DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING, SECOND CYCLE, 30 CREDITS

STOCKHOLM, SWEDEN 2019

Text-to-image Synthesis for Fashion Design

ZHENGRONG YI

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Text-to-image Synthesis for Fashion Design

ZHENGRONG YI

Master in Information and Network Engineering
Date: October 30, 2019
Supervisor: Mårten Björkman and Ali Ghadirzadeh
Examiner: Danica Kragic Jensfelt
School of Electrical Engineering and Computer Science
Swedish title: Text-till-bild-syntes för modedesign

Abstract

Generating high-quality images from textual descriptions is an active research direction in image generation and has aroused great interest in fashion design. The synthesized image should be consistent with the meaning of text as well as being of acceptable quality. Generative Adversarial Networks (GANs) successfully show the capability of synthesizing sharper images compared to other generative models. Many GAN-based methods have been developed to deal with text-to-image synthesis, generating compelling images on simple non-fashion datasets. Nevertheless, inherent problems of GANs and more complex datasets greatly increase the difficulty of synthesizing realistic and high-resolution images.

In this degree project, with the aim to study the respective impacts of network architectures and training data on the performance of text-to-image synthesis, two GAN-based algorithms are adopted, namely, Attentional Generative Network (AttnGAN) and Stacked Generative Network (StackGAN). They are applied on two fashion datasets separately, i.e., FashionGen and Fashion Synthesis. The models are evaluated using Fréchet Inception Distance (FID). For FashionGen, AttnGAN outperforms StackGAN with better FID and synthesizes high-quality images in most common categories of fashion items. For Fashion Synthesis, StackGAN achieves better FID, but some generated images are less realistic.

Sammanfattning

Att generera högkvalitativa bilder från textbeskrivningar är en aktiv forskningsriktning inom bildgenerering och har väckt stort intresse inom modedesign. Den syntetiserade bilden ska vara förenlig med betydelsen av texten och vara av acceptabel kvalitet. *Generativa Adversarial Networks* (GAN), visar framgångsrikt möjligheten att syntetisera skarpare bilder jämfört med andra generativa modeller. Många GAN-baserade metoder har utvecklats för att hantera text-till-bild-syntes, vilket genererar övertygande bilder från enkla icke-modebaserade dataset. Däremot ökar inneboende problemen med GAN:s och mer komplexa dataset ökar svårigheten att syntetisera realistiska och högupplösta bilder.

I detta examensarbete, med syftet att studera respektive inverkan av nätverksarkitekturer och träningsdata på prestanda för text-till-bild-syntes, tillämpas två GAN-baserade algoritmer, nämligen *Attentional Generative Network* (AttnGAN) och *Stacked Generative Network* (StackGAN). De tillämpas separat på två mode-databaser, *FashionGen* och *Fashion Synthesis*. Modellerna utvärderas med hjälp av *Fréchet Inception Distance* (FID). För *FashionGen* överträffar AttnGAN StackGAN med bättre FID och syntetiserar högkvalitativa bilder i de vanligaste kategorierna av modeobjekt. För *Fashion Synthesis* uppnår StackGAN bättre FID, men vissa genererade bilder är mindre realistiska.

Contents

Chapter 1

Introduction

We live with fashion all around us, from products like clothing, footwear, accessories, makeup, etc. to styles such as hairstyle, decor, lifestyle, etc. The fashion industry has long been one of the biggest businesses in the world, worth approximately 3,000 billion dollars as of 2018 according to FashionUnited, accounting for 2% of the global GDP. It includes the design, manufacturing, logistics, marketing and retailing. Nowadays, technology plays an indispensable role in most aspects of our society. Wherein, artificial intelligence (AI) is promoting a variety of professions, having a great prospect in massive applications. It also becomes increasingly appealing and influential in the fashion industry, delivering new solutions in every part. For instance, AI chatbots are already in use by lots of fashion retailers to connect the customers, offer personalized recommendations, reduce costs on customer service, etc.

The field of AI research was founded in 1956. Since then, it has gone through three ups and two downs. The current also the third boom of AI was firstly driven by advanced machine learning and then deep learning, a branch of machine learning. This success cannot happen without access to huge quantity of data and tremendous increase in computing power. In traditional machine learning techniques, human-designed representations are usually required to reduce the complexity of data and make patterns more visible for learning algorithms to work. However, the extraction of those hand-engineered features rely on domain expertise and are inflexible. The greatest strength of deep learning algorithms is that they can automatically learn good features from data from low to high level in hierarchical architectures. The multiple levels of representations also facilitate transfer learning and multi-task learning. Numerous successful deep learning algorithms have been developed when solving complex problems, e.g., convolutional neural networks

(CNNs) in computer vision, long short-term memory (LSTM) and other deep learning based methods in natural language processing (NLP).

Design is the first part in the fashion industry, which is also considering the use of AI for trend prediction that indirectly affects the designing process, or for direct product design, i.e., fashion image synthesis. Our project aims at the latter one, where the AI techniques are still nascent.

1.1 Motivation

Generating high-quality images from textual descriptions is a challenging research problem. The synthesized image should reflect the textual description as well as being of acceptable quality. Applying this technique to fashion design has practical importance. Firstly, it gives designers an automated tool for rapidly making and prototyping novel designs. Furthermore, generating different designs of fashion items by human-written descriptions will allow us to discover and choose our preferred designs conveniently and intuitively.

Generative Adversarial Networks (GANs) [1] have aroused wide interest recently because the synthesized images are sharper compared to other deep generative models [2]. Conditioning GANs on extra information has proven capable of directing the data generation process [3]. Such condition variables can be class labels [4], attributes [5], images [6], and texts [7], [8], among which directly using human-written sentences describing images shows great potential, although facing big challenges.

Therefore, it is interesting to study text-to-image synthesis algorithms, especially GAN-based approaches, and also apply some advanced techniques on fashion datasets to compare their performance and probably gain some insights in this application.

1.2 Research questions and objective

The research questions are formed as follows: What are the limitations of state-of-the-art methods at generating novel designs of fashion items conditioned on text descriptions? What is the best performance one can expect from such methods? How the performance can be improved by devising new learning frameworks and/or network architectures and/or training data?

In terms of the objective, a neural network generative model should be developed to generate high-quality images of fashion items conditioned on text descriptions, which can be controlled by an arbitrary low-dimensional la-

tent variable to choose different designs given the same text. Additionally, limitations and performance of state-of-the-art methods should be thoroughly studied. By devising new learning frameworks and/or network architectures and/or training data, the performance can be improved quantitatively and qualitatively.

1.3 Ethical and social considerations

As an application of AI, text-to-image synthesis can provide an automated tool to assist the work of designers, which can speed up the design process and thus add more value to fashion industry. By decreasing the cost of fashion design using this technique, some fashion companies would be able to put more budgets on the quality of fashion products, services, etc., to bring consumers better experience.

Like any business, with increasing mass production of fashion products and global reach, sustainability has become an urgent issue in fashion industry. On the one hand, the text-to-image synthesis methods would hopefully reduce the usage of textile and other raw materials during the design process, since a part of designs can be excluded by looking at the synthesized images. On the other hand, it may have opposite effects. Traditionally, fashion has been defined by constant change, which is bound to emergence of new designs. So, if the methods are excessively used, it can rapidly create plenty of novel designs which can easily become massive products afterwards.

Hence, if the text-to-image synthesis algorithms are ready to be applied in the fashion design, the designers and fashion companies should agree to and abide by the terms of sustainable development. They together with relevant sectors should invest more into research to find more sustainable ways of fashion design.

1.4 Outline

In Chapter 2, a review will be presented, where the limitations and performance of state-of-the-art methods for text conditioned image synthesis will be summarized. Besides, several basic models acting as the building blocks of employed approaches and some machine learning techniques will be briefly introduced.

In Chapter 3, the respective principles of two selected methods, namely, AttnGAN and StackGAN, will be elaborated, including the architectures, the

objectives, etc. Then, we will introduce the two adopted fashion datasets, relevant experiment settings and evaluation metrics.

In Chapter 4, the methods in Chapter 3 are implemented and applied on the two fashion datasets. Some intermediate results will be presented, e.g., the choices of key hyper-parameters in the models. Besides, synthesized images will be displayed and used to evaluate algorithm performance.

