AUTOMATIC GENERATION OF GARMENT DESIGNS USING GENERATIVE ADVERSARIAL NETWORKS

Abesinghe Mudiyanselage Hashan Darshana Karunathilaka

199335K

Degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa
Sri Lanka

June 2022

AUTOMATIC GENERATION OF GARMENT DESIGNS USING GENERATIVE ADVERSARIAL NETWORKS

Abesinghe Mudiyanselage Hashan Darshana Karunathilaka

199335K

Degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa
Sri Lanka

June 2022

Declaration

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other University or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature : . UOM Verified Signature Date: …24-10-2022…..

Name: A.M.H.D. Karunathilaka

The above candidate has carried out research for the Masters thesis under my supervision. I confirm that the declaration made above by the student is true and correct.

Signature of the supervisor: UOM Verified Signature Date: ……24-10-2022……

Name: Dr. Thanuja D. Ambegoda

Acknowledgement

My sincere appreciation goes to everyone who supported to make this thesis a success. I also thank my supervisor Dr. Thanuja Ambegoda, for the advice and the guidance to make this research a success. I also thank Dr. Sapumal Ahangama for introducing me to my supervisor.

Abstract

Generative models like GANs are able to generate realistic samples [1] https://thispersondoesnotexist.com. GANs have been used in the fashion domain as well. Manipulating attributes of a given garment [2], filling a garment sketch using a given fabric [3], generating new clothing on an image of a wearer through text description [4] are a few of such usages. However modeling a distribution of the dresses and manipulating the attributes of the generated dresses are not up to date with the advancement of the GANs. Firstly, the current state of the art GAN models and their applicability for the dress images is analyzed. Then the methods of manipulating the attributes of generated dresses by interpreting the latent space are explored. Finally, the application of the GANs for dress images and a way to interpret the latent code to manipulate the dress attributes successfully are presented with results.

Keywords: garment design generation, GAN

Table of Contents

List of Figures

List of Tables

1. INTRODUCTION

1.1 Background

Garment designing is a work of creative individuals. With the competition and short time to market in the fashion industry, coming up with innovative designs in a strict timeline can be challenging. A startup company might not be able to afford a good designer. Do-it-yourself people might not be able to come up with nice designs to make themselves some nice clothes. A model that could learn to innovate new designs will benefit in all such cases by breaking the barrier of innovation and creativeness. The ability to generate a garment design of a targeted type of garment that has given attributes could be even more beneficial.

1.2 Generative Adversarial Networks

Generative Adversarial Network is a generative model. GANs were inspired by game theory [5] and include a generator and a discriminator. In the training phase, the generator tries to generate fake samples and trick the discriminator to think that they are from real data. The discriminator tries to classify the samples from both fakes and reals. In the end, the generator has learned the distribution of real data. GANs have shown their effectiveness in image synthesis and editing. They have been extended (CGANs) to be able to generate samples based on provided conditions and therefore can be considered as a strong candidate for the objective of this research.

[1] thispersondoesnotexist.com, thiscatdoesnotexist.com shows realistic generated images and is the inspiration of this research. But these implementations have focused on faces and their attributes. Therefore, it might not be straightforward to reuse its architecture. Additionally, attribute-based conditioning must be included in the solution.

GANs are not straight forward to evaluate. The difficulty lies in measuring the difference of the real image dataset distribution and the synthesized image distribution [6].

1.3 The Problem

To generate novel fashion items, one must be creative. Generating unique creative ideas in a rush due to a time constraint must be exhausting. In the fashion industry, there is intense competition and a rush to produce attractive items to the market. A machine being able to be creative and produce novel ideas is a good thing to have in these conditions. Apart from that, people who could not afford to produce ideas themselves due to lack of creativity or the inability to hire a designer can benefit from such a thing.

Problem definition:

Training a model that can generate new fashion designs for given conditions, if possible being creative in doing so rather than generating the learned design distribution.

1.4 Motivation

The main motivation factor behind this research is exploring GANs for their ability to produce new garment designs and be creative.

1.5 Objectives

- Train a GAN that can generate new dress images.
- Manipulate the attributes of the generated images like sleeve length, etc.

2. LITERATURE REVIEW

2.1 Generative Models

Generative models are unsupervised learning models. Generative models learn to represent real distributions by learning a set of parameters that should fit with the real distribution estimated by the training data. However, the early generative models like Restricted Boltzmann Machines had limitations such as poor generalization.

2.2 Deep Generative Models

Deep generative models were able generalize well and generate better results in generative tasks. There are 3 main categories of deep generative models [7] ,

- Generative Adversarial Networks
- Variational Autoencoders
- Auto Regressive Networks

2.3 Variational Autoencoders

Autoencoders have a simple architecture. In between the encoder and the decoder, Autoencoders have a bottleneck layer which limits the amount of information that can go through the network. In the learning process of AEs, this bottleneck layer helps to learn the weights in such a way that the latent vector will be able to represent the distribution of the real data. In other words, the decoder will have to use the latent vector which is much smaller than the original input to regenerate the original input. As a byproduct of reducing the reconstruction loss of encoding and decoding, the encoder learns to compress / reduce the dimensionality of the input to a smaller latent vector. Due to the reduction of the representative space at the bottleneck, sharpness of the output can be reduced than the original.

Variational Autoencoders on the other hand, can be considered as an extension of the Autoencoders. Rather than just learning to produce a latent representation of the data and learning to recreate the original using the latent representation, VAEs learn the probability distribution of each latent attribute. Then generate samples from the latent space distributions and feed them into the decoder to generate a new output. However, VAE is not able to generate sharp images [8]

