of clothing items obtained by scrapping e-commerce clothing stores. Each image in the dataset is associated with a label that indicates the garment type. Furthermore, all the images in the dataset feature a single piece of clothing on a white background. Those attributes simplify the generation of noiseless images since the model only targets the generation of garments rather than

¹<https://azure.microsoft.com/en-us/services/devops/repos/>

the generation of different backgrounds, people’s faces, and bodies. In total, the dataset contains images from 19 different types of garments. All the labels and the number of images per label are described in Table 5.1. The labels information is provided in a CSV file with the format shown in Table 5.2. Figure 5.1 presents a sample of images extracted from the dataset.

Label Name	Number of images
Blazers	2 001
Blouses	2 905
Bodysuits	317
Cardigans	2 785
Dresses	6 236
Dungarees	68
Hoodies	629
Jackets	2 001
Jumpsuits	1 192
Polo Tops	623
Shirts	2 926
Shorts	3 306
Skirts	5 257
Sweaters	2 542
Tops	1 949
Trousers	2 235
Jeans	2 286
Leggings	1 038
T-shirts	1 991
Total	42 287

Table 5.1: Number of images per label

File Name	Type of garment
img01.jpg	t-shirt
img02.jpg	skirt
img03.jpg	dress
...	...

Table 5.2: Dataset csv file format

5.3.1.2 Training parameters

The CGG model was trained for 170 epochs with the dataset described above. The Adam optimization method was used with a *learning rate* = 0.002, $\beta_1 = 0$ and $\beta_2 = 0.99$. Path Length regularization was applied every 32 training steps with a weight of 1. Additionally, at every 20 training steps, R1 regularization was applied with a weight of 10. The data augmentation probability remained constant over all epochs at 0.22.



Figure 5.1: Sample of images from the multi-garment dataset used for the training of the CGG model

5.3.1.3 Evaluation Metric

The metric Fréchet Inception Distance (FID) was used to evaluate the model performance. Fréchet Inception Distance (FID) [4] compares the statistics of the generated images to the real images, instead of evaluating the generated images. So, it can compare the similarity between them. A lower FID indicates smaller differences between the data distribution of real and generated data. In other words, it points to a higher similarity between them. The FID metric was calculated for each label. A total of 19 different FID values were collected in each training epoch.

5.3.2 Experiment 2: Pattern Transfer

The second experiment carried out consists of training the Pattern Transfer model and evaluating the obtained results. This experiment aimed to successfully train the Pattern Transfer model to convert dress fashion sketches into real dresses images.

5.3.2.1 Datasets Used

The dataset used in this experiment is a paired dataset of dresses with over 25k images. Each pair is made up of an image of a dress and the correspondent fashion sketch. The dress's images were collected by scrapping several e-commerce retailers. Given that multiple retailers' websites were scrapped, each dress image contains a single dress in a white or grey background. Figure 5.2 displays a sample of image pairs that belong to the dataset.



Figure 5.2: Sample of images from the dresses paired dataset used to train the Pattern Transfer model

Given the difficulty of collecting real pairs of images, the edges extracted from dress images were used as fashion sketches. Extracting the edges by hand would be possible but very time-consuming. Therefore, choosing this option would be impractical given the amount of time available. In order to extract the edges of the images automatically, several deep learning models for the edge detection task were identified. A pre-trained version of DexiNed [62] model was used to extract the edges from the dress images. DexiNed is the state-of-the-art model for the edge detection task, and fortunately, the author published a pre-trained version of the model, which was crucial in the creation of this dataset.

While extracting the edges from dress images, was verified that the extracted edges sometimes did not match our expectations. To make the fashion sketch as realistic as possible, it was wanted

the extracted edges to match up with the seams, folds, and contours of the dresses as presented in Figure 5.3a and Figure 5.3b. However, in some cases, the dress fabric pattern would be extracted as well as presented in Figure 5.3c. For dresses made of one-colored fabrics or fabrics that incorporate smaller elements, the model would generate suitable fashion sketches, while for dresses made of fabrics with large elements, the model would generate noisy fashion sketches. Since the number of images in the dataset is fairly large and it is not feasible to modify them by hand, it was decided to use all of the images to ensure more diversity. Although, it may slightly affect the model's performance, given that some fashion sketch images have noisy pixels. Figure 5.3 shows three images generated by DexiNed model. The Figures 5.3a and 5.3b meet the expectations, while the Figure 5.3c does not, given that the fabric pattern is extracted as well.

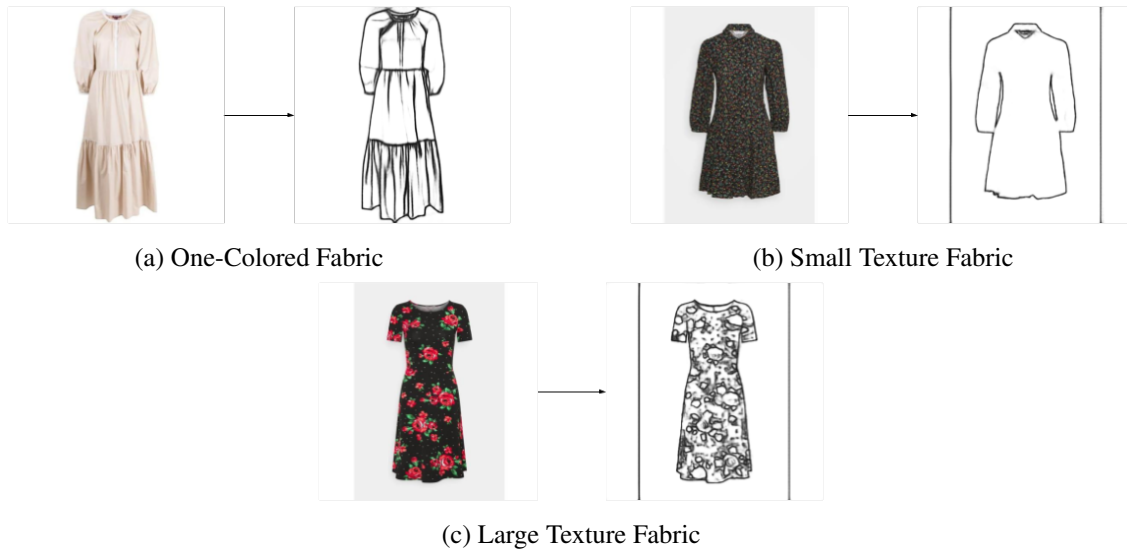


Figure 5.3: Images obtained using the pre-trained version of the DexiNed Model

5.3.2.2 Training parameters

In this experiment, six distinct training runs were performed with some differences in the hyperparameters. The Adam optimization method was used for all training runs with different learning rate values for the generator and the discriminator networks. For the generator the learning rate was $lr_g = 0.0001$ and for the discriminator it was $lr_d = 0.0002$. For both networks the Adam method used $\beta_1 = 0.5$ and $\beta_2 = 0.90$. Each training run was executed for 35 epochs with a batch size of 32 images and at 128x128 of image resolution. The minimum size of the texture patch's height and width was set to 20 pixels, and the max size was set to 45 pixels. Regarding the loss function component weights, both experiments had $w_{pixelL} = 10$, $w_{adversarial} = 1$, $w_{feature} = 1$ and $w_{style} = 1$. The hyperparameters changed in the six training runs were the type of generator and the weight of the pixel ab loss function component. Those experiment setups are depicted in Table 5.3.

Training Run	Generator	Pixel Weight $\alpha\beta$
1	Scribbler	70
2	Scribbler	100
3	TextureGAN	70
4	TextureGAN	100
5	Scribbler dilate 128	70
6	Scribbler dilate 128	100

Table 5.3: Hyperparameters changed between the training run of the proposed Pattern Transfer model

5.3.2.3 Evaluation Metric

The metric Structural Similarity Index Measure [66] (SSIM) was used to evaluate the Pattern Transfer model performance. The SSIM computes the similarity between two images. The metric is calculated several times in non-overlapping square regions of the two images. The final metric value is average SSIM over all regions. The metric values range from 0 to 1. The closer to 1, the more similar the images are. At each epoch, the metric is calculated across 50 batches of generated images and ground truth images. The metric is calculated using Equation 5.1:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (5.1)$$

Knowing that x and y are two non overlapping squared regions of the two images:

- μ_x is the average pixel values of squared region x .
- μ_y is the average pixel values of squared region y .
- σ_x is the variance of the pixel values of x .
- σ_y is the variance of the pixel values of x .
- σ_{xy} is the covariance between the pixel values of region x and y .

5.4 User Survey

A user survey was carried out to evaluate the performance of the proposed Pattern Transfer model. Several images were generated from the Pattern Transfer model to create the user survey. These images were generated using the training data fashion sketches with textures extracted from the ground-truth images and textures from a separate texture dataset.