In Chapter 5, we will analyze the results in Chapter 4 to better understand the strengths and limitations of each method and each dataset. In addition, we will propose some work in the future that can be done to complement our work.

In Chapter 6, we will summarize our work and main points in this project.

Chapter 2

Background

In this chapter, we will have a review of some basic knowledge of models that are building blocks of approaches in Chapter 3 as well as some relevant techniques in machine learning. Then, existing text-to-image synthesis algorithms and fashion synthesis algorithms will be summarized. At last, several challenges of the text-to-image synthesis for fashion design will be discussed.

2.1 Theoretical knowledge

2.1.1 Generative adversarial networks

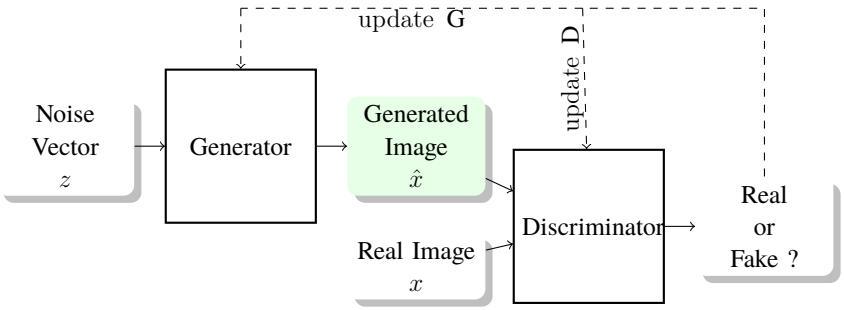


Figure 2.1: The architecture of GAN

Generative adversarial network (GAN) is a framework for training generative models, which is shown in Figure 2.1. The idea of GANs is to make two networks compete against one another: a generative model G (generator) that tries to learn the distribution of real data and a discriminative model D (discriminator) that estimates the probability that a sample comes from the training data

rather than the generator. The generator takes a noise vector z as input, which is usually sampled from the standard normal distribution $p_z(z)$, and it generates an image $\hat{x}$. The discriminator takes an image as input, which is either $\hat{x}$ or a real image from the training data distribution $p_{data}(x)$, and it outputs a scalar. If it is high, the input was real. Otherwise, the input was fake (from the generator). A minmax objective function $V(G, D)$ is used to train both models simultaneously:

$$\min_G \max_D V(G, D) = \mathbf{E}_{x \sim p_{data(x)}}[\log D(x)] + \mathbf{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]. \quad (2.1)$$

This encourages generator to fit $p_{data(x)}$ so as to fool discriminator with the generated sample $\hat{x}$. Both of them are trained by backpropagating the loss in Eq.(2.1) through their respective models to update the parameters.

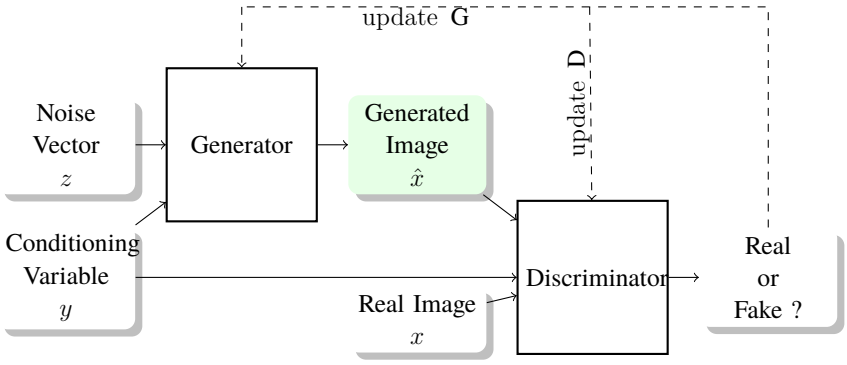


Figure 2.2: The architecture of conditional GAN

Conditional generative adversarial network (CGAN) is an extention of GAN where both generator and discriminator take an additional vector of information y as input, as shown in Figure 2.2. This can be any kind of auxiliary information, e.g., class labels, texts, etc. In the generator, the noise vector z and additional vector y are combined into joint representation. The objective function thus becomes

$$\min_G \max_D V(G, D) = \mathbf{E}_{x \sim p_{data}(x)}[\log D(x|y)] + \mathbf{E}_{z \sim p_z(z)}[\log(1 - D(G(z|y)))]. \quad (2.2)$$

This model allows the output of the generative model to be controlled by the conditioning variable y. In our task, y will be text describing the fashion item in the image.

2.1.2 Text encoders

The encoder in machine learning is a network that takes the input sequence and outputs the feature representations of it. A piece of text consists of a list of words, numbers, special characters and punctuation marks. The text encoder is a block used to encode the raw text into its numerical features.

Text tokenization

The first step of encoding is text tokenization that transforms the text into tokens. Depending on the requirements of tasks, a tokenizer can be set to filter out the punctuation (or not), unify the letters into lower-cases (or not), etc. Then, the sentences will be split into a bunch of tokens. Sometimes, it is required to have same sequence length for each text. In this case, the sequence of tokens will be either padded with a predefined padding token at the end of sequence or partly removed.

Word embeddings

The second step is to map each token into its corresponding word embedding. Word embedding methods represent raw words (tokens) as continuous vectors, which can capture lexical and semantic properties of words [9]. Benefiting from using dense and low-dimensional vectors as opposed to symbol representations and one-hot encoding, etc., word embeddings are more efficient in computation and more powerful in generalization.

There are multiple ways to learn word embeddings from raw text. One is using an embedding layer, which is actually a weight matrix. One dimension of it is the size of the embedding space, while another dimension is the number of tokens in the corpus. It is initialized with small random numbers and can be learned jointly with a neural network in a supervised way using backpropagation for a specific task like text classification. By this means, it may require a large amount of data for training and thus can be slow, but it is able to learn embeddings targeted to the specific text and the specific task simultaneously.

Besides, various algorithms are proposed to learn word embeddings more efficiently. Word2Vec is a prediction-based supervised model developed by Tomas Mikolov, $et\ al$ [10]. It is alterable by choosing one of the following components: the Continuous Bag-of-Words model (CBOW) and the Skip-Gram model. CBOW learns the embeddings by predicting target word (e.g. 'mat') based on context words ('the cat sits on the'). On the contrary, the skip-gram model predicts context words given the target word. A context window

(a window of neighboring words) will shift through the text and each shift is a training example. Since this approach is computationally-efficient, it facilitates the learning of larger-dimension word embeddings with much larger text dataset. Nevertheless, the usage of context window restricts the learning from the global statistics of the text. The unsupervised learning algorithm Global Vectors (GloVe) [11] constructs a global word-word co-occurrence matrix to capture the global statistics using matrix factorization techniques and also leverages the predition-based method in Word2Vec, resulting in generally better word embeddings.

Apart from using these techniques to learn word embeddings from scratch, their pre-trained embeddings are available for reusing in other language-related tasks in static or updated way to possibly improve the performance and reduce training time.

Finally, text embeddings can be constructed based on word embeddings for phrases, sentences or paragraphs. This can be done by simple operations on vectors and matrices, such as unweighted averaging. Some methods incorporate recurrent neural networks (RNNs) and/or convolutional neural neworks (CNNs) to model the sentences, etc. These two types of networks will be briefly introduced in the next two subsections.

2.1.3 Recurrent neural networks

Recurrent neural networks (RNNs) are known for the capacity to model context dependencies in inputs of arbitrary length, since they have internal state that can memorize the results of previous computations and use that information in the current computation. This makes them suitable for processing sequential data. Furthermore, bidirectional RNNs can incorporate information from both preceding data and following data, increasing the amount of input information.

In order to overcome the vanishing gradients and long-term dependency problems of traditional RNNs, long short-term memory (LSTM) were invented. Some typical variants are bidirectional LSTM (bi-LSTM), gated recurrent units (GRUs), bidirectional gated recurrent units (bi-GRUs), etc.

2.1.4 Convolutional neural networks

Convolutional neural networks (CNNs), especially deep convolutional networks have been widely used in the field of computer vision for image classification, object detection, instance segmentation, etc. CNNs can take advantage

of shape information in the input image by applying a series of filters to the raw pixels to extract and learn higher-level features. The hidden layers of a CNN typically consist of convolutional layers, non-linear layers (i.e. activation functions), pooling layers, fully connected layers and normalization layers.