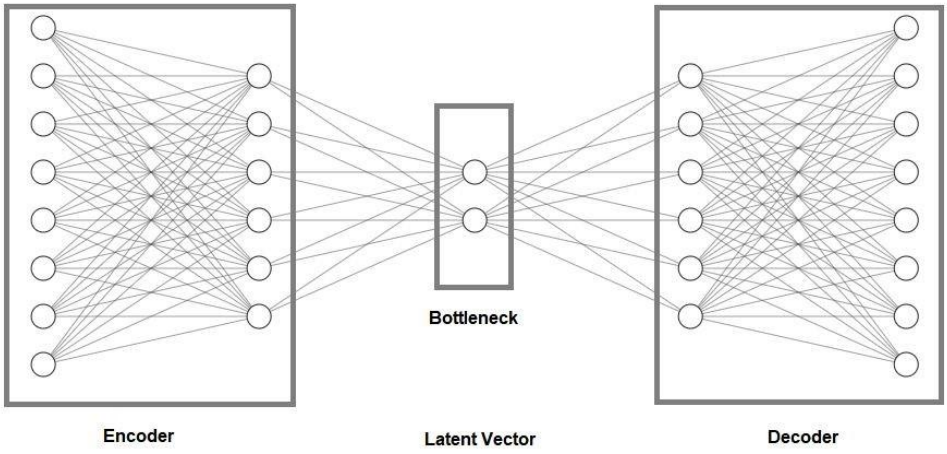

Figure 2.1 Autoencoder network

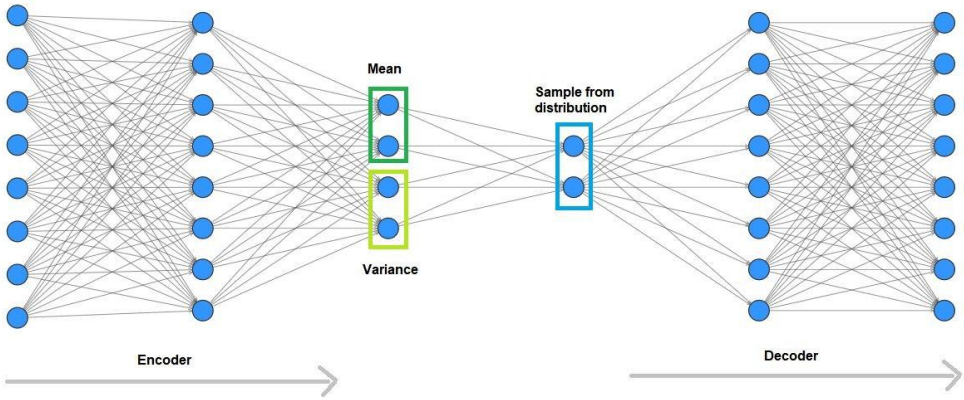

Figure 2.2 Variational autoencoders

2.4 Generative Adversarial Networks

Inspired by game theory, in GANs the generator and the discriminator compete [5]. Out of this competition, the generator learns the potential distribution of real data and gets the ability to generate realistic samples. Generator's input is a random noise vector z. Based on the random input z generator produces an image. The discriminator which is a binary classifier tries to learn to discriminate between real or fake. Discriminator gets to look at real data as well as fake. It learns through it's mistakes via back propagation. Generator learns in the same way. It learns through the feedback from the discriminator and tries to produce better fake samples.

GANs were proved to generate diverse results and sharper images compared to other models like Variational AutoEncoders [8]. Therefore the GANs have been popular in the research field lately. GANs were introduced by Goodfellow et al [5] in 2014.

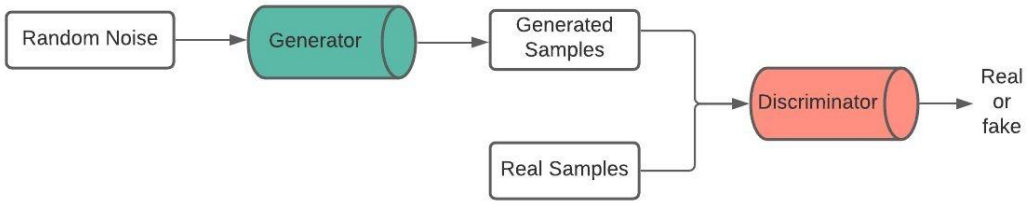

Figure 2.3 Generative Adversarial Network

2.4.1. Convolution GANs

For image based generative tasks, convolution has been used in the structure due its effectiveness in extracting image features [7]. For the generator of a Convolution GAN, transposed convolution layers are used to generate an image from a noise vector. For the discriminator, convolution layers are used.

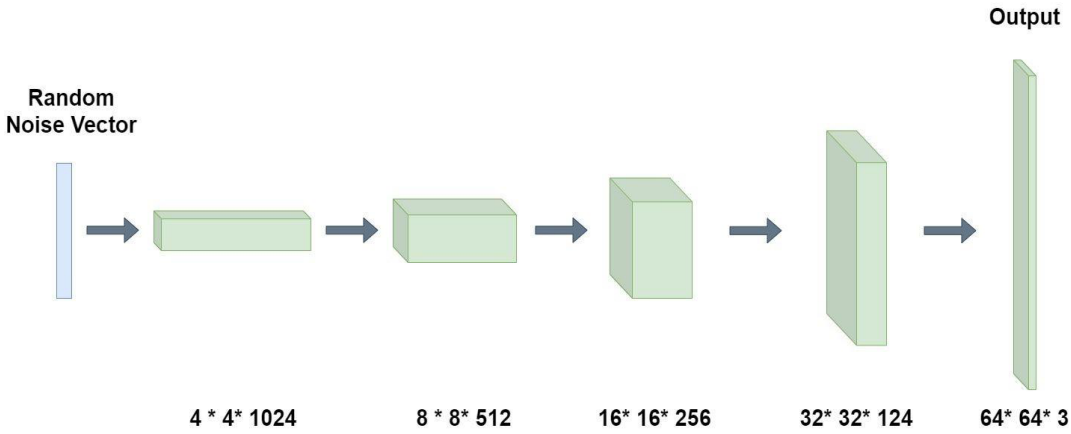

Figure 2.4 Generator architecture of a Deep Convolution GAN

2.4.2. Conditional GANs

As an extension of GANs, CGANs are capable of generating samples that have certain conditions or attributes instead of being a random attribute based on the random noise vector z. In CGANs, additional condition c is provided along with noise vector z as inputs to the generator. As shown in the figure, generator G, discriminator D and real data X all have a condition.

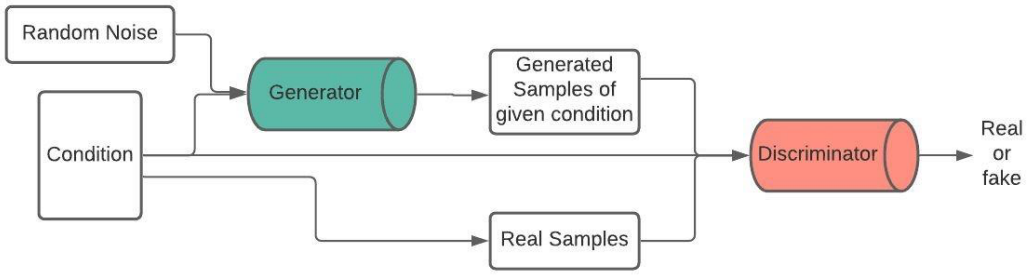

Figure 2.5 Conditional GAN Architecture

2.4.3. Challenges of GANs

The main 3 challenges in GANs are image quality, vanishing gradient and mode diversity [8]. Over the years researchers have tried new methods to overcome these challenges.

- Image Quality :

Generating realistic high quality images is a requirement of a GAN. The initial GAN [5] was able to generate low resolution images like those in MNIST, CIFAR-10 and Toronto face datasets. Later, architectures like Deep Convolution GANs (DCGAN [9]) were able to generate higher resolution images by using convolution to increase the architecture capacity. Modifications to the objective function and its ability to increase the image quality was proposed by the likes of WGAN [10] which introduced Wasserstein distance as a loss function. Likes of PROGAN [11] showed improved results by using modifications to training methods. PROGAN has a progressively growing training method. Progressive growing means that in the beginning it trains a smaller resolution like 8*8 with scaled down input images and with time adds the layers and higher resolution inputs progressively. This method showed the ability to learn higher resolutions better than the previous GAN architectures.

- Vanishing Gradient Problem :

In neural networks, the model is typically trained by gradient based learning methods and backpropagation. In training, the weights of the network will be updated "proportional to the partial derivative of the error function with respect to the weight" [12]. When the gradient becomes so small so that the update to the wights will be minimum or even zero causing the network to stop training.