The user survey aimed to evaluate the community's visual perception of the generated images and their marketability. The user survey is divided into five different sections. The first four sections are used to evaluate the quality of the generated images. The last section is used to evaluate the marketability of the generated garments. Each section is detailed below:

- **Section I:** comprised of only 3 linear scale questions. Participants were asked to assess the realism of the generated dresses presented on a scale of 1 (Not at all realistic) to 5 (Very realistic).
- **Section II:** comprised of 3 multiple-choice questions and one linear scale question. In each multiple-choice question, Participants were presented with two different garments and were asked to identify the generated one. The linear scale question aims to assess the difficulty the participants had in identifying the generated garment. Participants were asked to rate the difficulty level on a scale of 1 (Very Easy) to 5 (Very Hard).
- **Section III:** comprised of 3 multiple-choice questions, one open-ended question, and one linear selection question. In each multiple-choice question, participants were presented with a Fashion sketch and two identical dresses with different patterns and were asked to identify which image was generated from the fashion sketch presented. Participants were asked to evaluate the level of difficulty in the generated image identification. In the last question, they were asked to identify which artifacts they had found on the generated images that lead to their decision.
- **Section IV:** comprised of 3 single selection questions and two open-ended questions. In each multiple-choice question, participants were presented with two pairs of images. Each pair comprised a Fashion sketch with an overlaid texture patch and a dress that could be real or generated. Participants were asked to distinguish between the generated and real dresses. In the last question, participants had to explain what artifacts they had found on the pairs that led to their decision regarding which of the pairs had a generated image.
- **Section V:** comprised of 3 linear scale questions to evaluate the marketability of the generated images. In each question, participants were asked to express their opinion regarding the marketability of the generated dresses. Participants were asked to rate the sentence "Customers are likely to buy the generated dress" on a scale of 1 to 5, in which one means Strongly Disagree and five means Strongly Agree.

The user survey questions can be found in Appendix [A](#). Students of Faculty of Engineering of the University of Porto and members of Microsoft Portugal and XNFY Lab were asked to participate in the user survey. A total of 40 volunteers participated in the user survey.

Chapter 6

Results and Discussion

This chapter outlines the results obtained from the experiments conducted. The results of the CGG model and the Pattern Transfer model are carefully analyzed. In addition, the results of the user survey are also detailed and analyzed.

6.1 Experiment 1: Conditional garment generation

In this section, the results of the conditional garment generation experiment are presented. The experiment was evaluated using the metric FID. The FID was calculated per training epoch for each label using 12,288 generated images and all training images belonging to the specific label. The closer the FID value is to 0, the more similar the two sets of images refereed above will be.

Figure 6.1 presents dresses, skirts, and trousers generated by the conditional garment generation model throughout the training. By Analysing the generated images, we can conclude that the perceptible visual quality is quite good and increase during training, as can be observed in Figure 6.1.

The metric FID was calculated individually for each type of label. Considering that 19 different labels were used, it was not worth analyzing all metrics in detail. A more detailed analysis of the FID values for the label dress, skirt, and trousers will be presented. Figure 6.2 presents the FID values of the labels dress, skirt, and trousers throughout the training.

Overall, in the first 10 epochs, the metric value dropped by half. It dropped from 430 to 200, showing some variance along the way. From epoch 10, the metric kept dropping but at a slower speed. The FID for label Skirt remained stable with a value around 100 between epoch 50 and epoch 100. A slight increase occurred from epoch 100 to epoch 150, where the FID value has stabilized near the value 115. The FID values for the label's dresses and trousers decreased until epoch 75 and remained stable from then on, both metrics with values below 100. At the end of the training, dresses presented an FID value of 90, trousers an FID value of 84, and skirts an FID of 111.



Figure 6.1: Generated images collected throughout the training from the CGG model.

Given the perceptible quality of the generated images, it would be expected for the FID value to be considerably smaller. That being said, a comprehensive analysis of the experiment results was made in order to pinpoint the problem.

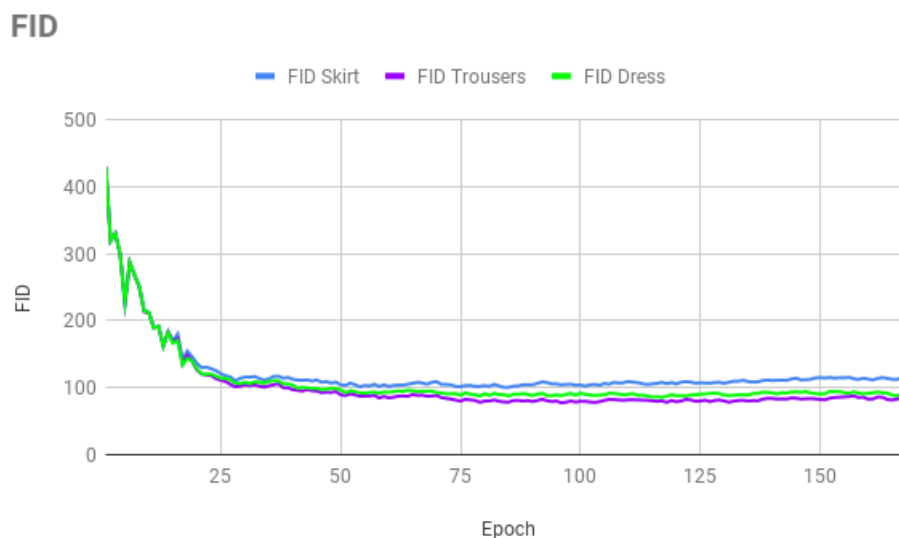


Figure 6.2: FID metric per training epoch for the labels Skirts, Trousers and Dresses

For all labels, the FID values started at 430. At the end of the training, depending on the label, the FID values ranged from 30 to 111. In an ideal scenario, this high variance between the FID values at the end of training should not happen, and the FID values should be closer to each other. Theoretically, a label with an FID of 30 should have better-generated images than a label with an FID of 111, however that was not observed. The labels that presented the lower FID values were the labels cardigan, blazer, and jacket. The number of images of each label in the training dataset is approximately 2000. A lower FID value does not necessarily mean that the obtained results are promising. Most of the images being generated from these labels do not fit within the label characteristics. In addition, the perceived quality of the generated images is the same for all labels.

The model is supposed to learn throughout the training to generate the garments according to certain conditions. The conditions that the model needs to learn are represented by the labels associated with each image of the training dataset. In this case, the condition is the type of garment. That being said, it is expected for the model to make mistakes at the beginning of training. Therefore, the model is more likely to generate images with characteristics that do not fit within the specific label in the first training epochs. The number of wrongly generated images is expected to reduce throughout the training until none are wrongly generated.

In other words, the model should learn that modifying the label vector while using the same latent vector implies changing the type of garment being generated. In labels with a higher amount of training images, such as dresses, skirts, and trousers, it can be observed that the shape of the generated garments is gradually being changed until it reaches the shape of the garment that is meant to be generated. This finding was made by analyzing images of different types of garments generated from the same latent code. The latent code of the generated garments only differs in the digits that identify the type of garment to be generated. Figure 6.3 shows the evolution of dresses,

skirts, and trousers generated from the same latent code throughout the training. It is noticeable that the image's shape gradually changes until it fits within the label to which the image belongs.

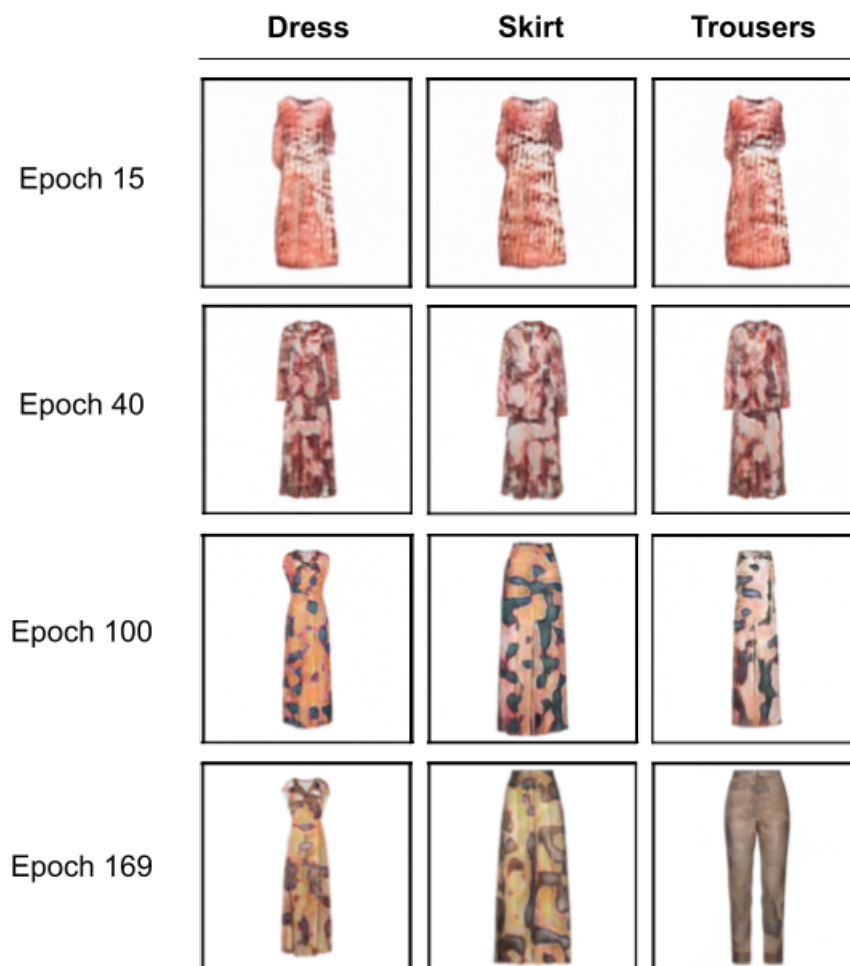


Figure 6.3: Different types of garments generated throughout the training from a common latent vector

Once it was concluded that the model is headed in the right direction, the hypothesis of the problem being in the implementation of the model was excluded. Since the problem does not seem to be in the model implementation, it must be related to the training dataset used.