Diverse architectures based on CNNs are proposed to achieve higher performance in image classification tasks, such as Alexnet, VGGNet, GoogLeNet, Inception V3, etc. It has been observed that the architectural improvements resulting in higher classification accuracy can be utilized to improve performance in most other computer vision tasks, since they all rely on high-quality latent features. This can also be found in our adopted approaches that inception V3 is used as the image encoder to map the synthesized images from the generative model to semantic vectors to be used in the measurement of the text-image similarity. In addition, inception V3 is the model used to compute the metrics, inception score (IS) and Fréchet Inception Distance (FID), to evaluate the quality of generated image samples from GANs.

2.1.5 Attention mechanisms

Attention mechanism was initially introduced by Bahdanau $et\ al$ [12] to tackle the problem of the Seq2Seq model [13] that it cannot handle long input sequence well. Since then, diverse variants of attention mechanism have been emerged and effectively used in machine translation, dialogue generation etc. Recently, this concept has also been extended to the field of computer vision, e.g., image generation.

The Seq2Seq model usually has a encoder-decoder structure, where the encoder takes a sequence of vectors $\mathrm{x} = (x_1, ..., x_{\mathrm{T}_x})$ and produces a context vector c, based on which the decoder generates an output sequence $\mathrm{y} = (y_1, ..., y_{\mathrm{T}_y})$. Considering the case where the encoder and decoder are RNNs, the hidden states of encoder and decoder are denoted as h and s, respectively. The context vector c can be, for example, the last hidden state of the encoder. In most previous work, the encoder bore the burden to encode all the information in the variable-length sequence x into the fixed-length vector c, which is the reason why this model cannot perform well on long input sequences.

In [12], a sequence of context vectors $\mathrm{c} = (c_1, ..., c_{\mathrm{T}_y})$ are constructed to alleviate that problem. At time i, a distinct context vector c_i is used to compute y_i. c_i is a weighted sum of the encoder hidden states given by

$$c_i = \sum_{j=1}^{\mathrm{T}_x} \alpha_{i,j} h_j, \quad \text{where} \quad \alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{k=1}^{\mathrm{T}_x} \exp(e_{i,k})}, e_{i,j} = a(s_{i-1}, h_j). \quad (2.3)$$

The special weight $\alpha_{i,j}$ or $e_{i,j}$ computed by the alignment model a reflects the importance of h_j of encoder with respect to the previous hidden state s_{i-1} of decoder. With this mechanism, the decoder can dynamically retrieve the relevant information in the input sequence, that is, decide the parts in the input to pay attention to.

The alignment model can be parametrized by a feedfoward neural network that will be jointly trained with the rest of network. This is known as the additive attention. Besides, by changing the alignment models, different attention mechanisms are derived, such as dot-product attention, scaled dot-product attention, content-base attention, location-base attention, etc.

2.2 Related work

2.2.1 Text-to-image synthesis algorithms

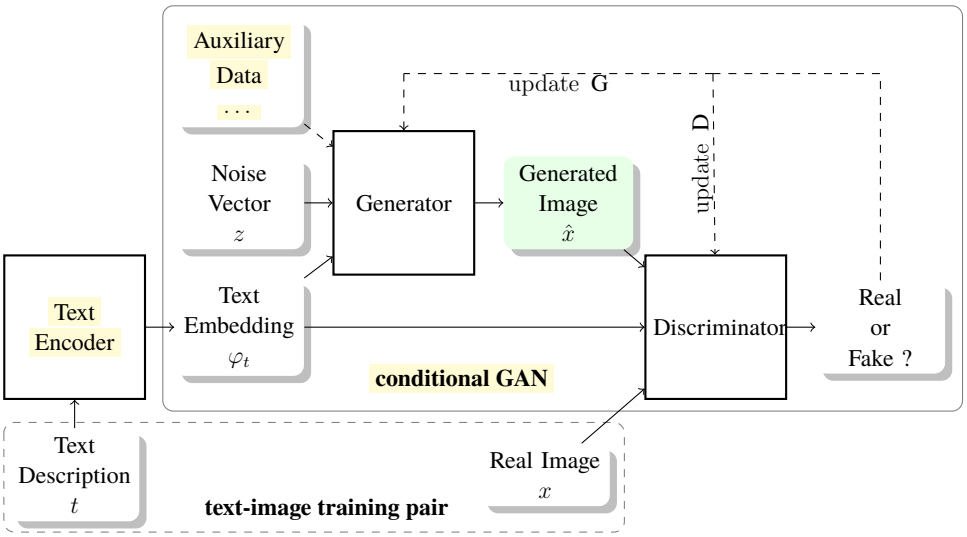


Figure 2.3: The architecture of text-to-image synthesis system

Text-to-image synthesis is an active research direction in image generation, mostly benefiting from GANs. It is desired to generate photo-realistic and high-resolution images. The realism refers to the property that the generated images should be indistinguishable from those in the used dataset. Meanwhile, the resolution of images in the dataset for reference will determine the highest resolution the synthesised images can have. A general framework of text-to-image synthesis system is depicted in Figure 2.3, which indicates three major

research objectives to improve the algorithms in this field, i.e., auxiliary data, text encoder and the structure of conditional GAN.

The first one is adopting auxiliary data apart from text-image pairs to enhance the input of network such as object location [14], dialogues on the textual descriptions [15], and visual knowledge [16]. They usually follow up some related work and leave the main network structure unchanged. However, these don't boost the performance much.

In addition, some work are related to the design of text encoder that transforms original descriptions into embeddings to be fed into the condional GANs. For example, CNN-RNN and its variants are proposed by Reed $et\ al$ [17]. It is found that the choice of text encoder can have a large impact on the quality of synthesized images, though it is more related to natural language processing and not the emphasis of research in text-to-image synthesis.

A majority of work focuses on the structure of conditional GAN, either adapting a widely used GAN by conditioning it on text or devising a new one. Reed $et\ al$. [18] first succeeded in generating visually-plausible 64×64 images with a Deep Convolutional GAN (DCGAN) [19] conditioned on the sentences, while lacking necessary details. Zhang $et\ al$. [7] proposed StackGAN consisting of multi-stage conditional GANs for generating high resolution images (e.g., 256×256) through a sketch-refinement process. This breakthrough inspired many text-to-image generative models afterwards. In contrast to the models where training is stage-by-stage [7] [20], models like [21] with hierarchically-nested structure can be efficiently trained in an end-to-end fashion. Recently, Xu $et\ al$. [8] explored attention mechanism on two novel components in their AttnGAN, so that it has not only sentence level conditioning as usual but also word level conditioning to better reflect the text meanings in the generated images. Their work achieves state-of-the-art Inception Scores (introduced in subsection 3.4.1) on CUB [22] and COCO [23] datasets.

2.2.2 Fashion synthesis algorithms

Text-to-image synthesis in the context of fashion is an interesting application in fashion synthesis [24], [25], [26]. So far, only a small amount of work on text-to-image synthesis for fashion design with GANs has been carried out. A close research is language based image editing. Zhu $et\ al$. [27] trained a two-stage GAN to generate new outfits onto a given image conditioned on textual description, rendering better structural coherence including consistent appearance and pose. However, the resolution (128×128) is restricted by their

dataset and segmentation maps as additional information are costly to obtain. Without using any spatial constraints to fulfill the same purpose, G$\ddot{u}$nel $et\ al.$ [28] incorporated a conditioning mechanism (FiLM) to fuse visual representations and textual representations, yet having some degradation as a result. In general, human appearance and pose, clothing shape and texture are two groups of elements in fashion synthesis. Part of these elements need to be unchanged simultaneously or disentangled depending on different application scenarios. In our task, the target is not the human appearance and pose, but the fashion design corresponding to the text itself describing clothing shape and texture, etc.

2.2.3 Discussion on the challenges

The challenges of our task are basically derived from inherent problems of GANs, demands for generating high-quality images, and issues of datasets.

Regarding the first aspect, the training of GANs is known to be unstable and often results in 'mode collapse' (i.e., the generator learns to generate samples from only a few modes of the distribution) [29]. Wide techniques have been developed to mitigate the training instability and improve the sample diversity by designing new architectures with new learning objectives, using regularization methods (e.g., spectral normalization [30]) and balancing the convergence between the generator and discriminator (e.g., TTUR [31]), which are partly considered in the aforementioned text-to-image synthesis models or can be employed to improve them.