In GANs, if the discriminator is becoming very good at identifying the fake and real samples, the feedback it can give to the generator approaches 0 [13]. It slows down or can completely stop learning of the Generator. This is the vanishing gradient problem of GANs.

Few remedies can be found for this problem. One is to use activation functions like ReLU, LeakyReLU instead of sigmoid and tanh which squashes the values between a smaller range like (0,1) and makes the gradient decrease in a large portion. Another way is to use a loss like Wasserstein loss [10] or modified minimax loss [5] which were proposed to deal with the vanishing gradient problem of GANs.

- Mode collapse:

When the generator produces only a few different samples rather than a diverse set of samples, this problem is called the mode collapse. This happens when the “discriminator gets stuck in a local minimum” and doesn't learn a way to classify a sample generated by the generator as fake. In the next iterations, the generator will keep generating the same sample that was able to fool the discriminator. “As a result, the generators rotate through a small set of output types. This form of GAN failure is called mode collapse” [14].

Wasserstein loss can be used as a remedy for mode collapse. In Wasserstein loss, the cost function continues to grow without a ceiling between 0 and 1, therefore the discriminator won't be stuck in the local minimum. Now the generator will have to learn to generate new results rather than keep generating the same set of samples that were able to fool the discriminator.

2.4.4. Evaluation Metrics

Evaluation metrics are a way to measure the difference between reals and fakes generated from the GANs. It acts as a way of quantifying how well the generator is doing its task after the training is completed. An ideal metric should determine the quality and the variety of the synthesized images. Some common evaluation metrics found in literature are below. But the 2nd one is most common in the literature.

1. Inception Score
2. Fréchet Inception Distance (FID)

Inception Score

This metric was first introduced in [11]. As mentioned in [15] “Inception Score was shown to correlate well with human scoring of the realism of generated images . For the calculation of the score it uses the pre-trained “Inception V3 Network”, which was designed to classify the images in ImageNet. The name of the Inception Score resembles it as well.

The Inception Network outputs a vector of probabilities for an image to belong to a particular class. Using these scores of a set of images, the final score is calculated by the “KL-divergence between the conditional class distribution and the marginal class distribution" as per [15]. The first distribution is supposed to give an indication of the quality of an image generated. The second distribution is supposed to give an indication of how diversified the generated images are.

$$\mathrm{IS}(G) = \exp\left(\mathbb{E}_{\mathbf{x}\sim p_g} D_{KL}(p(y|\mathbf{x}) \parallel p(y)) \right).$$

p(y|x) : conditional class distribution , p(y) : marginal class distribution

Fréchet Inception Distance (FID)

In this metric, an intermediate layer of the Inception Network is used to get the features of an image. Then these features are modeled as a multivariate Gaussian distribution. 2 distributions for real data and generated data will be modeled. Then for the final calculation, the distance between those two distributions will be calculated as below. Closer the distributions better the generation.

$$\mathrm{FID}(x, g) = ||\mu_x - \mu_g||_2^2 + \mathrm{Tr}(\Sigma_x + \Sigma_g - 2(\Sigma_x\Sigma_g)^{\frac{1}{2}}),$$

x: real, g: generated

2.4.5 Advanced GANs

Since the early GANs like FCGAN and DCGAN, there have been improvements in the training methods, architecture and objective functions to improve the stability, quality and variation. Models like Progressive GAN (PROGAN), Self-Attention GAN (SAGAN), StyleGAN, BigGAN have generated better results in higher resolution.

2.4.5.1. Progressive GAN (PROGAN)

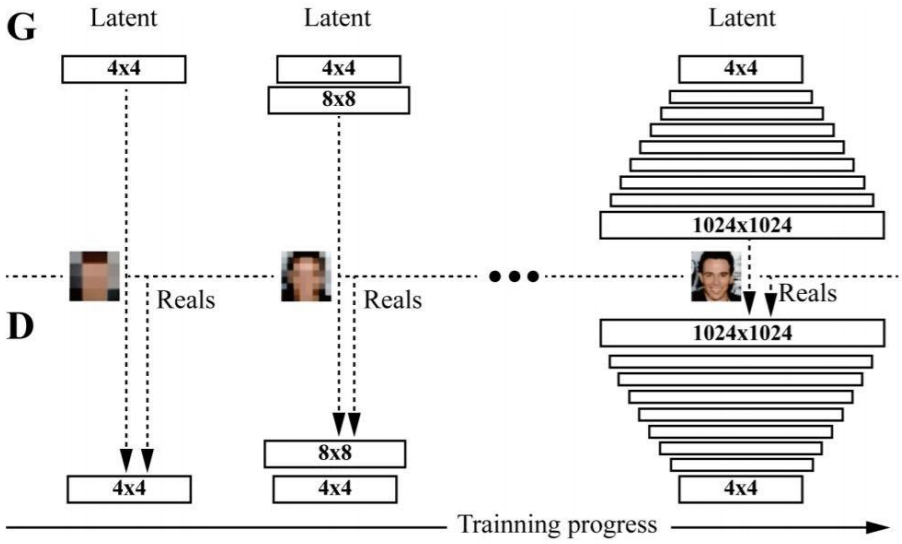

Figure 2.6 Progressive GAN training process figure from [2]

Progressive GAN [11] is an improvement of GANs to give them the ability to generate larger images. In PROGAN, the training starts with a small image resolution. As the training progresses, layers with higher resolution are incrementally added to both the discriminator and the generator. PROGANs have shown that they are more stable to train and can generate realistic images than the previous models according to [6]. The idea is to learn the high level features of the image first and incrementally learn the finer details by adding higher resolution layers to the network.

2.4.5.2. StyleGAN

StyleGAN is an improvement of ProGAN. The authors presented additional improvements in the generator network. The ProGAN's insight was that lower resolution layers at the initial layers learn the high level features such as layout, pose and the later high resolution layers learn the finer features. Based on that knowledge, the authors propose an alternative generator to traditional generator networks that are capable of automatically learning the separation of high to low level attributes of generated images. Additionally, the StyleGAN has improved the disentanglement of latent vectors. The distribution quality measuring FID metric was improved compared to the previous best ProGAN model.The paper focuses on human face synthesis problems to showcase their architecture.

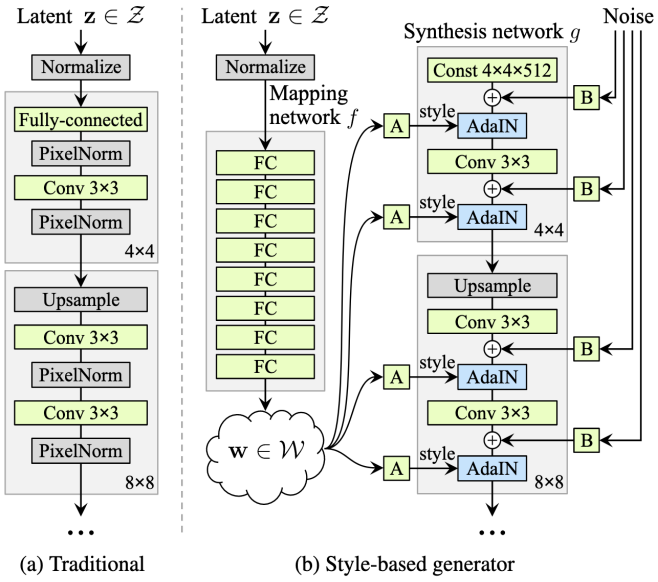

Figure 2.7 StyleGAN Generator Architecture from [1]

Different from the traditional generator inputs which were passed through the first layer of the generator, StyleGAN passes the latent vector to all layers of the generator through a novel fully connected mapping network. W space is the output of the mapping network for the Z space inputs. As mentioned above, these w values are passed through an learned affine transformation to AdaIN along with the previous layer features.