Although 19 different labels have been used, many of them could be grouped due to their similarities. For humans, it is easier to distinguish a blouse from a t-shirt, but for computers, the difference is almost non-existent. The dataset labels can be grouped into three different sets based on their image shape similarities. The three sets are upper-body garments, lower body garments, and whole-body garments. In the upper body garments, any kind of jackets and shirts are included. Lower body garments include skirts, trousers, and shorts, and whole-body garments are comprised of dresses, jumpsuits, and dungarees. These three sets have many similarities, especially in the shape of the garments.

The upper body garments group 11 different labels were used, which gives a total of 20 669 images which correspond to almost half of the dataset. Given the size, the model will be encouraged to generate more upper body garments. The total of whole body garments is 7496, which corresponds to 18% of the training images. The lower body garments comprise 14 122 images, 5559 trousers images, 5257 skirts, and 3306 shorts.

It was observed that the label that obtained more garments correctly generated, that is, more garments that fit within the label, was the label dresses. The label with the higher number of images in the training dataset was the label dresses. Thus, we concluded that the number of images per label significantly impacts the model performance.

Despite most generated images being dresses, some were not, which explains the FID value obtained at the end of training. The metric FID was compromised because of the presence of other types of garments. The presence of those garments makes the FID value higher than it should be, considering the perceived quality of the generated dresses.

All upper body garments presented the lower FID values at the end of training. In addition, most of the generated garments did not belong to the label they should belong to. The FID calculated from different upper-body garments will always be lower than those calculated from dresses, trousers, and shirts. The reasoning behind it is the shape; upper-body garments all share a very similar shape—upper-body garments presented at the end of training an FID between 33 and 67. The model has learned that those labels should generate upper-body garments but not which type of garment exactly. Thus, the FID value is smaller even though the correct type of garments are not being generated.

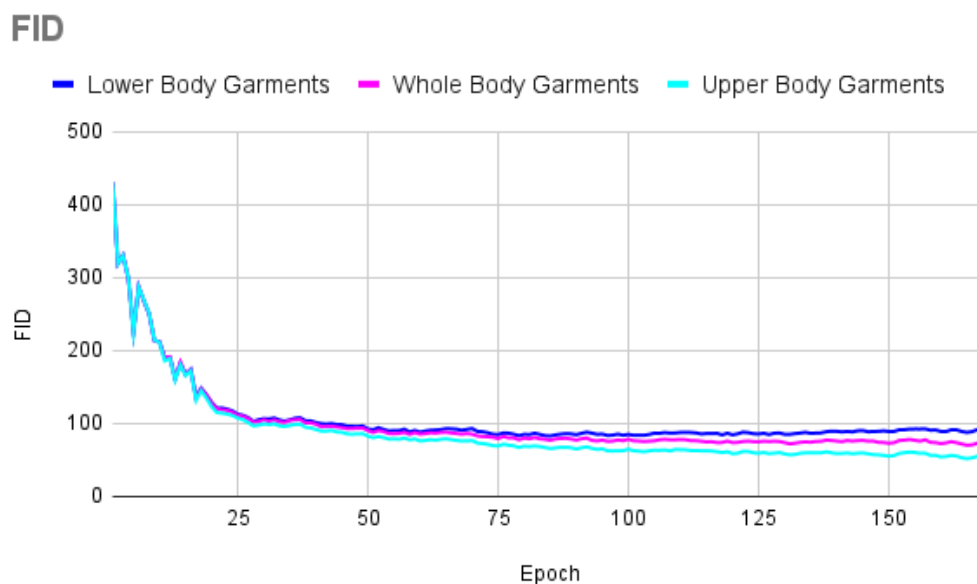


Figure 6.4: FID metric per training epoch for Whole Body, Lower Body and Upper Body garments

Figure 6.4 presents the average FID values for lower-body, whole-body and upper-body garments. The upper-body garments FID is represented in light blue color, the whole-body garments in pink, and lower-body garments in dark blue. By analyzing the figure, we can confirm that upper-body garments have the lowest FID values. In order to improve the model performance, a more balance dataset should have been used. The dataset should have an even number of images for each label and at least 6000 images per label to ensure that the model performs as expected. In addition, the training data had different labels for very similar types of garments (*i.e.* shirts and t-shirts) that should have been grouped to improve the model performance.

6.2 Experiment 2: Pattern Transfer

In this section, the results of the six Pattern Transfer training runs are presented and analyzed. The metric SSIM (Structural Similarity Index Measure) was used to evaluate the training runs performance. The SSIM metric was calculated throughout the training over 1600 generated images and ground-truth images. The metric ranges from 0 to 1, the closer to 1, the closer the sets of images being evaluated are. If the metric returns one, then the generated images are equal to the ground truth images. Thus, the higher the value, the higher the performance of the model. Six training runs were conducted with different hyperparameters values. The hyperparameters changed were the weight of the loss component pixel ab and the type of generator used. The results of the different training runs are presented in table 6.1. The SSIM values for all training runs vary from 0.813 to 0.859. For all training runs, the metric values are very close. It can be observed from training runs 5 and 6 that the training runs with worse performance use the generator Scribbler Dilate 128. The training run that presented better results were training run 3 and 4. Both training runs use the TextureGAN generator, which leads us to conclude that the TextureGAN generator performs better than the other generators. Using the value 100 in pixel weight ab seems to increase the model performance when using the Scribbler and the Scribbler dilate 128 generators. However, the training run that achieved the best results of 0.859 used the TextureGAN generator and a pixel weight ab of 70.

The Scribbler dilate 128 generator shows the worse performance leading us to conclude that the use of dilated convolutions does not help improve the model performance.

The pixel ab weight value has some influence on the SSIM value obtained. Nevertheless, to deeply understand the influence of pixel ab weight, more training runs with different weights needed to be conducted.

Figure 6.5 shows images generated by each generator at the epoch with the best SSIM. The visual perception of the images generated by the scribbler generator and the TextureGAN generator is quite similar. Nevertheless, it is noticeable that the images generated by the Scribbler Dilate 128 generator seem to be more blurred than the other two images.

A detailed analysis was conducted to evaluate the model's performance at converting fashion sketches to dresses. First of all, it was studied the model performance in generating one-colored

Training Run	Generator	Pixel Weight ab	Epoch	SSIM
1	Scribbler	70	28	0.841
2	Scribbler	100	33	0.844
3	TextureGAN	70	33	0.859
4	TextureGAN	100	25	0.845
5	Scribbler dilate 128	70	26	0.805
6	Scribbler dilate 128	100	28	0.813

Table 6.1: Hyperparameters changed between the training run of the proposed Pattern Transfer model



Figure 6.5: Garments generated by Scribbler, TextureGAN and Scribbler dilate 128 generators from the training runs with the highest performance

dressess and secondly, in generating dresses with patterns and multiple colors. The generated images and metrics from training run 3 were used to conduct this analysis.

Figure 6.6 shows one-colored dresses generated from the Pattern Transfer model in training run 3. On the left, the Fashion sketch is presented, in the middle, the generated garments, and on the right, the corresponding ground truth images are presented.

Although the generated garments being very identical to the ground-truth garments, some shading is lost. Hence, the ground-truth images present more contrast than the generated images. The model fails in applying the garment color constraints for colors that are poorly represented in the training dataset images, as can be seen in Figure 6.7. This way, the color applied is slightly different from the one in the input texture patch. In some cases, minor color stains can be found in the background of the images.

Considering the dataset used and the problems mentioned in Chapter 5, only a few dresses with patterns have a suitable fashion sketch. For patterns with small elements, the model performs well, as can be seen in Figure 6.8. The model can propagate the texture throughout the whole garment. Nevertheless, it sometimes fails to produce the correct color. Dresses with fabrics with large elements have a noisy fashion sketch. Thus, the model did not learn how to generate the dresses properly. For patterns with large elements, the model expects the fashion sketch to have