Secondly, the difficulty of generating higher-resolution images using GANs increases significantly. This is seen as the consequence that natural image distribution and the implied model distribution may not overlap in high dimensional space [32].

Furthermore, the image quality can also be linked to the quality and complexity of datasets. Very recently, Rostamzadeh $et\ al.$ [33] introduced Fashion-Gen, a new fashion dataset specifically designed for text-to-image generation, intending to remedy the image shortcomings possibly caused by the lack of high-quality datasets. Existing methods have shown the capability of synthesizing compelling scenes on simple datasets of birds and flowers, while numerous experiments indicate that the models generally tend to generate malformed and interpretable images on complex datasets like COCO, where each image contains diverse objects.

Chapter 3

Methods

In this chapter, two advanced GANs are chosen, namely, Stacked Generative Adversarial Network (StackGAN) [7] that first devises the stacked stages to generate images from low to high resolution and becomes inspiration of others work; and Attentional Generative Adversarial Network (AttnGAN) [8] which integrates attention mechanism in its two novel components and outperforms others on two non-fashion benchmarks. We are going to elaborate each of them in the following sections.

3.1 Stacked generative adversarial network

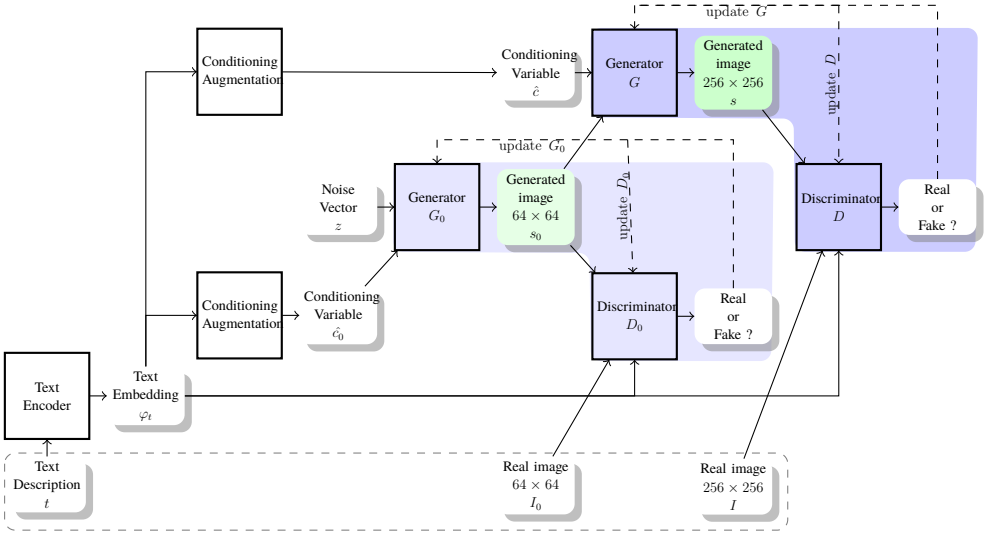


Figure 3.1: The architecture of StackGAN

As shown in Figure 3.1, through an externally pre-trained text encoder, the text description t is transformed into the text embedding φ_t. The two main contributions of StackGAN are the pre-processing of the original text embeddings, known as the Conditioning Augmentation (CA) technique, and the sketch-refinement process that decomposes the text-to-image synthesis into two stages. In the first stage, the Stage-I GAN (G_0 and D_0) learns the primary shape and colors of the object from the conditional text description concatenated with a random noise vector, producing a low-resolution image. Then, the Stage-II GAN (G and D) corrects the flaws in the image generated from the first stage and provides more details omitted by the Stage-I GAN by learning the text description again, resulting in a higher-resolution image.

3.1.1 Conditioning augmentation

Previously, some algorithms use fixed conditioning text variable combined with the noise vector as the input of the generator. However, since the dimension of a text embedding is usually high but the quantity of data is limited, it usually causes discontinuity in the latent conditioning manifold which will hinder the training of generator.

By contrast, CA blocks are added before the generators to generate variant conditioning text variables as can be seen in Figure 3.1. More specifically, in the Stage-I, the variant conditioning text variable $\hat{c}_0$ is a Gaussian random vector sampled from an independent multivariate Gaussian distribution $\mathcal{N}(\mu_0(\varphi_t), \Sigma_0(\varphi_t))$, where $\mu_0(\varphi_t)$ and $\Sigma_0(\varphi_t)$ are the mean vector and diagonal covariance matrix (where the diagonal elements are equal to σ_0) of embedding φ_t, respectively. Both of them are jointly learned with the rest of the network. Firstly, φ_t is fed into a fully connected layer to obtain μ_0 and σ_0. Secondly, $\hat{c}_0 = \mu_0 + \sigma_0 \odot \epsilon$, where $\odot$ is the element-wise multiplication and ϵ is from $\mathcal{N}(0, I_n)$ (I_n is the identity matrix with n equal to the dimension of φ_t). Finally, $\hat{c}_0$ is concatenated with a standard Gaussian noise vector as the final input of G_0.

Similar to the Stage-I, the variant conditioning text variable $\hat{c}$ is generated in the Stage-II. The difference consists in that φ_t is fed into another fully connected layer to generate different means and standard deviations, allowing the Stage-II GAN to capture the omitted information in the Stage-I GAN.

With these steps, it helps to increase the number of text-image training pairs and introduce randomness, which proves to be effective in improving the diversity of the generated images and stabilizing the training of the conditional GANs. Additionally, a regularization term is added into the objectives when

training the generators to further smooth the conditioning manifold, which is the Kullback-leibler (KL) divergence as defined below:

$$D_{KL}(\mathcal{N}(\mu(\varphi_t), \Sigma(\varphi_t))||\mathcal{N}(0, I_n)). \tag{3.1}$$

3.1.2 Two-stage GANs

The model and objectives of the Stage-I GAN

Through a series of up-sampling, G_0 can generate a low-resolution image. In terms of D_0, the real image or the generated image is down-sampled and φ_t is compressed and spatially replicated so that they have the same spatial dimensions to be concatenated as a single tensor. This tensor will pass a 1×1 convolutional layer to extract the joint features of image and text and then these features will be connected to a single node to output the decision. The matching-aware discriminator [18] is adopted in both stages in order to render better alignment between the synthesized images and corresponding texts. Specifically, the discriminator will take positive pairs and negative pairs during training. The former one consists of real images and corresponding descriptions, while the latter one not only includes generated images and corresponding descriptions but also real images and mismatched descriptions.

With the text-image training pair (t, I_0) from the true data distribution p_{data} and the noise vector z from the distribution p_z, Equation 2.2 is translated into two objectives to train the discriminator D_0 and generator G_0 by alternatively maximizing $\mathcal{L}_{D_0}$ in Equation 3.2 and minimizing $\mathcal{L}_{G_0}$ in Equation 3.3,

$$\begin{aligned}\mathcal{L}_{D_0} =& \mathbf{E}_{(t,I_0)\sim p_{data}}[\log D_0(I_0, \varphi_t)]+ \\ & \mathbf{E}_{z\sim p_z, t\sim p_{data}}[\log(1 - D_0(G_0(z, \hat{c}_0), \varphi_t))],\end{aligned} \tag{3.2}$$

$$\begin{aligned}\mathcal{L}_{G_0} =& \mathbf{E}_{z\sim p_z, t\sim p_{data}}[\log(1 - D_0(G_0(z, \hat{c}_0), \varphi_t))]+ \\ & \lambda D_{KL}(\mathcal{N}(\mu(\varphi_t), \Sigma(\varphi_t))||\mathcal{N}(0, I_n)),\end{aligned} \tag{3.3}$$

where λ is a regularization parameter (1 by default).