Mapping network

Mapping network reduces the entanglement of the latent vector by having an intermediate W space generated through the mapping layer. The authors argue that the latent space Z has to match the corresponding density in the training data. It causes factors to be entangled. When there is a mapped intermediate space W, it does not have to match the density of the training data. Hence, W is less entangled.

Stochastic Variation

Minor details of the images such as exact placement of hair, eyebrows can be varied keeping the overall perception of the image unchanged. This can be considered as the stochastic. It helps to achieve more variety in the generated images. StyleGAN allows to add stochasticity to the images by passing a noise to each pixel of the convolution layer output independently. By doing so, they allow only stochastic variation to be changed by the noise, not the global features like pose and background. They further added that since AdaIN takes statistics of a complete feature map of image, as an input of AdaIN, the style (derived from W) affects global features such as pose, lighting and background of the image.

Style Mixing and Coarse, Middle, Fine Styles

During training, two random latent codes (z) are used at a time and their relevant styles(two corresponding w codes) are passed randomly to each AdaIN operation rather than passing a single w code. The intuition behind is to avoid the network assuming that two close by styles are correlated. So that independent styles can be passed after training to control the style of different layers. However, there is nothing preventing passing the same style to all the layers after training to synthesize new images. This ability to get independent styles for each layer in the generator network was proved to be useful to control different type of features

- Coarse styles: *Pose, general hair style, face shape, eye glasses.*

By changing the style(w) of the first few layers (4*4 to 8*8) found to be controlling the coarse features / spatial features. such as pose, general hair style, face shape, eye glasses.

- Middle styles: *Small scale facial features, hair style, eyes open / closed*
  Middle layers (16*16 to 32*32)
- Fine styles: *Color scheme, micro structure*
  Last layers (64*64 to 1024*1024)

Changing one level of styles had minimum impact on the other levels of styles.

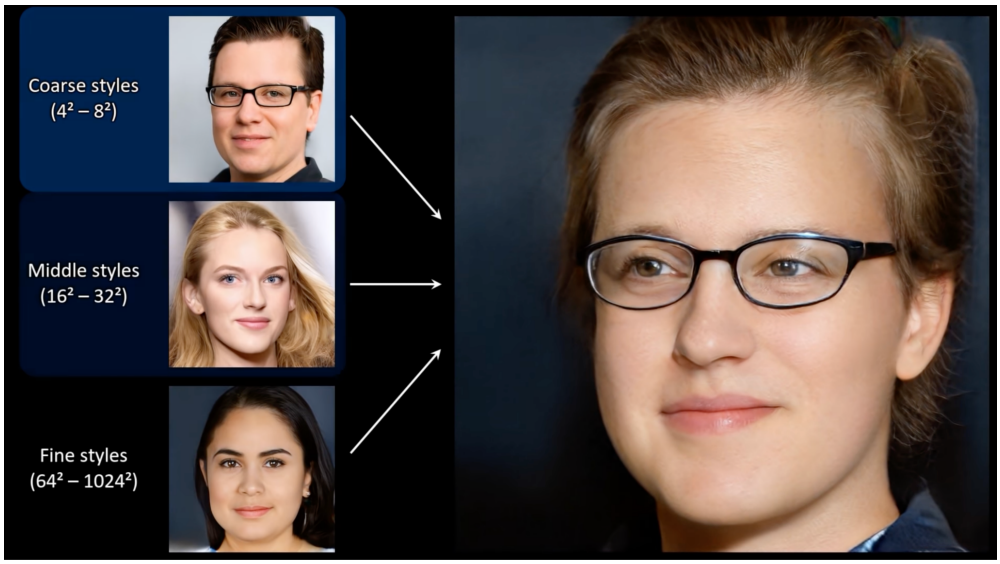

Figure 2.8 Mixing coarse, middle and fine styles in StyleGAN from [1]

2.4.6. Latent Space Exploration to Generate Desired Features

Latent Space

The generators get a latent code of n number of values (features) as its input. The synthesized image features decide the synthesized image features. Since the different values the latent code can represent, the input a generator can get will be in a n-dimensional latent space. Each input the generator gets is one data point of the latent space.

However, there might not be a one to one mapping of latent features to image features. This is called the feature entanglement problem. Due to this, manipulating one latent feature hoping to manipulate a certain feature can cause multiple features of the image to be changed. The mapping network and intermediate W space in the StyleGAN paper is an attempt to make the latent space less entangled.

Interpreting The Latent Space

In InterfaceGAN [16], the authors achieved significant improvements in interpreting the latent space and manipulating the features. They came up with a method to learn the latent boundaries for image features in the latent space by reusing the semantic knowledge within the trainend GANs. Additionally, they learned in their work that when a generative model is trained well, the latent code is disentangled if transformed linearly.

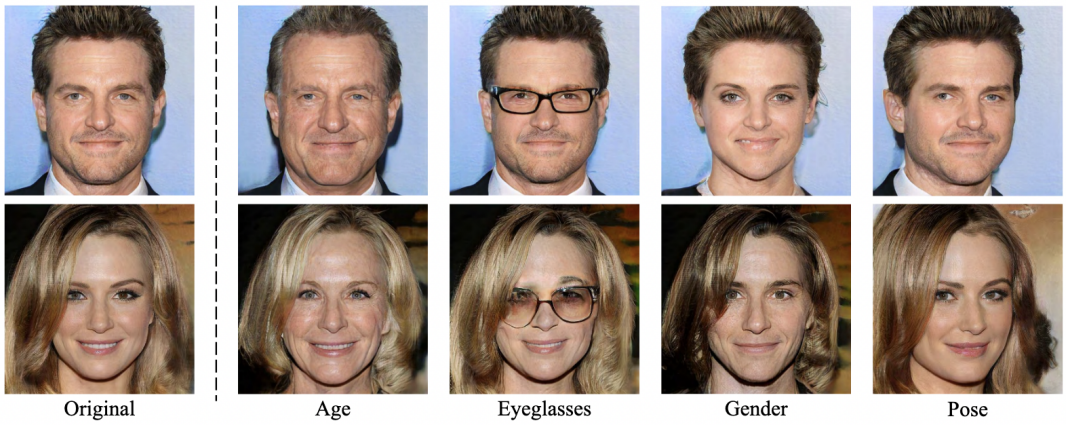

Figure 2.9 Attribute manipulation results from the InterfaceGAN [16]

They build on the previous findings of when linearly interpolating between two latent codes z1 and z2, the appearance of the generated image changes continuously. It suggests that in the transition, semantics also change gradually.