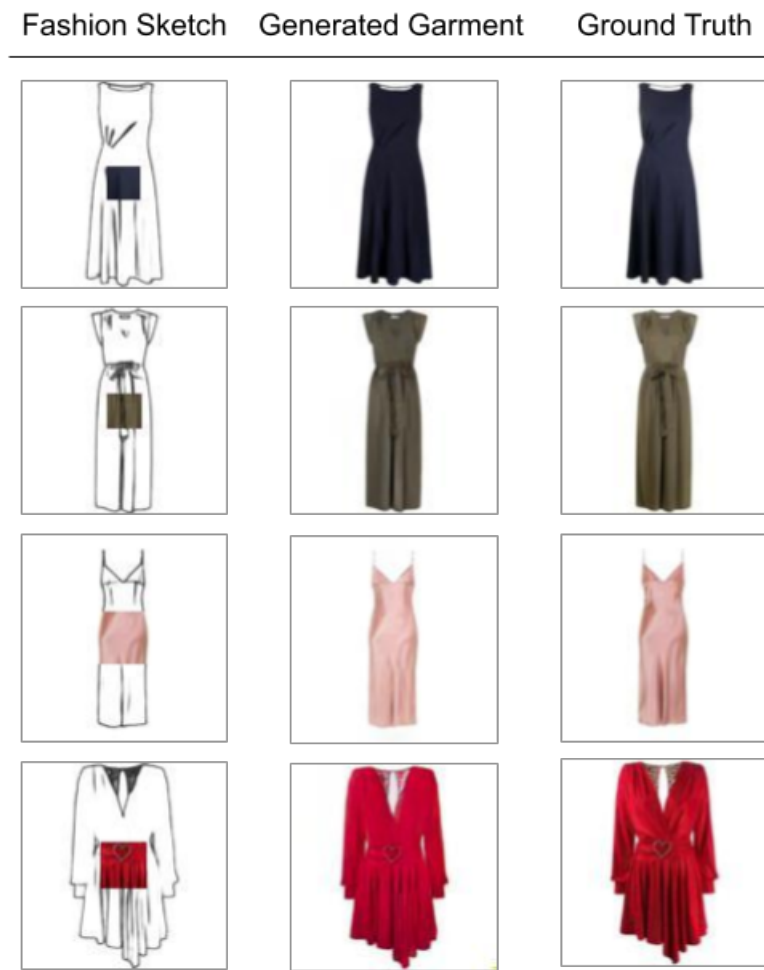


Figure 6.6: Pattern Transfer results for the training run 3 model using single colored dress

the fabric patterns throughout the whole sketch. So, when applying a texture to a new fashion sketch, the model fails to propagate the texture through the garment as presented in Figure 6.9.

The SSIM metric values throughout the training are very volatile. The SSIM value has changed considerably over the epochs, as can be seen in Figure 6.10. Over the course of training, the SSIM showed sharp ups and downs, reaching the highest value at epoch 33. The generator Loss has drooped exponentially from epoch 1 to epoch 13 and stabilized from then on, as can be observed in figure 6.11. Considering the SSIM values observed, it can be concluded that the implemented Pattern Transfer model is not stable. Nevertheless, it can generate very realistic images and successfully convert Fashion sketches into images of garments.



Figure 6.7: Pattern Transfer results which failed in applying the color constraints

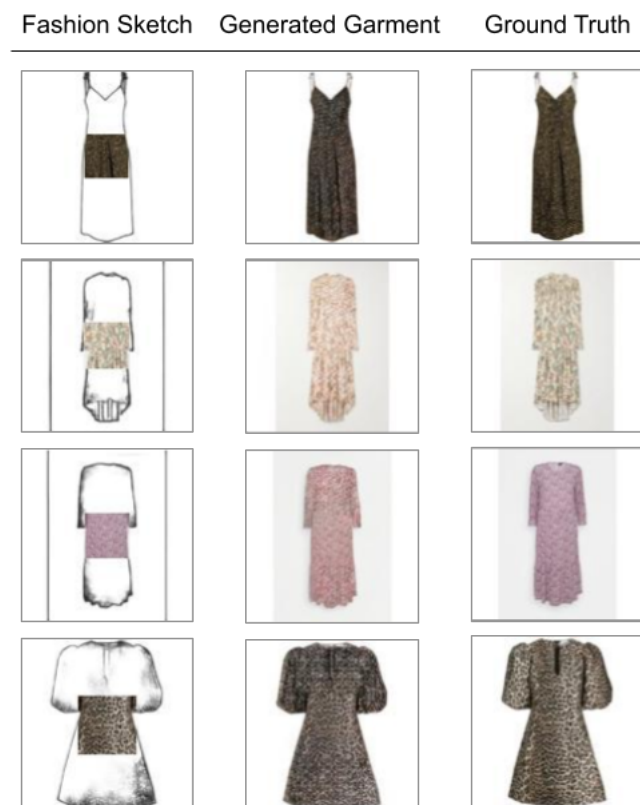


Figure 6.8: Pattern Transfer results for the training run 3 model using dresses with patterns

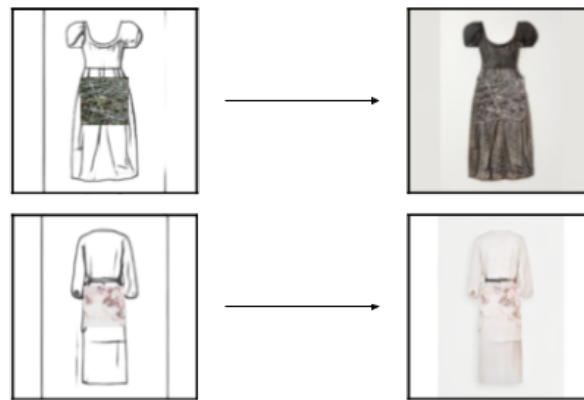


Figure 6.9: Pattern Transfer results that didn't propagate correctly the textures through the dress

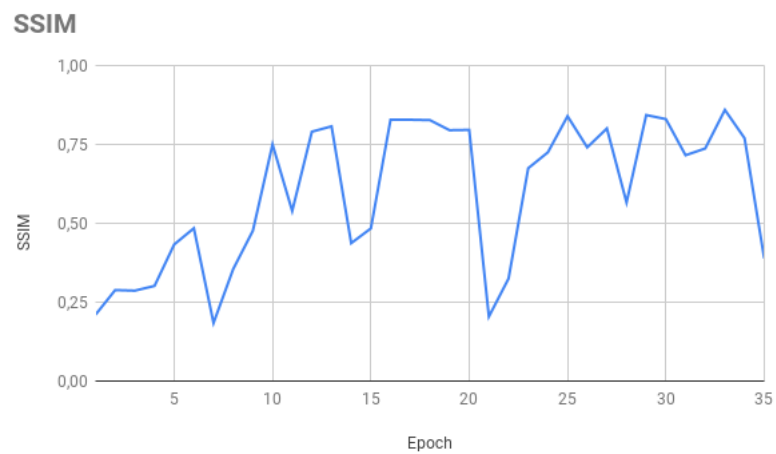


Figure 6.10: SSIM metric per training epoch for the training run 3 model

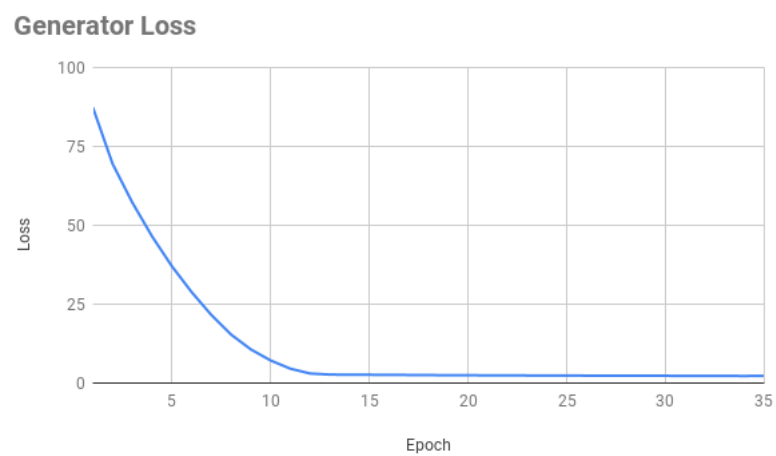


Figure 6.11: Generator Loss per training epoch in training run 3

6.3 User Survey

This section discusses the results of the user survey outlined in Section 5.4. The complete user survey can be found in Appendix A. The user survey aimed to evaluate the performance of the Pattern Transfer model in terms of the realness and marketability of the generated garments. The user survey comprises five different sections. The first four sections aim to assess the realness of the generated images given different levels of information to the participants in each section. The fifth section aims to evaluate the marketability of the generated garments.

A total of 40 individuals have participated in the user survey. The results of each user survey section are discussed below.

6.3.1 Section I - Visual perception of generated garments

The first section aims to evaluate the realness of the generated images. Participants were asked to rate on a scale of 1 to 5 the realness of three generated images. On average, 64% of the participants evaluated the images as realistic and very realistic. 21% of the participants rated as neutral, and 15% considered the images to be unrealistic. Figure 6.12 illustrates the participant's answers to Section I questions. On average, the rating of the generated image's realness was 4 (*i.e.*, realistic), leading us to conclude that the generated images look real.

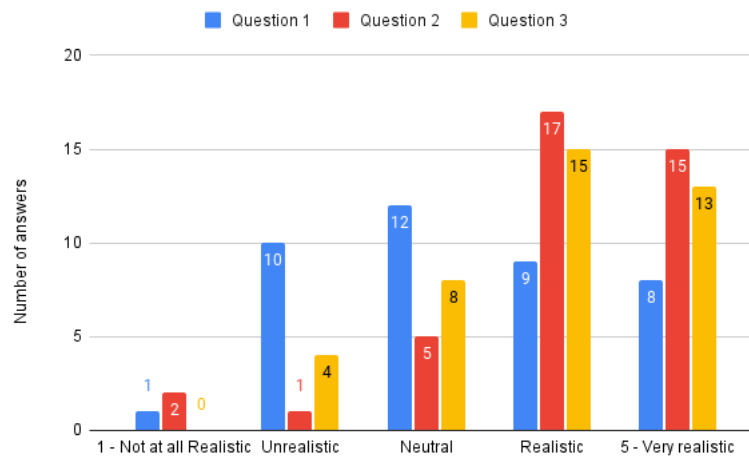


Figure 6.12: User survey answers for Section I

6.3.2 Section II - Identify the generated dress

In Section II, participants in each multiple-choice question were asked to identify the generated image. Each question comprised a real dress and a generated dress. Table 6.2 presents the percentage of participants who answer correctly to each multiple choice question. Most participants failed to identify the generated image on question 1. More than half identified correctly the generated image in question 3, while only 5% of the participants failed to identify the generated image in

Question	Participants who answered correctly(%)
1	25%
2	95%
3	55%

Table 6.2: Percentage of correct answers for Section II multiple choice questions

question 2. On average, the percentage of correct answers per participant was 58%. Each participant answered 1,74 questions correctly out of 3. Only 5% of the participants answers correctly to all of the questions presented in this section. The participants were also asked to evaluate the difficulty in identifying the generated garments. 77,5% of the survey participants considered hard and very hard the task of identifying the generated images while 7,5% considered easy and 15% considered neutral. On average, the participants considered it difficult to distinguish between real and generated images.