The model and objectives of the Stage-II GAN

In Equation 2.2, other than the conditioning text variable, the other input of GAN should be the noise vector, but here the generated image s_0 from the Stage-I is used, i.e., $G_0(z, \hat{c}_0)$, assuming that the randomness is preserved by s_0. It is down-sampled to have the same spatial dimension of the text features

spatially replicated from $\hat{c}$. The text features and image features are then concatenated along the channel dimension and fed into residual blocks to learn multi-modal representations. In the end, they are up-sampled to generate a high-resolution image with more vivid details and less defects. As for the discriminator D in the Stage-II, its structure is same as that in the Stage-I except that more down-sampling blocks are used on the larger synthesized image. Similarly, the objectives of the discriminator D and generator G are defined in Equation 3.4 and Equation 3.5, respectively.

$$\begin{aligned}\mathcal{L}_D =& \mathbf{E}_{(I,t)\sim p_{data}}[\log D(I, \varphi_t)] + \\ & \mathbf{E}_{s_0\sim p_{G_0}, t\sim p_{data}}[\log(1 - D(G(s_0, \hat{c}), \varphi_t))],\end{aligned} \tag{3.4}$$

$$\begin{aligned}\mathcal{L}_G =& \mathbf{E}_{s_0\sim p_{G_0}, t\sim p_{data}}[\log(1 - D(G(s_0, \hat{c}), \varphi_t))] + \\ & \lambda D_{KL}(\mathcal{N}(\mu(\varphi_t), \Sigma(\varphi_t))||\mathcal{N}(0, I_n)),\end{aligned} \tag{3.5}$$

where I is the real image in the Stage-II.

3.2 Attentional generative adversarial network

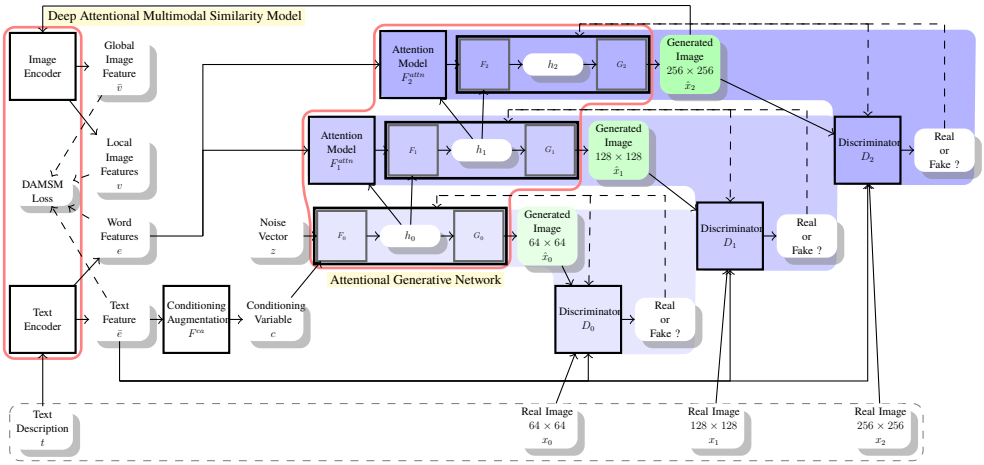


Figure 3.2: The architecture of AttnGAN

The architecture of AttnGAN is depicted in Figure 3.2. It has two novel components: Deep Attentional Multimodal Similarity Model (DAMSM) and Attentional Generative Network. Both of them integrate the attention mechanism (i.e., dot-product attention). Instead of generating text embeddings with

the help of an existing text encoder, DAMSM is trained ahead of the GANs in order to produce both text features and word features, acting as an internal text encoder aiming at aligning images and corresponding descriptions at the word level. Meanwhile, a new loss function based on DAMSM is proposed and helps to guide the training of the generators in the GANs. On the other hand, the word features are utilized in the generators for the first time, synthesizing fine-grained sub-regions of the images by paying attention to the relevant words.

3.2.1 Deep attentional multimodal similarity model

DAMSM is designed to measure the similarity between the whole text t and the whole image $\tilde{x}$ ($\tilde{x}$ is either x_2 or $\hat{x}_2$ when the target size of output image is 256×256) at the word level. One text encoder and one image encoder constitute this model. It is pretrained using the ground-truth text-image (t, x_2) pairs. While training the GANs, it encodes the text description t and also extracts the image features from the generated images $\hat{x}_2$ to compute the proposed DAMSM loss added in the objectives of the generators.

Text encoder

The text encoder has three sequential parts: an embedding layer, a drop-out layer, and a bi-directional LSTM (bi-LSTM). Their weights are updated during the pretraining of DAMSM. Once completed, the fixed text encoder will be used in the training of the GANs.

The original description can be a sentence or a paragraph of sentences. It will be cleaned and prepared. To begin with, the punctuation marks in it will be removed so that only alphanumeric characters (words and numbers) are preserved which are then unified into lowercase and tokenized. Thus, each description will become a sequence of tokens. All unique tokens in the corpus constitute a dictionary and each token has a unique index in it. So, each text will become a sequence of indices that will be mapped into an embedding matrix by retrieving the corresponding word embeddings from the embedding layer with the indices. The matrix will be passed through the drop-out layer and ultimately fed into the bi-LSTM to extract final word features $e \in \mathbf{R}^{D \times T}$ and sentence feature $\bar{e} \in \mathbf{R}^{D}$, where D is the dimension of the word feature and T is the number of tokens(words). Each word feature e_i, i.e., each column in e is constructed as the concatenation of the two hidden states in the bi-LSTM. While $\bar{e}$ is the text embedding by concatenating the last two hidden states.

Image encoder

The image encoder is adapted from the Inception V3 model (primary branch) without the final layers for classification. Two additional layers are applied at two certain positions of that model to obtain local features and global feature of an image. The weights in Inception V3 are loaded from the pretrained model on ImageNet. They are fixed and only the weights in the two newly added layers are learned when pretraining DAMSM. Again, once the training finishes, the fixed image encoder will be used in the training of the GANs.

Before inputting images into the image encoder, they have to be rescaled to have size $3 \times 299 \times 299$, which is the default input image size for Inception V3. When it comes to the intermediate layer $Mixed\_6e$, the size of the output in origin should be $768 \times 17 \times 17$. Now, a 1×1 convolutional layer is applied here to convert the image features into a common semantic space of the text features. Therefore, the size of the output becomes $D \times 17 \times 17$, where D is the dimension of the word feature. After reshaping, it yields the visual feature matrix $v \in \mathbf{R}^{D \times 289}$, where 289 is considered as the number of sub-regions in the image. So, each column of v represents the local feature for a sub-region. Similarly, the original size of the output of the last average pooling layer is 2048×1 and this output is passed through a newly-added fully-connected layer and becomes the global feature $\bar{v} \in \mathbf{R}^D$ of the whole image.

The DAMSM loss

The DAMSM loss is the one of two places in AttnGAN that integrates the attention mechanism. It is the loss function used to pretrain DAMSM. Besides, it is added into the objectives of the generators, which will be presented in the next subsection.

First of all, a similarity matrix for all pairs of word feature e_i and local visual feature v_j in a text-image pair is calculated as defined below:

$$s = e^T v, \tag{3.6}$$

where $s \in \mathbf{R}^{T \times 289}$ and $s_{i,j}$ is the dot-product similarity between the i^{th} word in the description and the j^{th} sub-region in the image, which is exactly dot-product attention. It is normalized as follows:

$$\bar{s}_{i,j} = \frac{\exp(s_{i,j})}{\sum_{k=0}^{T-1} \exp(s_{k,j})}. \tag{3.7}$$

Then, an attention model is built to compute the region-context vector r_i for the i^{th} word. It is the weighted sum of the local visual features related to

the i^{th} word, that is,

$$r_i = \sum_{j=0}^{288} \alpha_{i,j} v_j, \quad \text{where} \quad \alpha_{i,j} = \frac{\exp(\gamma_1 \bar{s}_{i,j})}{\sum_{k=0}^{288} \exp(\gamma_1 \bar{s}_{i,k})}. \tag{3.8}$$

Note that γ_1 is a hyper-parameter that represents how much attention should be paid to those local visual features.

Next, the relevance between the region-context vector r_i and the word feature e_i is measured using cosine similarity, namely,

$$R(r_i, e_i) = \frac{r_i^T e_i}{\parallel r_i \parallel \parallel e_i \parallel}. \tag{3.9}$$

The matching score between the whole description t and the whole image $\tilde{x}$ is defined as follows:

$$S(t, \tilde{x}) = \log\Big(\sum_{i=1}^{T-1} \exp(\gamma_2 R(r_i, e_i)) \Big)^{\frac{1}{\gamma_2}}, \tag{3.10}$$

where γ_2 is a hyper-parameter representing how much to magnify the importance of the most relevant pair of region-context vector and word feature.