- Therefore they mention that there is a hyperplane in the latent space that acts as a boundary for binary features such as young/old (age).
- When some features are entangled, they propose a subspace projection method to disentangle those features.
- Real image manipulation is possible using GAN inversion then following their method of feature manipulation.

2.4.7. GAN Applications

Many GAN related applications were carried out for images. Major types of these applications are image super resolution, image-to-image-translation, image repair and image synthesis.

Fashion Generation

GANs have been applied for various tasks in the fashion domain. Filling sketches of garments with a fabric [3], changing attributes of a given garment image [2], change the attributes of wearer's image based on text description [4], generate texture on sketches using a texture sample [3], swapping fashion articles on people images [4] and generate personalized fashion items for a given fashion item [17] are some of the related work. AttGAN [18] produced good results in changing only required attributes in face images. In [19], the usability of AttGAN in changing attributes of garments and some alterations to the AttGAN to work with garment images were suggested. The resolutions of the images are small. Conditions seem to be limited. [19] can be understood more as a quick visualization of how modifications on an existing garment would look like rather than the purpose of this research which looks to generate novel fashion items.

FashionGAN [3], attempted 3 things. 1. “Fashion sketch to garment image” 2. “Garment image to another garment image” 3. “Contour image to garment image”. They have achieved good results. It is an image to image translation work and not the scope of this research. Fashion-AttGAN [20] also has similar work with more (22) attributes.

Figure 2.10 Output samples of paper [19]

Figure 2.11 Results of [3] FashionGAN

In Be your own Prada [4], the attempt was to generate new cloth on a wearer’s images based on textual description using GANs. It uses two GANs. 1st one to generate a segmentation map of the image of the wearer. Second GAN to render a new image according to the text instructions. Even though this has new garment generation, it has low resolution images and hard to see the design. The focus in [4] is to generate designs according to text instructions to see how different styles might look on a person, rather than generate new fashion ideas and focusing on the creative element.

Figure 2.12 Results of Be your own prada [4]

According to [4], in most fashion synthesis papers, the fashion items were kept the same. Only their appearances or modalities were changed after the translation. However, in personalized fashion design [17], the objective is to generate new fashion items according to users' previous preferences and the generated item should match a query fashion item. From all the previous work talked about, [4] had the most similar objective to this research. But [17] focuses more on matching one item with a new item (ex: a top wear to a bottom wear) and does not have the ability to generate required types of items with given attributes. Apart from that, in this paper, it is expected to explore the possibility of teaching a GAN to be more creative in generating fashion items rather than producing only from the learned distribution as in [17].

Figure 2.13 Personalized fashion design [17]. Designed items for different users given the same query

Table 2.1 Summary of previous work

References		Summary	Type	Limitation
Fashiongan: Display your fashion design using conditional generative adversarial nets - 2018	Computer Graphics Forum	Display how a contour/existing garment looks like in a given fabric.	Display fabric on garment	Results are good only for single colour or stripes like fabric. Only able to edit the given image
Attribute Manipulation Generative Adversarial Networks for Fashion Images - 2019	IEEE/CVF Internati- onal Conference on Computer Vision	Attribute manipulations on specific Regions of garment images	Attribute manipulation	Only able to edit the given image
Be Your Own Prada: Fashion Synthesis with Structural Coherence - 2017	IEEE international conference on computer vision	Change clothes of a person while keeping the person's pose unchanged based on a text description	Text description based attribute manipulation	Designed for images with plain background only. Only able to edit the given image

Fashion-AttGAN: Attribute-Aware Fashion Editing with Multi-Objective GAN	IEEE/CVF Conference on Computer Vision and Pattern Recognition	Improved editing of garment attributes compared to AttGAN [18]	Attribute manipulation	Only able to edit the given image
Garment Design with Generative Adversarial Networks	arXiv preprint	Attribute editing of garment images by changing the loss function	Attribute manipulation	Only able to edit the given image

3. METHODOLOGY

As per the objective of the research, the main task is to generate new garment images. Another auxiliary task is to manipulate the attributes of generated images. The dataset, GAN architecture and attribute manipulation technique used for this task will be described in this section .

3.1 Dataset

Two datasets were selected for this research. They both have annotations of their fashion images.

VITON dataset

Figure 3.1. Images in VITON

The dataset has 14221 fashion images with its class in 256 * 192 pixel resolution. The images only have the garment in it with white background which would be easier for the GAN to understand. However, few images have an angled view rather than the majority, which is the front view.

The image set has precompiled attribute annotations with it. The csv file contained image filename against 268 attributes. Some of the important attributes with their categories are listed down below. The number of images that has the attribute qualities are indicated within brackets. Out of 268 attributes, there were some attributes had no representation (count is 0) as well.

Type of garment

```
Dress (1147)
Women's vest (1720)
Women's vest 1 (61)
Women's Knitwear (239)
Women's sweater (21)
Women's POLO shirt (688)
Women's sweater.1 (105)
Women's shirt (122)
Women's T-shirt (10091)
Women's down (2)
Women's trench coat/long coat
(2)
Women's suits (1)
Women's short coat (15)
Swimsuit swim trunks (5)
```

Collar/Neck type

```
Stand-up collar (148)
Fur collar (2)
Suit collar (1)
Doll collar (46)
Shirt collar/shirt collar (82)
POLO/ T-shirt collar(716)
Hooded collar (37)
Bib collar/ pile collar (42)
Department leader (18)
Square collar (10)
U-neck (64)
One word collar (25)
Hanging lead (449)
Other collared (2)
Round neck (10784)
```

Homewear/pajamas (2)

V-neck (926)
Small V-neck (23)

Texture

Plain (9190)
Color matching (313)
Gradient (4)
Mixed colors (87)
Pinstripe (972)
Thick stripes (63)
Vertical stripes (42)
small square (37)
Large square (4)
Diamond lattice (7)
Houndstooth (3)
Wave point (213)
Personalized printing (340)
pattern (2597)
printing (283)
Geometric print (12)
Leopard (18)
Camouflage (17)
Jacquard (2)
Secondary suit pattern (13829)
Torso plain (23)
Local plain (284)
Color matching. 1 (26)
Fine horizontal stripes. 1 (4)
Vertical stripes. 1 (2)
Personalized printing.1 (2)
Pattern. 1 (51)
Pattern. 2 (11644)
Block print (135)
letter and number (1603)
Alphanumeric + other (439)
Plant flowers (54)
Face portrait (29)
Animal cartoon (125)
Other patterns (192)

Colour

Red (1742)
Pink (771)
Orange (248)
yellow (146)
green (681)
blue (903)
purple (215)
gray (1619)
black (3323)
white (2928)
Beige (53)
brown (15)
brown.1 (26)
Coffee color (85)
Camel (41)
apricot (45)
blue.1 (31)
Navy blue (1254)
Silver (10)
Color (85)
Secondary color (13324)
Red. 1 (128)
Pink. 1 (7)
Orange. 1 (4)
Yellow.1 (7)
Green. 1 (15)
Blue.1 (42)
Purple. 1 (2)
Gray.1 (30)
Black.1 (258)
White.1 (336)
Beige. 1 (1)
Navy blue. 1 (37)
Color. 1 (30)

Sleeve type

Sleeveless (2137)
Short sleeve (7287)
Cap sleeve (468)
Middle sleeve / 5 minutes slee
ve (12)
6-point sleeve/7-point sleeve/
8-point sleeve (152)
9-point sleeve (14)
Long sleeve (4151)

3.2 Solution Development and Challenges

The obvious generative model was GANs due its superior results as published in many previous work. The latest state of the art GAN is the StyleGAN and its later improvements. However the StyleGAN needs a large amount of data as well as high end computer power. Below are the training requirements for the StyleGAN. It required to run for a long duration of 14 days 22 hours in one GPU for 256*256 images.