6.3.3 Section III - Fashion sketch to dress I

Section III comprised three multiple-choice questions, one short answer question, and one linear scale question. For each multiple-choice question, participants were presented with a fashion sketch and two dress images. Participants were informed that one of the dresses was generated by our model, and the other was the dress from which the fashion sketch was extracted. The aim of those questions was to verify if the presence of the fashion sketch could have any influence on the participant's answers. In line with the previous section questions, participants were asked to identify the generated dress. The percentage of correct answers can be found in Table 6.3

Question	Participants who answered correctly(%)
1	75%
2	52,5%
3	70%

Table 6.3: Percentage of correct answers for Section III multiple choice questions

Less than half of the participants failed to identify the generated image on question 2. 25% of the participants failed to identify the generated image in question 1 and 30% in question 3. On average, the percentage of correct answers per participant is 65,8%. Each participant answered 1,97 questions correctly out of 3. Only 22,5% of the participants answers correctly to all of the questions presented in this section. It was concluded that having information about the fashion sketches facilitates identifying the generated garments.

The participants were also asked to rate the difficulty in identifying the generated garments. 82,5% of the survey participants considered hard and very hard the task of identifying the generated images. 5% considered easy, and 12,5% considered neutral. Participants considered it more challenging to identify the generated dress when the fashion sketch employed is known.

The short answer question aimed to find what artifacts and characteristics participants have identified in the generated images that led their decision regarding the images chosen as the generated ones. Given that not all of the justifications were useful for identifying the generated images' quality problems, some were ignored. Thus, only 32 justifications were selected. The justifications pointed out by participants were:

- **Lack of shadows:** 6 justifications (19%). The participants identified a global lack of shadows in the generated garments with relation to real garments.
- **Images with less details:** 6 justifications (19%). Overall, participants have noticed that generated garments had fewer details compared to real garments.
- **Poorly defined contours:** 5 justifications (16%). Participants have found the contours of the generated garments to be blurred and poorly defined.
- **Image Blur:** 3 justifications (9%). Participants considered that the generated garments are blurrier and have less definition when compared with real garments.
- **Artificial brightness:** 4 justifications (13%). Participants considered the generated garments' brightness to look artificial compared to the brightness of real garments.
- **Folds position:** 4 justifications (13%). Participants have found that not all of the folds present in the fashion sketch were present and correctly placed in the generated garments.
- **Inconsistent colours:** 4 justifications (13%). Participants have found some inconsistencies in the generated garments colors. In other words, they found inconsistencies in the color tones of the garments containing some areas with lighter tones and some areas with darker tones.

6.3.4 Section IV - Fashion sketch to dress II

In this section, participants had to answer three multiple-choice questions and justify their answers in 2 of them. For each multiple-choice question, the participants were presented with two pairs of images. Each pair had an image of a fashion sketch with an overlaid texture patch and an image of a dress. For both pairs, the participants were asked to identify the pairs that had a generated dress. The percentage of correct answers can be found in Table 6.3

Question	Participants who answered correctly(%)
1	22,5%
2	7,5%
3	12,5%

Table 6.4: Percentage of correct answers for Section IV multiple choice questions

For question 1, both pairs were generated; only 22,5% of the participants answered correctly. 47,5% only selected pair A while 20% only selected pair B. 10% considered that the images to be

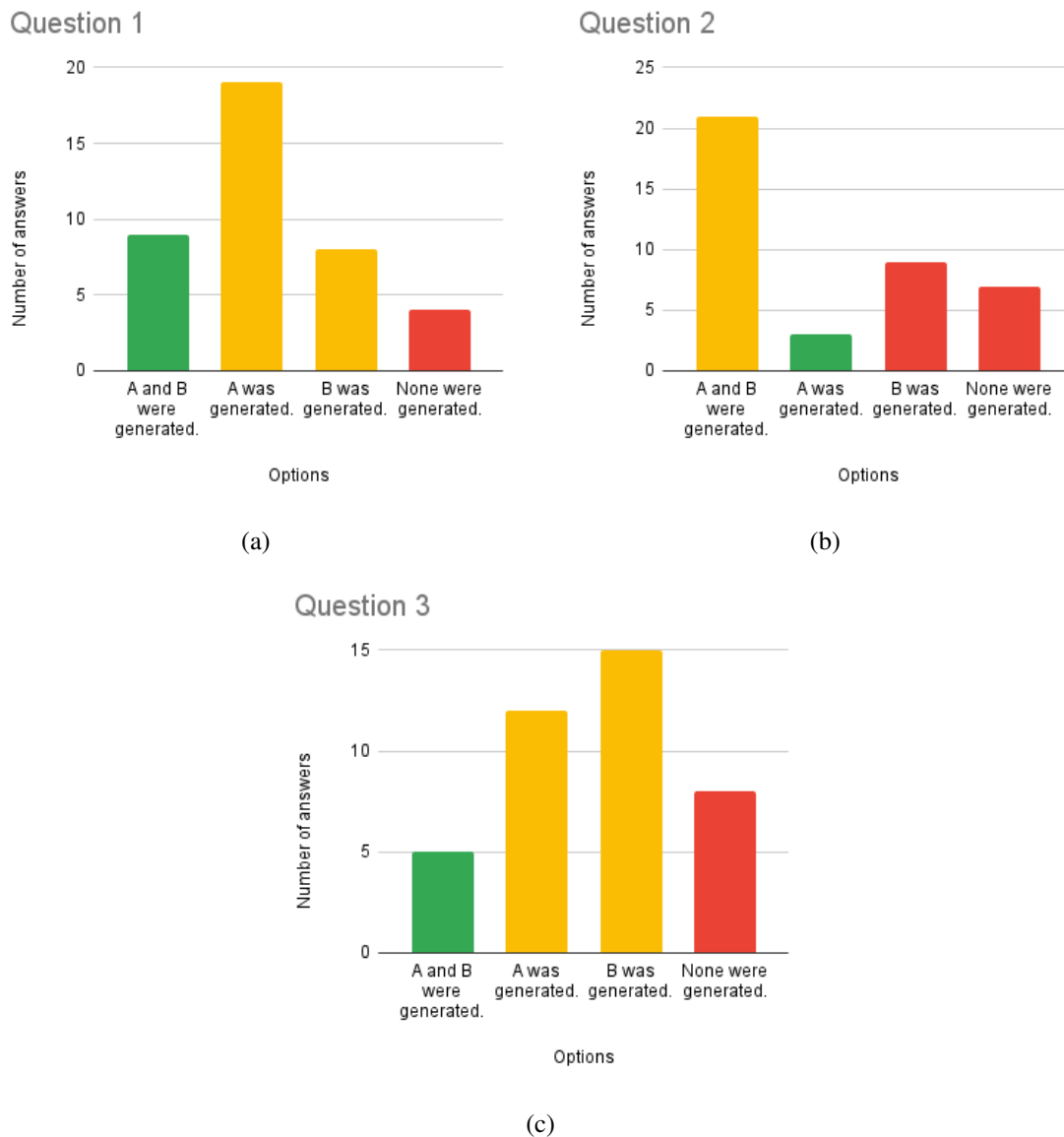


Figure 6.13: User survey answers for multiple-choice questions of Section IV, (a) Question 1, (b) Question 2, (c) Question 3

both real. In question 2, most of the participants answered incorrectly. In this question, the correct answer was pair A. More than half of the participants considered that the dresses from pairs A and B were generated. For question 3, only 12,5% answered correctly. The correct answer was that both pairs had generated dresses. 67,5% of the participants only selected one of the pairs, while 20% believed that none of the pairs contained generated dresses. On average, 14% of participants answered correctly. Figure 6.13 illustrates the number of participants who answered correctly to the multiple-choice questions. Correct answers are displayed in green, partially correct answers in yellow and wrong answers in red.

Some participants who selected that none of the images were generated while both images were generated justified that the dresses presented looked so real that they could not have been

generated by our AI model. Other participants had justified their decision by saying that they could not distinguish the generated images from the real images, and therefore their decisions were either random or based on their intuition.

The short answer questions of this section aimed to find what artifacts and characteristics participants have identified in the pairs of images that led their decision regarding the images chosen as the generated ones. In addition to the characteristics and artifacts of the generated images pointed out by participants in the previous section, other features of the generated images directly related to the fashion sketch provided were mentioned by participants in this section.