Finally, given a group of text-image pairs $\{(T_i, \tilde{X}_i)\}_{i=1}^{M}$, the posterior probability of description T_i being matching with image $\tilde{X}_i$ is calculated by

$$P(T_i|\tilde{X}_i) = \frac{\exp(\gamma_3 S(T_i, \tilde{X}_i))}{\sum_{j=1}^{M} \exp(\gamma_3 S(T_i, \tilde{X}_i))}, \tag{3.11}$$

where γ_3 is also a hyper-parameter determined by experiments. Note that only T_i matches with $\tilde{X}_i$, while the other $M-1$ description candidates are mismatching ones. Symmetrically, we can also calculate $P(\tilde{X}_i|T_i)$ that is the posterior probability of image $\tilde{X}_i$ being matching with description T_i. Thus, the loss function at the word level $\mathcal{L}^w$ is defined as

$$\mathcal{L}^w = -\sum_{i=1}^{M} \log P(T_i|\tilde{X}_i) - \sum_{i=1}^{M} \log P(\tilde{X}_i|T_i). \tag{3.12}$$

On the other hand, when taking the sentence feature $\bar{e}$ and global visual feature $\bar{v}$ into account, the matching score in Equation 3.10 can be redefined as

$$S(t, \tilde{x}) = \frac{\bar{v}^T \bar{e}}{\parallel \bar{v} \parallel \parallel \bar{e} \parallel}. \tag{3.13}$$

After replacing it in Equation 3.11, we can get the loss function at the sentence level $\mathcal{L}^s$. Eventually, the DAMSM loss is the sum of the two losses, i.e.,

$$\mathcal{L}_{DAMSM} = \mathcal{L}^w + \mathcal{L}^s. \tag{3.14}$$

3.2.2 Attentional generative network

The architecture

Taking away DAMSM, the rest of AttnGAN is a pure conditional GAN. As can be seen in Figure 3.2, the attentional generative network has a stack of generators that synthesize images from small to large scales. The generator at the i^{th} stage is composed of a hidden net F_i that generates hidden image features h_i and a network G_i that maps h_i to the image $\hat{x}_i$. However, this fashion of stack differs from that of StackGAN. Namely, each generator stacks on the hidden state of previous stage rather than the generated image. The corresponding discriminator D_i takes positive pairs and negative pairs as input like that in StackGAN, mentioned in subsection 3.1.2. Furthermore, the CA technique is also involved in this text-to-image algorithm, denoted as F^{ca}. But it is merely applied before the first generator, converting the sentence feature $\bar{e}$ to the conditioning variable c. In terms of the unique attentional generative network, it can be expressed more precisely as follows:

$$
\begin{aligned}
h_i &= \begin{cases} F_i(z, c), & i = 0; \\ F_i(h_{i-1}, F_i^{attn}(h_{i-1}, e)), & i = 1, ...m-1, \end{cases} \\
\hat{x}_i &= G_i(h_i),
\end{aligned} \tag{3.15}
$$

where z is a standard Gaussian noise vector.

The attention models F^{attn} is the other place integrating the attention mechanism in AttnGAN. The inputs of F_i^{attn} are the word features $e \in \mathbf{R}^{D \times T}$ and hidden image features h_{i-1}. The shape of h varies at different stages. For the convenience of description, it is denoted as $h \in \mathbf{R}^{\hat{D} \times N}$, where $\hat{D}$ is the dimension of a single hidden feature and N is viewed as the number of sub-regions in the image to be generated, i.e., each column in h is the feature of a specific sub-region. The approach to get a word-context vector w_j for the j^{th} sub-region is quite similar to that for computing a region-context vector r_i for the i^{th} word when deriving the DAMSM loss. The word features firstly need to be converted into the common semantic space of the hidden image features. This is accomplished by passing them through a 1×1 convolutional layer. Then, they become $e' \in \mathbf{R}^{\hat{D} \times T}$. We have

$$
s' = h^T e', \tag{3.16}
$$

where s' is the similarity matrix for all pairs of word features and hidden image features in a text-image training pair. The word-context vector w_j is computed

as follows:

$$w_j = \sum_{i=0}^{T-1} \beta_{j,i} e'_i, \quad \text{where} \quad \beta_{j,i} = \frac{\exp(s'_{j,i})}{\sum_{k=0}^{T-1} \exp(s'_{j,k})}. \tag{3.17}$$

That means the word-context vector w_j is the weighted sum of the word features related to the j^{th} sub-region. Thus the word-context matrix for hidden image features h is $F^{attn}(h, e) = (w_0, w_1, ..., w_{N-1})$.

The hidden net F and network G have different layer components at different stages, such as fully-connected layers, upsampling blocks, joining layers, residual blocks and convolutional layers. There are also some drop-out layers, batch normalization layers, leaky-relu layers, etc., after each component.

The objectives

The loss function for the generator G_i at the i^{th} stage of the attentional generative network is given by

$$\mathcal{L}_{G_i} = -\frac{1}{2}\mathbf{E}_{\hat{x}_i \sim p_{G_i}}[\log D_i(\hat{x}_i)] - \frac{1}{2}\mathbf{E}_{\hat{x}_i \sim p_{G_i}}[\log D_i(\hat{x}_i, \bar{e})], \tag{3.18}$$

where $\hat{x}_i$ from the model distribution p_{G_i} at the i^{th} scale has been defined in Equation 3.15. This function jointly approximates both conditional (with $\bar{e}$) and unconditional (without $\bar{e}$) distributions. Specifically, the unconditional loss determines if the generated image is real or fake, while the conditional loss reflects if the generated image matches with the sentence feature. $D_i(\hat{x}_i)$ is obtained by directly passing $\hat{x}_i$ through D_i to get the scalar output. It is a little different when computing $D_i(\hat{x}_i, \bar{e})$, because $\hat{x}_i$ and $\bar{e}$ need to be constructed as a single tensor through an additional network apart from D_i.

The final loss function of the attentional generative network is composed of three parts: the sum of generators' losses, the DAMSM loss for the last scale and the CA-based regularization term similar to Equation 3.1, i.e.,

$$\mathcal{L} = \mathcal{L}_G + \lambda \mathcal{L}_{DAMSM} + D_{KL}(\mathcal{N}(\mu(\bar{e}), \Sigma(\bar{e}))||\mathcal{N}(0, I_n)), \tag{3.19}$$

where $\mathcal{L}_G = \sum_{i=0}^{m-1} \mathcal{L}_{G_i}$, λ is a hyper-parameter balancing the terms.

Alternatively to the training of the attentional generative network that is updated as a whole by minimizing $\mathcal{L}$ in Equation 3.19, each discriminator is trained in parallel to the other discriminators since they are disjointed in the architecture. It is done by maximizing $\mathcal{L}_{D_i}$ defined as follows:

$$\begin{aligned} \mathcal{L}_{D_i} = &-\frac{1}{2}\mathbf{E}_{x_i \sim p_{data_i}}[\log D_i(x_i)] - \frac{1}{2}\mathbf{E}_{\hat{x}_i \sim p_{G_i}}[\log(1 - D_i(\hat{x}_i))] \\ &-\frac{1}{2}\mathbf{E}_{x_i \sim p_{data_i}}[\log D_i(x_i, \bar{e})] - \frac{1}{2}\mathbf{E}_{\hat{x}_i \sim p_{G_i}}[\log(1 - D_i(\hat{x}_i, \bar{e}))], \end{aligned} \tag{3.20}$$

where x_i is from the training data distribution p_{data_i} at the i^{th} scale. The first two terms are unconditional losses and the last two terms are conditional losses.

3.3 Datasets and settings

3.3.1 Datasets

Two fashion datasets are utilized in our work. One is the Fashion Synthesis benchmark in DeepFashion [34]. The other is FashionGen [35] (256×256 version). Their statistics referenced in our experiments are displayed in Table 3.1. From top to bottom, it shows information of basics, image set, and text set.

Attributes \ Datasets	FashionGen	Fashion Synthesis
Number of samples	train: 260490 validation: 32528 test: 32528	train: 70000 test: 8979
Categories	48	19
Resolution	256×256	128×128
Poses	multiple	multiple
Background	white	varied
Description	detailed	brief
Vocabulary	6872	501

Table 3.1: Statistics of the datasets

As far as data split is concerned, we only have access to the training and validation set of FashionGen. Its test set will not be publicly released but only provided in the FashionGen Challenge. As for Fashion Synthesis, it does not provide validation set. Technically, we should use cross-validation on the training set to determine the hyper-paramters. However, we treat the test set as the validation set for simplicity.