Table 3.1 Training time of StyleGAN for optimal results

GPUs	1024×1024	512×512	256×256
1	41 days 4 hours	24 days 21 hours	14 days 22 hours
2	21 days 22 hours	13 days 7 hours	9 days 5 hours
4	11 days 8 hours	7 days 0 hours	4 days 21 hours
8	6 days 14 hours	4 days 10 hours	3 days 8 hours

3.2.1. Computing power

It was known that the physical high end machines are not available for training the neural networks for this research. Therefore, some analysis of the cloud service providers, their configurations and affordability was assessed. In the cloud ML training environments, the affordable resources are limited to 1 high end GPU. Google Colab PRO seemed to be a good paid service compared to Kaggle and PaperSpace. Google Colab PRO provides an interactive environment via the Jupiter interface.

Table 3.2 Cloud Environment Comparison

Provider	Configurations	Pricing	Advantages	Disadvantages / Limitations
Colab PRO	Tesla T4 GPU (15GB) 27 GB System memory Interactive notebook interface	9.79$ / month	fixed price per month high end gpu high memory	resets the machine when idled for a long time depending on the high usage
PaperSpace	Multiple GPU Types (V100)	2 types of subscriptions. Highest end gpus with per hour pricing. Ex: 2.30$/hr for v100 GPU. Notebook based monthly subscription 8$ / month.	Highest end GPUs are available.	Monthly subscription allows lower end GPUs than Colab PRO. Notebooks seem to have bugs and does not work fluidly like in colab
Kaggle	Tesla P100 (16GB / 12 GB)	Free, weekly 30 hours	Free	30 hours per week is the maximum quota

To overcome the reset and disconnection of the remote machine, the snapshots were saved in Google Drive. So that the next time the training can be continued from the saved snapshot.

3.2.2. Environment

Colab comes with pre-installed Python, Tensorflow and PyTorch versions. However, the different implementations like StyleGAN, StyleGAN2, StyleGAN2-ADA and InterfaceGAN required different libraries and Tensorflow/PyTorch versions.

- Conda seems to be a good option here as it can create project specific environments with specific versions and the step is recreatable in any environment and depends only on Conda.
- Conda environments creation and installing specific Tensorflow/PyTorch did not work as expected. Implementations did not run as expected and there were errors thrown that were not easy to debug or fix.
- Even though the authors provided the core library versions each implementation requires, dependencies between different libraries were not captured.
- pip package management based installation was compatible with Colab machines.

3.3 Solution Architecture

Table 3.3 Solution Summary

Dataset	CP-VTON (14221 images of 256 * 192)
Generator Architecture	StyleGAN2-ADA
Attribute Manipulation	Learn the attribute latent direction and linear interpolate
Trained in	Google Colab PRO

3.3.1 Generator Architecture

StyleGAN2-ADA is an improved version of StyleGAN. Additionally it supports data augmentation to make the GAN training possible for smaller datasets. Given the situation of limited GPU resources, using a smaller dataset like CP-VTON with augmentation (StyleGAN2-ADA) is feasible rather than training using several 100k images in a StyleGAN architecture. It allows training the model faster than previous versions on a lower number of samples.

3.3.2. Manipulating the attributes

According to the InterfaceGAN [16], the image attribute manipulation was carried out. Using the methods in InterfaceGAN, latent boundaries of each feature can be learned. However the attributes have to be of binary class. Such as Age(young/old), Smile(yes/no). This can be mentioned as a limitation of this Approach.

The implementation authors did not support the StyleGAN2-ADA network out of the box. Therefore, I had to go through their implementation and created a new script that supports latent manipulation for StyleGAN2-ADA.

Steps

1. Write a script to extract w codes for the seeds (seeds provide the randomness to generate z code) and export as CSV
2. Manually label the sleeve types as long/short for a sample of 500 generated images.
3. Train the linear SVM classifier using w codes as features and sleeve type (short/long) as the binary classification
4. Save weights learned by the SVM as the attribute latent boundary.
5. Write a reusable script to manipulate the given seed's image attributes and save the images from the given start distance to given end distance from the learned boundary from the step above

3.4 Contributions

- Enhancement of InterfaceGAN to support StyleGAN2-ADA.

InterfaceGAN supported the attribute editing of models trained only in official StyleGAN2, StyleGAN. Stylegan2-ada-pytorch GitHub repo was forked and committed a wrapper function in python to make InterfaceGAN compatible with StyleGAN2-ADA.