The most relevant feature pointed out in this section was the color inconsistency from fashion sketch to dress. In some generated dresses, 18% of participants noticed that the color of the fashion sketch texture patch did not match with the color of the generated dress. Hence, the Pattern Transfer model sometimes fails in applying the correct color constraint, which is consistent with the results analysis presented in Section 6.2.

6.3.5 Section V - Evaluating the marketability of the generated garments

The last section aimed to evaluate the marketability of the generated garments. Participants were presented with three generated garments and were asked to indicate on a scale of 1 to 5 whether they agree or disagree with the statement "Customers are likely to buy the generated dress.". Figure 6.14 illustrates the participants answers to Section V questions. On average, 63% of the participants agreed or strongly agreed with the formulated statement. 15% of the participants disagreed or strongly disagreed with the statement. That being said, the majority of the participants agreed with the formulated statement. Hence, customers are likely to buy the generated dresses.

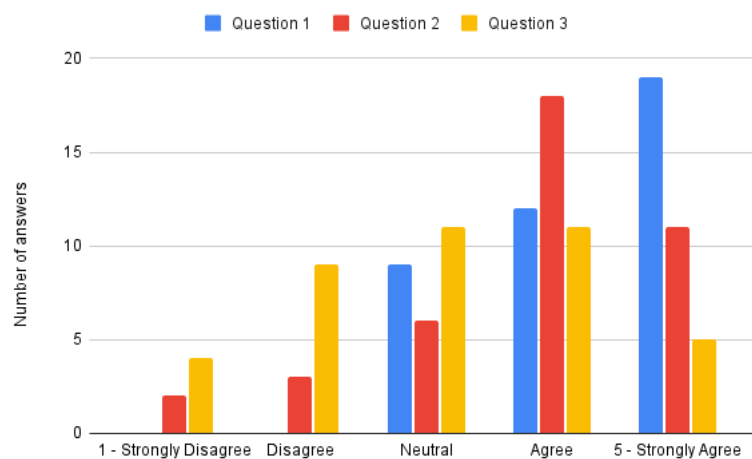


Figure 6.14: User survey answers distribution for questions of Section V

Chapter 7

Conclusion and Future Work

In the present chapter, the research findings and the research achievements of this research work will be carefully explained and summarized. In addition, the challenges faced during the implementation and execution of the models will also be presented. Furthermore, the limitations of the research work and future research directions will be outlined.

7.1 Research Findings

Two deep learning models that tackled two distinct computer vision tasks were proposed and implemented in this research work. The first model, the Conditional Garment Generation model, tackles the task of conditional image generation, while the second model, the Pattern Transfer model, tackled the task of paired image-to-image translation.

To summarise, the research question formulated for this research project is as follows:

Can AI models be used to assist designers during the design creation stage by generating complete clothing collections and turning fashion sketches into real images of garments?

The following research objectives were formulated to guide the research project:

- **Research Objective 1** - To research the available AI technology applied to fashion and to the tasks of image generation and image-to-image translation.
- **Research Objective 2** - To implement a conditional generative model capable of generating different types of garments based on certain conditions.
- **Research Objective 3** - To implement an image to image translation model that converts fashion sketches into real garment images.
- **Research Objective 4** - Evaluate the performance of the models based on user surveys and evaluation metrics selected.

Below, a description of how the research objectives formulated were met.

- To research the available AI technology applied to fashion and the tasks of image generation and image-to-image translation.

The research objective was fulfilled through the analysis of AI research works within the fashion field and the analysis of research works for image generation and image-to-image translation tasks as presented in Chapter 3. Models to solve problems such as garment attribute manipulation, garment generation, garment translation, garment extraction, and garment transfer were discussed in detail. Furthermore, deep learning models for image-to-image translation and image generation tasks were also discussed. Considering the identified research works, it was possible to conclude that the majority of them are based on ANNs and GANs. The most important concepts and components of Machine Learning were presented. Furthermore, a review over deep learning, ANNs, CNNs, and GANs was performed and presented in Chapter 2.

- To implement a conditional generative model capable of generating different types of garments based on certain conditions.

This research objective was met with the implementation of the conditional garment generation model. The conditional garment generation model is a generative model that generates garments that follows a specific criterion defined by the user. The criterion used was the type of garments to be generated. The model architecture is based on the StyleGAN2 [44] model architecture. Data augmentation techniques from StyleGAN2-ADA [27] were implemented to improve the model performance in smaller datasets. The components of the model can be found in Section 4.3.

- To implement an image to image translation model that converts fashion sketches into real garment images.

This research objective was accomplished with the implementation of the Pattern Transfer model. The Pattern Transfer model is able to turn fashion sketches with overlaid texture patches into real garments images. The model requires a paired dataset made up of garments and their fashion sketches. The Pattern Transfer model can be trained with 3 different generators (*e.g.*, Scribbler, Scribbler dilate 128, and TextureGAN). The loss functions and generators are based on TextureGAN [68] model, while the discriminator is based on the CGG model Discriminator. The models components can be found in Section 4.4.

- Evaluate the performance of both models based on user surveys and evaluation metrics selected.

This research objective was fulfilled by evaluating the proposed models in terms of performance metrics and a user survey. For the conditional garment generation model, the performance metric FID was collected and pointed out for a worse value than expected, given the perceived image quality. We concluded that the unbalanced dataset used was the cause of that. Using a bigger and well-balanced dataset could improve the metric values.

Regarding the Pattern Transfer model, the metric SSIM was collected throughout the model training, and a user survey was conducted to evaluate the generated images. The SSIM value

pointed to a high quality in the generated garments. The user survey aimed to assess the realness and marketability of the garments generated from fashion sketches. Most participants failed to distinguish the generated garments from the real garments. On average, participants only correctly identified the generated garment in 46% of the questions. In addition, most participants agree that customers are likely to buy the generated garments.

7.2 Research Contributions

The major contributions of the research work are described below:

- A literature review on applications of AI in the fashion area especially applied in the creative field and a background review on AI. The background review includes machine learning, deep learning, and also an explanation of the principal components of deep learning models, namely ANNs, CNNs, and GANs. In addition, research works on image-to-image translation and image generation were studied and outlined.
- A model capable of conditionally generating new garments based on specific conditions. The user can choose the type of garment to be generated by the model. The model has trained with only a category of labels, *i.e.*, the category type of garment. However, the implementation is prepared for multi-label datasets. In other words, it is prepared for datasets with more than one label per image. (*e.g.* A dataset of Dresses with two labels per dress, a label for the category sleeve length, and a label for the category type of neckline.)
- A garment labeled dataset with over 44 000 images of 19 different types of women's garments. The types of garments include upper body garments, whole-body garments, and lower body garments. All the images in the dataset feature a single-piece garment on a white background.
- A model capable of converting fashion sketches with an overlaid texture patch into real garment images. The model produces realistic dresses in line with the input fashion sketches and the textures.
- A paired fashion dataset with over 25 000 pairs of fashion images. Each pair is made up of an image of a dress and the correspondent fashion sketch. The fashion sketch was created using the DexiNed [62] model. Multiple retailers' websites were scrapped, each dress image contains a single dress in a white or grey background.
- An evaluation of the implemented models' performance with commonly used performance metrics (FID and SSIM) and user surveys. The FID metric used to evaluate the conditional garment generation model turned out to be worse than expected. Hence, it led to a deeper analysis of the training dataset's impact on the model's performance and results. The SSIM metrics showed that the images generated by the Pattern Transfer model are high quality

and very similar to the ground-truth images. The user survey assessed the realness and marketability of the Pattern Transfer model garments generated. On average, participants only correctly identified the generated garment in 46% of the questions provided. In addition, most participants agree that customers are likely to buy the generated garments.

7.3 Threats to Validity

In this research work, some limitations can be found. The conditional garment generation model was trained with women's garments. Thus, the model only generated women's garments which can only assist designers in creating women's clothing collections. Therefore, the model would not be suitable for designing menswear or children's clothing.

The Pattern Transfer model was trained with a dataset of dresses. Thus, it is uncertain if the model will produce good results with fashion sketches from other types of garments.

The target audience of the user surveys should be people within the fashion industry and fashion consumers. Unfortunately, it was impossible to collect answers from the target audience within the resources and time frame available. The user survey participants were students from the faculty of engineering of the University of Porto and Microsoft and XNFY Lab workers. Thus, the conclusions drawn from the user survey results cannot be generalized for the target audience.

7.4 Challenges

One of the biggest challenges of the present research work was the amount of computation power needed to run the implemented models. Powerful graphic cards and high amounts of video RAM are needed. Most personal computers cannot fulfill those requirements and conduct the experiments in a reasonable time frame. Therefore, the experiments were carried out in the cloud with Microsoft Azure Machine Learning virtual machines. In the virtual machines, the experiments would take between 5 to 20 minutes to start running.

Furthermore, the time spent debugging was huge, and the task was very frustrating. When a problem was encountered, it would take at least 15 minutes to see if the changes made in the code would fix the problem. Thus, a vast amount of time was wasted just in debugging. If the experiment could be carried out on personal computers in seconds, the model would start running, and debugging would be faster.