In terms of the categories, in addition to the 48 main categories, FashionGen also provides a fine-grained division, i.e., 121 subcategories. In our experiments, however, we only consider the main categories. Moreover, FashionGen includes more than clothing. It also has shoes, bags, accessories, etc.

For each dataset, their distributions of samples per category are shown in Figure 3.3 and Figure 3.4 separately.

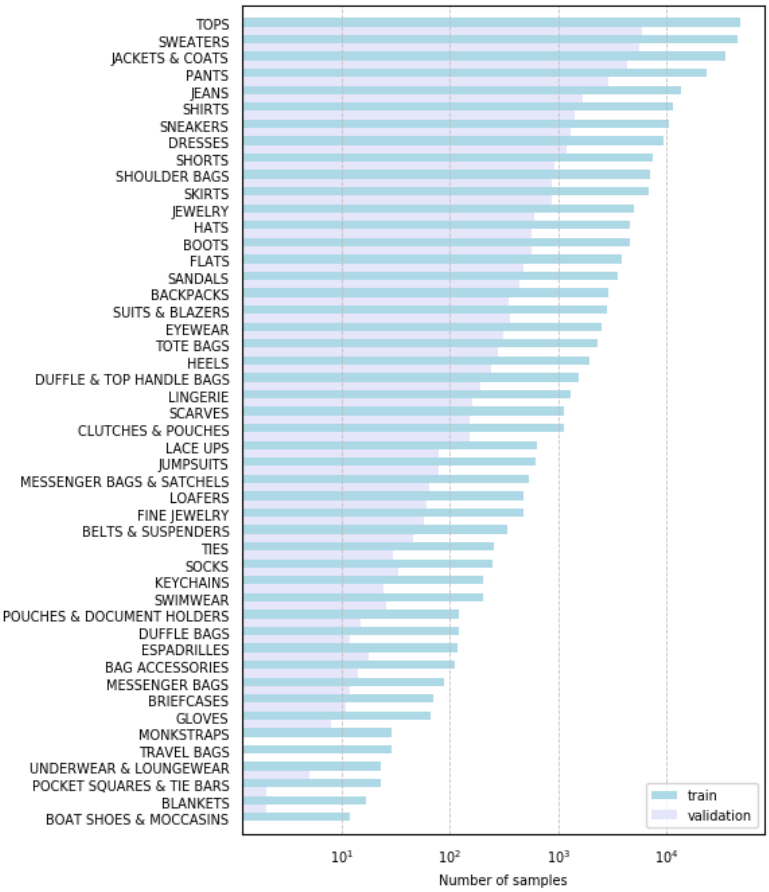


Figure 3.3: The distribution of samples per category in FashionGen

Regarding the image set, each fashion item in FashionGen is photographed from a few arbitrary angles, e.g., half body, whole body, profile, etc, resulting in a group of samples where different images are paired with the same description. In Fashion Synthesis, the images are cropped from the original images having varied backgrounds in another benchmark of DeepFashion.

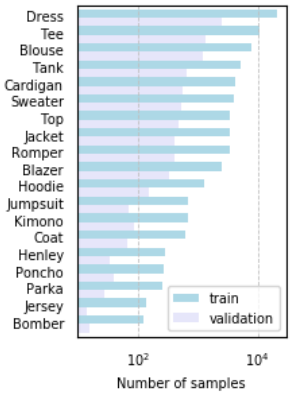


Figure 3.4: The distribution of samples per category in Fashion Synthesis

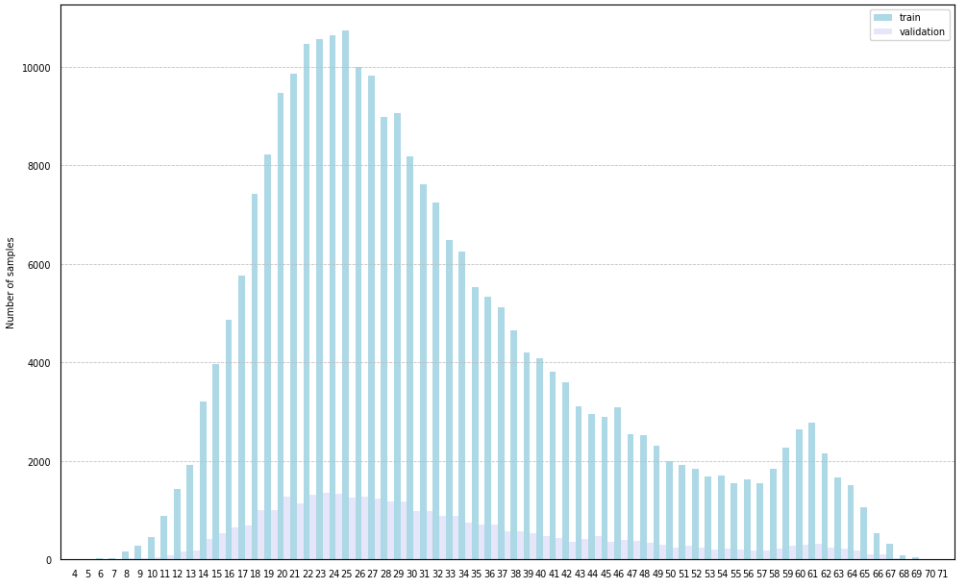


Figure 3.5: The distribution of the description lengths in words in FashionGen

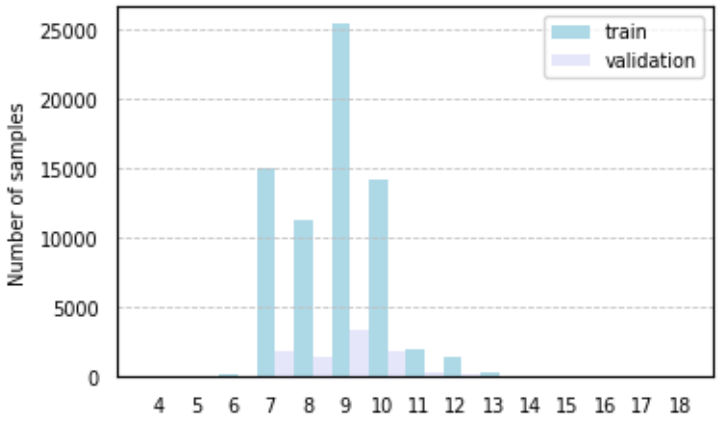


Figure 3.6: The distribution of the description lengths in words in Fashion Synthesis

Regarding the text set, FashionGen provides professional descriptions of design. Since the focus of description is on the design including texture, shape, etc., it does not contain information of gender, though this is available in another metadata. By contrast, each description in Fashion Synthesis has gender information, but it does not say much about the design and is more like a general impression on the items. Using a specific tokenizer, we can count the unique words(tokens) in the whole text set, it also shows a large gap between them with respect to the complexity of vocabulary. For each of them, the distributions of the description lengths in words are shown in Figure 3.5 and Figure 3.6, respectively.

3.3.2 Experiment settings

We use the source code of the two papers, based on which we extended and modified some parts to fulfill our desired functionalities. The code of this project is available from this link: `https://github.com/Zhengro/Text-to-image-Synthesis-for-Fashion-Design`. The two text-to-image synthesis algorithms are implemented using Python 2.7 with PyTorch. The calculation of FID is implemented using Python 3 with Tensorflow. The models are trained on a GPU cluster, supported by division of Robotics, Perception and Learning at KTH Royal Institute of Technology. Three versions of Nvidia GPUs are used, GeForce GTX 1080 Ti, Titan Xp, and GeForce RTX 2080 Ti. The corresponding GPU memories are 11178 MiB, 12196 MiB, and 11019 MiB.

The experiments have two branches corresponding with two text-to-image synthesis algorithms. Each of them is applied on the two datasets introduced in the last subsection. All the cases are AttnGAN on FashionGen, AttnGAN on Fashion Synthesis, StackGAN on FashionGen and StackGAN on Fashion Synthesis.