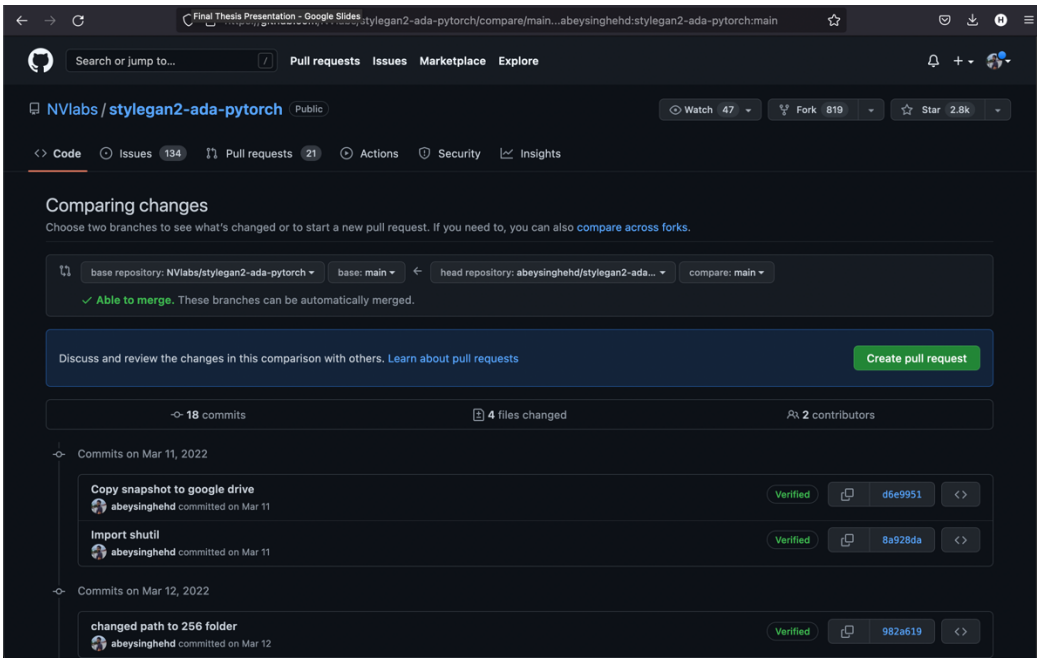

Figure 3.2 StyleGAN2-ADA-PYTORCH fork

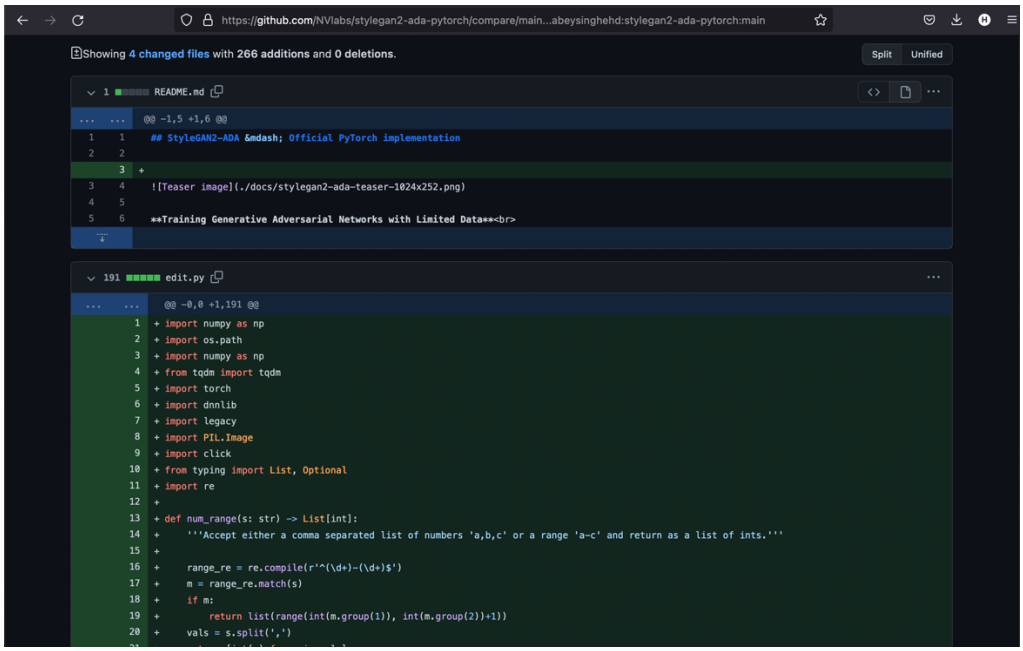

Figure 3.3 Repo screenshot

- Environment setup steps for StyleGAN2-ADA in Colab

StyleGAN2-ADA was not straightforward to run in Google Colab (or other environments) just by using the steps provided in the official GitHub repository. A requirements.txt (Python library management PIP's default way of requirements list declaration) was compiled so that the environment will be ready to run the StyleGAN2-ADA in one step.

To prepare the requirements file, just have to create a new .txt file name requirements and include the below text.

```
install
click
requests
tqdm
pyspng
ninja
imageio-ffmpeg==0.4.3
torch==1.7.1+cu110
torchvision==0.8.2+cu110
torchaudio===0.7.2
```

The environment is recommended to have python 3.7 installed. Then the below command in terminal should install all the libraries.

```
!pip install -r ./requirements.txt -f https://download.pytorch.org/whl/torch_stable.html
```

- Trained GAN model is published in a GitHub repo

The trained StyleGAN2-ADA-Pytorch model is published in the following repo : https://github.com/abeysinghehd/garment-generation-research-resources
This model can be used to train further or to generate new garment images