Another problem found, was the amount of time required for training each implemented model. For instance, an experiment using the Conditional garment generation model with 128x128 of resolution took 130 hours to be completed, which correspond to more than five days. Therefore, given the lack of time or resources, not all planned training runs were conducted. The Pattern Transfer experiments were not as heavy as the conditional garment generation experiments. Each experiment only took 6 hours to be completed.

The Pattern Transfer model is very unstable. Almost half of the time invested in this research work was spent changing the Pattern Transfer model architecture and components to achieve better

stability and results. Throughout the training, the generated images would achieve good results and gradually become darker and darker. The cause of the problem, despite the efforts, could not be found. Furthermore, many training runs were conducted, which took a vast amount of time and computational resources.

7.5 Future Research

The conditional garment generation model was implemented to help designers in the creation of clothing collections. However, the model is not 100% accurate and sometimes does not generate the correct type of garment. The cause of the model's incorrect behavior was the training dataset used. In the future, the model should be trained with a bigger and more balanced dataset to improve the model's results.

It would be interesting in the future to train the conditional garment generation model with a multi-label dataset. It would be beneficial for designers to generate garments with features according to their specifications (*e.i.*, Designers could have the possibility to generate dresses with a v-neckline and short sleeves if the model was trained with a dataset of dresses with annotations on each dress sleeve length and neckline type.)

The patterns transfer model only generated images in a resolution of 128x128. The ideal image resolution should be at least 512x512 so, in the future, the model can be improved in terms of both, stability and image resolution. Also, it would be interesting to implement a model capable of converting garments images into fashion sketches. The model would learn to ignore the texture of the garments and only extract the contours, folds, and seams. This model could then be used to create a better-paired dataset. Thus, it would improve the Pattern Transfer model performance.

7.6 Final Remarks

The fashion industry is one of the world's largest industries and one of the most volatile ones. Furthermore, the fashion industry is one of the most polluting in the world in an era where sustainability is becoming increasingly important. Integrated sustainability while maintaining a competitive advantage with low prices and a fast time to market is almost impossible. Sustainability usually implies that more costs are involved. Thus the final prices will be higher. Sustainability-conscious consumers probably would not mind an increase in the product's price, but the same would not happen with the majority of consumers that would choose to buy cheaper garments from competitors.

This research work presented two models that will reduce the costs and time spent in the design creation stage, allowing the industry to integrate sustainability while maintaining the competitive advantage and product prices. The first proposed model aims for the generation of different types of garments. The second proposed model is capable of turning fashion sketches into real images of garments. The first model was evaluated using the performance metric FID. Given the perceptible quality of the generated images, it would be expected for the FID value to be better. It was found

that a more balanced and bigger dataset should have been used to achieve the expected results. The second model was evaluated using the performance metric SSIM and a user survey. The metric pointed out to high-quality generated images almost identical to the ground truth images. The user survey results indicate that the generated garments appear real and that customers are most likely to buy them.

Appendix A

User Survey

The user survey conducted to evaluate the performance of the pattern transfer model proposed is presented below. The survey aim's to assess the community's visual perception of artificially generated garments and study their marketability.

The images shown on the user survey's questions that were generated by the proposed pattern transfer model are presented below.

- **Section I: Visual perception of generated garments**
 - All dresses presented are generated.
- **Section II: Identify the generated dress**
 - **Question 1:** Blue Dress
 - **Question 2:** Black Dress
 - **Question 3:** Black Dress
- **Section III: Fashion sketch to dress I**
 - **Question 1:** Green Dress
 - **Question 2:** Blue Dress
 - **Question 3:** Pink Dress
- **Section IV: Fashion sketch to dress II**
 - **Question 1:** A and B
 - **Question 3:** A
 - **Question 4:** A and B
- **Section V: Evaluating the marketability of the generated garments**
 - All dresses presented are generated.

Visual perception of generated garments

Question 1

On a scale of 1 to 5, please rate the degree of realism of the image presented.



1 2 3 4 5

Not at all realistic ☐ ☐ ☐ ☐ ☐ Very realistic

Question 2

On a scale of 1 to 5, please rate the degree of realism of the image presented.



1 2 3 4 5

Not at all realistic ☐ ☐ ☐ ☐ ☐ Very realistic

Question 3

On a scale of 1 to 5, please rate the degree of realism of the image presented.



1 2 3 4 5

Not at all realistic ☐ ☐ ☐ ☐ ☐ Very realistic

Identify the generated dress

Question 1

Among the following images identify which one is generated.



☐ Blue Dress



☐ Green Dress

Question 2

Among the following images identify which one is generated.



☐ Blue Dress



☐ Black Dress

Question 3

Among the following images identify which one is generated.



☐ Black Dress



☐ Pink Dress

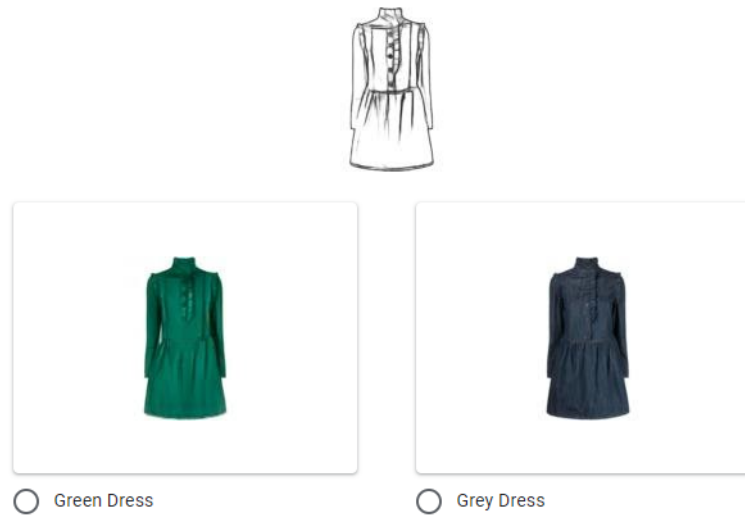
Question 4

On a scale of 1 to 5, please rate how difficult it was to identify the generated images.

Very easy 1 2 3 4 5 Very hard
☐ ☐ ☐ ☐ ☐

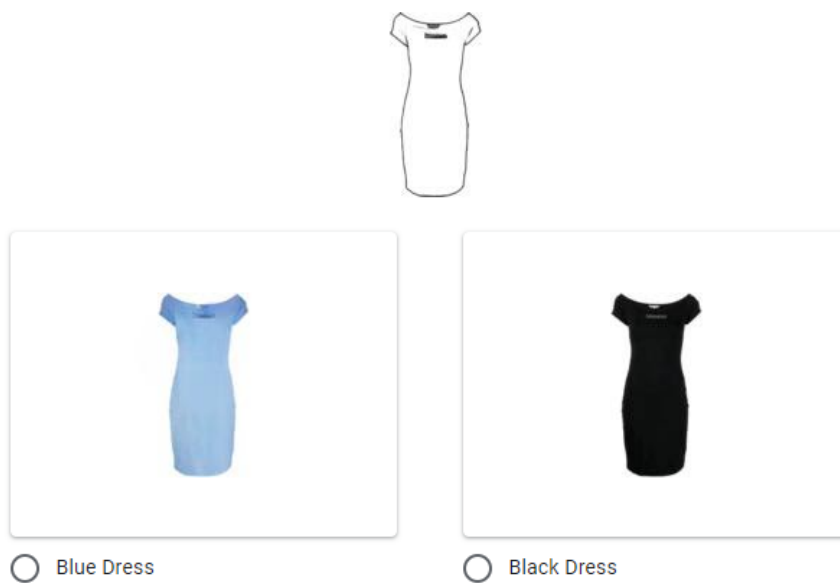
Fashion sketch to dress I**Question 1**

Identify which image was generated from the following sketch. Note: one of the images is generated and the other is a real image from which the sketch was extracted.



Question 2

Identify which image was generated from the following sketch. Note: one of the images is generated and the other is a real image from which the sketch was extracted.



Question 3

Identify which image was generated from the following sketch. Note: one of the images is generated and the other is a real image from which the sketch was extracted.



☐ White Dress



☐ Pink Dress

Question 4

Please justify your decisions.

Question 5

On a scale of 1 to 5, please rate how difficult it was to identify the generated images.

	1	2	3	4	5	
Very easy	☐	☐	☐	☐	☐	Very hard

Fashion Sketch to dress II

Question 1

Please identify which image was generated.



- ☐ A was generated.
- ☐ B was generated.
- ☐ A and B were generated.
- ☐ None were generated.

Question 2

Please justify your decision.

Question 3

Please identify which image was generated.

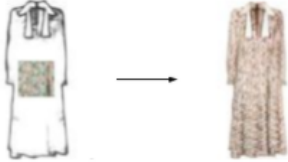


- ☐ A was generated.
- ☐ B was generated.
- ☐ A and B were generated.
- ☐ None were generated.

Question 4

Please identify which image was generated.

A



B



☐ A was generated.

☐ B was generated.

☐ A and B were generated.

☐ None were generated.

Question 5

Please justify your decision.

Evaluating the marketability of the generated garments

The following questions aim to assess the marketability of the generated garments. Please indicate whether you agree or disagree with the statements that are presented.

Question 1

Customers are likely to buy the generated dress.



1

2

3

4

5

Strongly Disagree

☐

☐

☐

☐

☐

Strongly Agree

References

- [1] Kenan Ak, Ashraf Kassim, Joo-Hwee Lim, and Jo Yew Tham. Attribute manipulation generative adversarial networks for fashion images. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10540–10549, 2019.
- [2] Kenan Ak, Hwee Lim, Jo Tham, and Ashraf Kassim. Semantically consistent text to fashion image synthesis with an enhanced attentional generative adversarial network. *Pattern Recognition Letters*, 135, Mar 2020.
- [3] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.
- [4] Hamed Alqahtani, Manolya Kavakli-Thorne, and Dr. Gulshan Kumar Ahuja. An analysis of evaluation metrics of gans. In *International Conference on Information Technology and Applications (ICITA)*, July 2019.
- [5] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.
- [6] David Berthelot, Thomas Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks, 2017.
- [7] Ali Borji. Pros and cons of gan evaluation measures, 2018.
- [8] Jason Brownlee. How to develop a conditional gan (cgan) from scratch, Sep 2020.
- [9] A. Burkov. *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019.
- [10] Yunje 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, 2018.
- [11] François Chollet. *Deep Learning with Python*. Manning, November 2017.
- [12] Choosing color deviation tolerances. shorturl.at/dnwKZ. Last accessed 1 September 2021.
- [13] Clustering k-means. <https://ichi.pro/pt/clustering-k-means-66942525093873>, Jan 2021. Last accessed 4 July 2021.
- [14] J. Galindo and P. Tamayo. Credit risk assessment using statistical and machine learning: Basic methodology and risk modeling applications. *Computational Economics*, 15(1-2):107–143, 2000. cited By 124.

- [15] Yuying Ge, Ruimao Zhang, Lingyun Wu, Xiaogang Wang, Xiaoou Tang, and Ping Luo. Deepfashion2: A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images. *CoRR*, abs/1901.07973, 2019.
- [16] Aurelien Geron. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, Sebastopol, CA, 2017.
- [17] Global fashion industry statistics. <https://fashionunited.com/global-fashion-industry-statistics/>. Last accessed 1 August 2021.
- [18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Y. Bengio. Generative adversarial networks. *Advances in Neural Information Processing Systems*, 3, June 2014.
- [19] Filip Isakov. Filip has a degree in English language, literature, Marija Kovachevska, Filip Isakov, and Name *. 15 fashion industry statistics for staying on trend in 2021, Apr 2021.
- [20] How computer vision applications are changing our world. <https://techsee.me/blog/computer-vision-applications/>, Mar 2021. Last accessed 2 July 2021.
- [21] How much do our wardrobes cost to the environment? <https://www.worldbank.org/en/news/feature/2019/09/23/costo-moda-medio-ambiente>, Apr 2021. Last accessed 1 August 2021.
- [22] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization, 2017.
- [23] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.
- [24] Tomáš Horváth João Mendes Moreira, André C. P. L. F. de Carvalho. *A General Introduction to Data Analytics*, chapter 9, pages 187–210. John Wiley Sons, Ltd, 2018.
- [25] Tomáš Horváth João Mendes Moreira, André C. P. L. F. de Carvalho. *A General Introduction to Data Analytics*, chapter 8, pages 159–185. John Wiley Sons, Ltd, 2018.
- [26] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2018.
- [27] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *CoRR*, abs/2006.06676, 2020.
- [28] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019.
- [29] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan, 2020.
- [30] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks, 2017.
- [31] J. Langr and V. Bok. *GANs in Action: Deep learning with Generative Adversarial Networks*. Manning Publications, 2019.

- [32] Kathleen M Lewis, Srivatsan Varadharajan, and Ira Kemelmacher-Shlizerman. Vogue: Try-on by stylegan interpolation optimization, 2021.
- [33] Jae Hyun Lim and Jong Chul Ye. Geometric gan, 2017.
- [34] Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. Pacgan: The power of two samples in generative adversarial networks, 2018.
- [35] Jinhuan Liu, Xuemeng Song, Zhumin Chen, and Jun Ma. Mgcmm: Multi-modal generative compatibility modeling for clothing matching. *Neurocomputing*, 414, June 2020.
- [36] L. Liu, H. Zhang, X. Xu, Z. Zhang, and S. Yan. Collocating clothes with generative adversarial networks cosupervised by categories and attributes: A multidiscriminator framework. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):3540–3554, 2020.
- [37] 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.
- [38] Sidharth Mehra and Mohammed Hasanuzzaman. Detection of offensive language in social media posts., May 2020.
- [39] Matiur 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.
- [40] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.
- [41] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.
- [42] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. MIT Press, Cambridge, Mass. [u.a.], 2013.
- [43] Michael A. Nielsen. Neural networks and deep learning, 2018.
- [44] Nvidia released state-of-the-art stylegan2 model. <https://neurohive.io/en/news/nvidia-released-state-of-the-art-stylegan2-model/>, June 2020. Last accessed 1 August 2021.
- [45] Xu Ouyang and Gady Agam. Accelerated wgan update strategy with loss change rate balancing, 2020.
- [46] Nilesh Pandey and Andreas Savakis. Poly-gan: Multi-conditioned gan for fashion synthesis, 2019.
- [47] Krut Patel. Convolution neural networks - a beginner's guide. <https://towardsdatascience.com/convolution-neural-networks-a-beginners-guide-implementing-a-mnist-hand-wri> Oct 2020. Last accessed 1 August 2021.
- [48] Prabhu. Understanding of convolutional neural network (cnn) - deep learning, Nov 2019.
- [49] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

- [50] Amir Hossein Raffee and Michael Sollami. Garmentgan: Photo-realistic adversarial fashion transfer. *CoRR*, abs/2003.01894, 2020.
- [51] Anand Srinivasa Rao and G. Verweij. Sizing the prize: what’s the real value of ai for your business and how can you capitalise?, 2017.
- [52] Negar Rostamzadeh, Seyedarian Hosseini, Thomas Boquet, Wojciech Stokowiec, Ying Zhang, Christian Jauvin, and Chris Pal. Fashion-gen: The generative fashion dataset and challenge, 2018.
- [53] Cyprien Ruffino, Romain Hérault, Eric Laloy, and Gilles Gasso. Pixel-wise conditioned generative adversarial networks for image synthesis and completion, 2020.
- [54] D. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.
- [55] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.
- [56] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color, 2016.
- [57] Yangyun Shen, Runnan Huang, and Wenkai Huang. Gd-stargan: Multi-domain image-to-image translation in garment design. *PLOS ONE*, 15(4):1–15, Abr 2020.
- [58] Pamela Simmons. When do fashion seasons start? <https://www.leaf.tv/articles/when-do-fashion-seasons-start/>, May 2021. Last accessed 1 August 2021.
- [59] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.
- [60] Montek Singh, Utkarsh Bajpai, Vijayarajan V., and Surya Prasath. Generation of fashionable clothes using generative adversarial networks: A preliminary feasibility study. *International Journal of Clothing Science and Technology*, 32:177–187, Abr 2020.
- [61] Aaron Smith and Janna Anderson. Ai, robotics, and the future of jobs, Aug 2014.
- [62] Xavier Soria, Edgar Riba, and Angel D. Sappa. Dense extreme inception network: Towards a robust cnn model for edge detection, 2020.
- [63] Audrey Stanton. What is fast fashion, anyway? <https://www.thegoodtrade.com/features/what-is-fast-fashion>, Mar 2021. Last accessed 1 August 2021.
- [64] The stylegan code released: Neural network for faces generation by nvidia. <https://neurohive.io/en/state-of-the-art/stylegan-code-released/>, Mar 2019. Last accessed 1 August 2021.
- [65] Bochao Wang, Huabin Zheng, Xiaodan Liang, Yimin Chen, Liang Lin, and Meng Yang. Toward characteristic-preserving image-based virtual try-on network, 2018.
- [66] Zhou Wang, Alan Bovik, Hamid Sheikh, and Eero Simoncelli. Image quality assessment: From error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13:600 – 612, 05 2004.

- [67] What is fast fashion? <https://goodonyou.eco/what-is-fast-fashion/>, Jul 2021. Last accessed 1 August 2021.
- [68] Wenqi Xian, Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Texturegan: Controlling deep image synthesis with texture patches. *CoRR*, abs/1706.02823, 2017.
- [69] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.
- [70] Cui Yirui, Q. Liu, C. Gao, and Zhuo Su. Fashiongan: Display your fashion design using conditional generative adversarial nets. *Computer Graphics Forum*, 37:109–119, Oct 2018.
- [71] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into deep learning, 2021.
- [72] Haijun Zhang, Yanfang Sun, Linlin Liu, Xinghao Wang, Liuwu Li, and Wenyin Liu. Clothimgout: a category-supervised gan model for clothing segmentation and retrieval. *Neural Computing and Applications*, 32:4519–4530, May 2020.
- [73] Haijun Zhang, Yanfang Sun, Linlin Liu, and Xiaofei Xu. Cascadegan: A category-supervised cascading generative adversarial network for clothes translation from the human body to tiled images. *Neurocomputing*, 382, Dec 2019.
- [74] Lei Zhao, Sihuan Lin, Ailin Li, Huaizhong Lin, Wei Xing, and Dongming Lu. *Spatial-GAN: Progressive Image Generation Based on Spatial Recursive Adversarial Expansion*, page 2336–2344. Association for Computing Machinery, New York, NY, USA, 2020.
- [75] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2020.
- [76] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A. Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation, 2018.