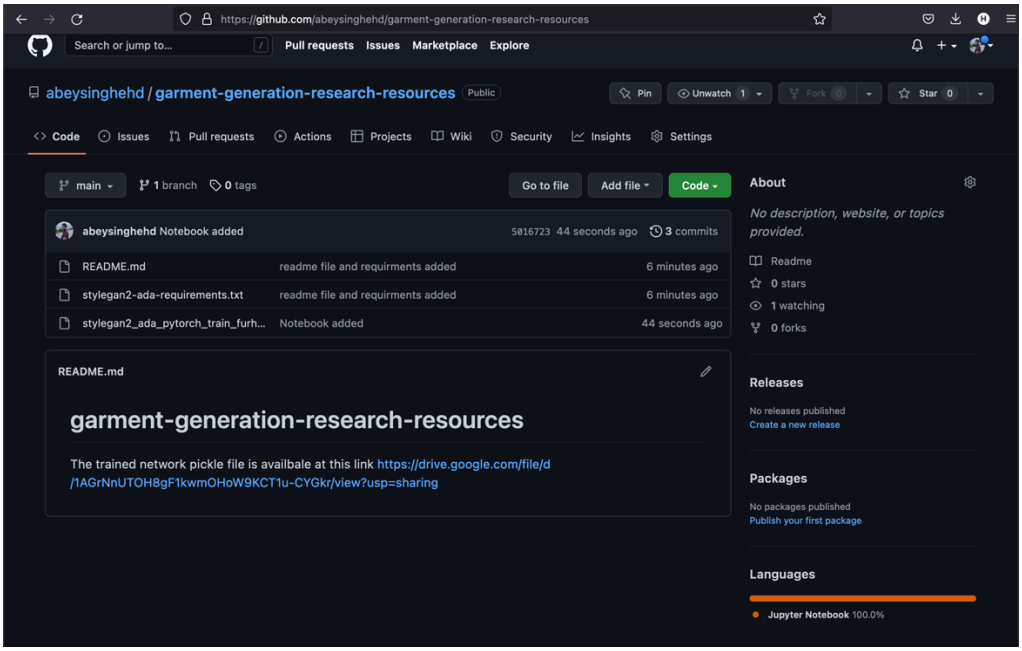

Figure 3.4 GitHub repo : published resources

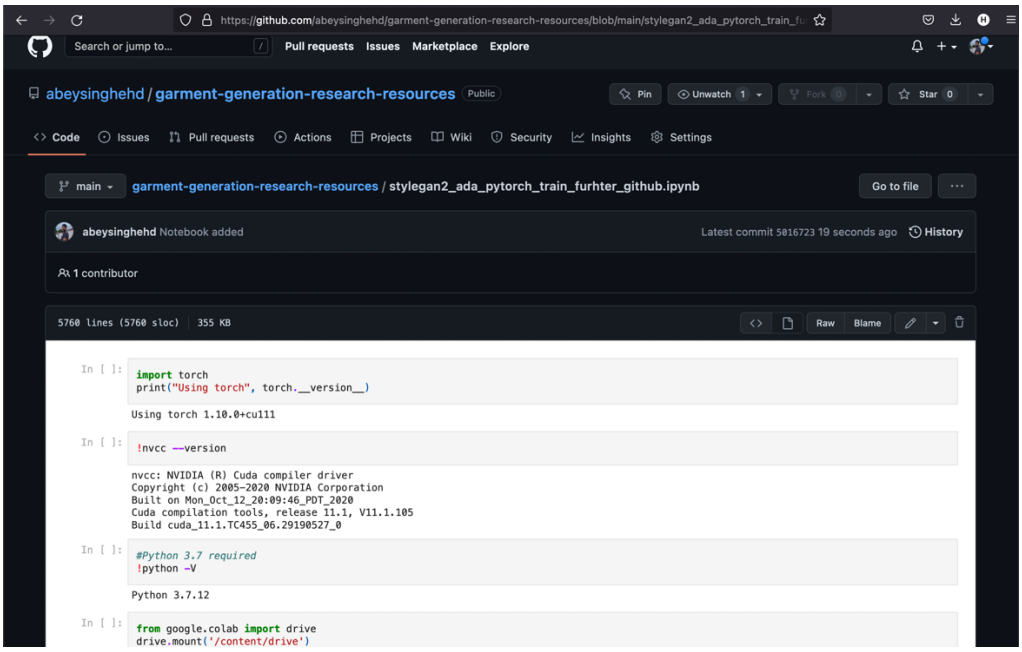

Figure 3.5 GitHub repo: published resources 2

4. RESULTS

4.1 Synthesized Dresses

The model trained for approximately 4.5 days. There were interruptions from Colab due to user inactivity. 4.5 days is not continuous training but the cumulative time of resumed training after the interruptions.

Some generated images from the trained model are shown below. From the example below, it is evident that the model has learned different patterns, colours, types, sleeve lengths, neck types and even side angle display of the dresses from the training data set.

Figure 4. 1 Synthesized dress results

Figure 4.2 Synthesized dress variations

4.1 Attribute Manipulation Results

Linear interpolation along the learned sleeve length attribute boundary results are shown below.

Figure 4.3 Sleeve length manipulation

Granular control of sleeve length editing can be seen clearly in the results. However, the colour, texture features seem to be entangled with the sleeve length attribute in the first two examples.

4.2 Comparison with previous work

Figure 4.4 Results from the [10]

Figure 4.5 Result from new method

- Previous attribute manipulation results could control either no sleeve or longest sleeve garment designs. Clearly this method can control the length garment more.
- Previous method could not generate a clear garment design when it tries to control the sleeve length feature. This method can generate more realistic images with clearer features.
- However, the previous work [19] have not provided any benchmarks or scores to compare with this method.

5. CONCLUSION

As shown in the results section, the state of the art GAN models are applicable to the dress domain very well. The new ways of disentangling the latent space by intermediate w space and mapping network seems to have an effect given the ability to manipulate the sleeve length of the given model.

However, the manipulation method used here is limited to binary classifiable attributes such as long sleeve - short sleeve, texture-plain. Given the ability of the manipulation accuracy and having small changes on the other attributes, the manipulation method was well suited for the w space of the StyleGAN2-ADA.

The solution should be scalable to more detailed 1024*1024 dress images. Due to the limitations of high end GPU availability, that scale was hard to achieve in this research. It is scalable in binary attributes as well. Attributes such as having buttons, texture can be trained. The generated images have to be labelled manually or should be labeled using a binary attribute classifier. Given a good dataset with suitable attribute data, a binary attribute classifier can be trained to mitigate the need to label generated images manually to train the SVM classifier. With a larger amount of attribute binary values to train the SVM classifier, it will be able to learn the attribute boundaries more accurately, therefore the entanglement on feature modification can be improved further.

Compared to previous fashion attribute manipulation work like Design-AttGAN [19] relied on the conditional training ability of the generator models. The synthesized images were not crisp when conditioned on features and the attribute manipulation was not possible at a granular level like in the results of this research.

References

[1] T. Karras, S. Laine and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019.

[2] K. E. Ak, J. H. Lim, J. Y. Tham and A. A. Kassim, "Attribute manipulation generative adversarial networks for fashion images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[3] Y. R. Cui, Q. Liu, C. Y. Gao and Z. Su, "Fashiongan: Display your fashion design using conditional generative adversarial nets," in *Computer Graphics Forum*, vol. 37, Wiley Online Library, 2018, pp. 109-119.

[4] S. Zhu, R. Urtasun, S. Fidler, D. Lin and C. Change Loy, "Be your own prada: Fashion synthesis with structural coherence," in *Proceedings of the IEEE international conference on computer vision*, 2017.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial networks," *Communications of the ACM,* vol. 63, pp. 139-144, 2020.

[6] N. Jetchev and U. Bergmann, "The conditional analogy gan: Swapping fashion articles on people images," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017.

[7] Pan, Zhaoqing, W. Yu, A. Asifullah, A. Khan, F. Yuan and Y. Zheng, "Recent progress on generative adversarial networks (GANs): A survey," *IEEE Access,* vol. 7, pp. 36322-36333, 2019.

[8] Z. Wang, Q. She and T. E. Ward, "Generative adversarial networks in computer vision: A survey and taxonomy," *ACM Computing Surveys (CSUR),* vol. 54, pp. 1-38, 2021.

[9] A. Radford, L. Metz and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434,* 2015.

[10] M. Arjovsky, S. Chintala and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, 2017.

[11] T. Karras, T. Aila, S. Laine and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196,* 2017.

[12] Wikipedia, "Vanishing gradient problem," [Online]. Available: https://en.wikipedia.org/wiki/Vanishing_gradient_problem. [Accessed 05 July 2022].

[13] M. Wiatrak, S. V. Albrecht and A. Nystrom, "Stabilizing generative adversarial networks: A survey," *arXiv preprint arXiv:1910.00927,* 2019.

[14] Google, "Common Problems," [Online]. Available: https://developers.google.com/machine-learning/gan/problems. [Accessed 05 July 2022].

[15] S. Barratt and R. Sharma, "A note on the inception score," *arXiv preprint arXiv:1801.01973,* 2018.

[16] Y. Shen, C. Yang, X. Tang and B. Zhou, "Interfacegan: Interpreting the disentangled face representation learned by gans," *IEEE transactions on pattern analysis and machine intelligence,* 2020.

[17] C. Yu, Y. Hu, Y. Chen and B. Zeng, "Personalized fashion design. In Proceedings of the IEEE/CVF International Conference on Computer Vision," 2019.

[18] Z. He, W. Zuo, M. Kan, S. Shan and X. Chen, "Attgan: Facial attribute editing by only changing what you want," *IEEE transactions on image processing,* vol. 28, pp. 5464-5478, 2019.

[19] C. Yuan and M. Moghaddam, "Garment Design with Generative Adversarial Networks," *arXiv preprint arXiv:2007.10947,* 2020.

[20] Q. Ping, B. Wu, W. Ding and J. Yuan, "Fashion-AttGAN: Attribute-aware fashion editing with multi-objective GAN," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2019.

[21] W. Xian, P. Sangkloy, V. Agrawal, A. Raj, J. Lu, C. Fang, F. Yu and J. Hays, "Texturegan: Controlling deep image synthesis with texture patches," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[22] A. Elgammal, B. Liu, M. Elhoseiny and M. Mazzone, "Can: Creative adversarial networks, generating" art" by learning about styles and deviating from style norms," *arXiv preprint arXiv:1706.07068,* 2017.

[23] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford and X. Chen, "Improved techniques for training gans," *Advances in neural information processing systems,* vol. 29, 2016.