For AttnGAN, we left the architecture of GAN part unchanged and tried a few variants of DAMSM, attempting to maximize the performance of the text encoder. In this case, bi-LSTM and bi-GRU are compared with respect to the RNN component in DAMSM. Also, the pretrained GloVe word embedding is compared with the random initialization of the embedding layer. The hyper-parameters we tuned in DAMSM are learning rate, number of epochs, batch size, maximum number of tokens for a description, dimension of the embeddings, γ_1 in Equation 3.8, γ_2 in Equation 3.10, and γ_3 in Equation 3.11. The other two significant hyper-parameters in AttnGAN are the learning rate for training the network and λ in Equation 3.19.

For StackGAN, we used the produced text embeddings from AttnGAN as the input of StackGAN so that we can compare the performance of the two network architectures. Likewise, we do not change the architecture of StackGAN in the original paper when generating images of size 256×256 on FashionGen. Nevertheless, we modified several layers of generator and discriminator in the Stage-II GAN so that it can produce 128×128 images on Fashion Synthesis. The hyper-parameters in this algorithm are fewer than that in AttnGAN, which are learning rates of generator and discriminators, batch size for each stage of StackGAN and number of epochs.

3.4 Evaluation

3.4.1 Inception Score

Inception Score (IS) is a widely used metric when evaluating the performance of GANs. The computation is based on a pretrained Inception V3 model that is originally proposed for image classification, where for each input image $\boldsymbol{x}$ it outputs a list of probabilities $p(y|\boldsymbol{x}) \in [0, 1]^N$. Here, y is the set of class labels and N is the number of labels in the dataset. Mathematically, IS is given by

$$\exp\big(\mathbf{E}_{\boldsymbol{x}} D_{KL}(p(y|\boldsymbol{x}) \parallel p(y))\big), \tag{3.21}$$

where $\boldsymbol{x}$ is a generated image from the GAN, i.e., $\boldsymbol{x} = G(z)$, $p(y) = \int p(y|\boldsymbol{x} = G(z))dz$ is the marginal label distribution. The conditional label distribution

$p(y|\boldsymbol{x})$ should be narrow, meaning that the image contains a distinct object and thus is sharp. On the other hand, the marginal label distribution $p(y)$ should be uniform, indicating that the images are from all the classes and have variety. If both of them are satisfied, the KL divergence between the two distributions will be large. Hence, larger IS means better performance the GAN can have.

However, IS has some inherent limitations and also problems caused by wrong usages as pointed out in [36]. For example, IS is sensitive to different implementations of Inception model or small changes in weights of Inception model that do not affect the final classification accuracy when pretraining it. IS should be used only when the Inception model has been trained on the same dataset that is used to train the generative model. In our work, we fail to train a Inception model on FashionGen for image classification. Thus, the two algorithms are only evaluated with FID to be introduced in the next setion.

3.4.2 Fréchet Inception Distance

Considering that one of the drawbacks of IS is that the statistics of real samples are not used, Fréchet Inception Distance (FID) is proposed to improve it. Its computation is also based on a Inception model. However, it utilizes the statistics of an intermediate layer rather than the output, which is modeled by a multivariate Gasussian distribution $\mathcal{N}(\mu, \Sigma)$ (μ is mean vector, Σ is covariance matrix). For real samples, the feature distribution is denoted as $\mathcal{N}(\mu_x, \Sigma_x)$. Correspondingly, the feature distribution of generated samples is $\mathcal{N}(\mu_{\hat{x}}, \Sigma_{\hat{x}})$. FID is the Fréchet distance between them given by

$$\| \mu_x - \mu_{\hat{x}} \|_2^2 + \mathrm{Tr}(\Sigma_x + \Sigma_{\hat{x}} - 2(\Sigma_x \Sigma_{\hat{x}})^{\frac{1}{2}}), \tag{3.22}$$

where Tr sums up all the diagonal elements. It has been shown that FID is more consistent with the noise level than IS and it is better than IS when measuring the image diversity.

Chapter 4

Results

In this chapter, we follow up the methods in Chapter 3 and present the results in this project. More plainly stated, both AttnGAN and StackGAN share the same text encoder, which is learned during the training of DAMSM in AttnGAN. Hence, we are first going to deliberate on the results of DAMSM, including findings when tuning some parameters or using different components. After that, we demonstrate a series of images in the context of AttnGAN, such as attention areas and synthesized images. Lastly, images synthesized by StackGAN are displayed. All the numerical results can be found in Table 4.1.

4.1 The results on the deep attentional multimodal similarity model

When tweaking the hyper-parameters in DAMSM, we rely on the DAMSM loss. As shown in Equation 3.14, it consists of loss at the word level (w) and loss at the sentence level (s). Additionally, we plot both learning curves on the training and validation/test set. Hence, for each parameter, there are 4 curves in the plot. (Note that, when using FashionGen to pretrain DAMSM, the losses are calculated on a fixed subset of its validation set that is randomly selected to save time, which does not affect much compared to using the whole validation set.) One may notice that the losses are large in some figures below, which is quite related to batch size to be explained in setion 4.1.2.

4.1.1 Components of the text encoder

Random initialization and pretrained GloVe embeddings

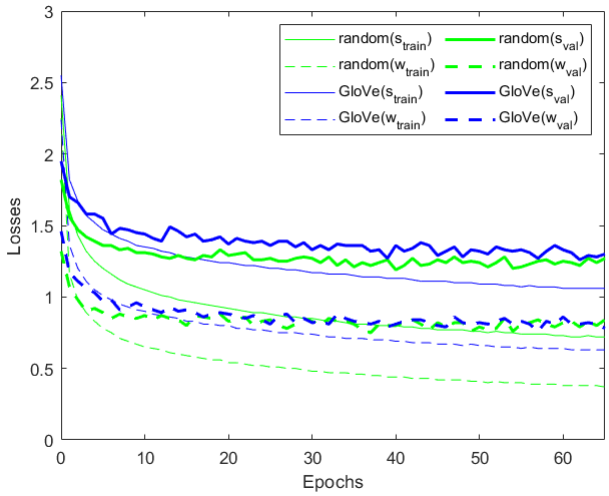


Figure 4.1: The comparison between random and GloVe initialization of the embedding layer in DAMSM on the training set and partial validation set of FashionGen (w: loss at the word level; s: loss at the sentence level.)

One component of the text encoder in DAMSM is the embedding layer storing learned word embeddings. Here, we attempt to apply the pretrained 300-dimension GloVe word embeddings available from `https://nlp.stanford.edu/projects/glove/`. The corpus for pretraining is Wikipedia 2014, which has 400 thousand vocabulary and produce 6 billion tokens. Take FashionGen as an example, 5927 out of 6872 tokens of FashionGen are found. The embeddings of the left 945 tokens are initialized randomly. As opposed to this, another embedding layer with the same size is completely initialized with random vectors. Their respective learning curves are shown in Figure 4.1.

It indicates that the pre-trained embeddings cannot represent the training data very well. Though training the embeddings from scratch takes much time, it is still preferable because it learns from the specific context of our datasets. In this case, random initialization of the embedding layer is applied in DAMSM in all our experiments.

LSTM and GRU

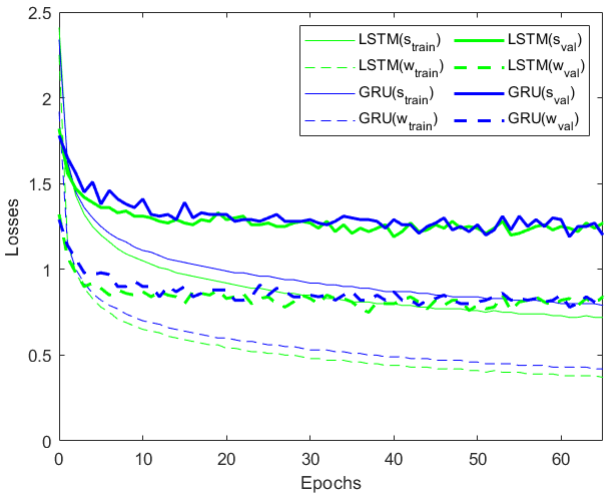


Figure 4.2: The comparison between bi-LSTM and bi-GRU in DAMSM on the training set and partial validation set of FashionGen

Another component of the text encoder is a RNN. It is chosen to be a bi-LSTM in AttnGAN. However, bi-GRU can be another good option considering its wide applications, so we try both of them on FashionGen. they are both one-layer and their hidden states are of same size. In our use case, bi-LSTM works slightly better than bi-GRU, as can be seen in Figure 4.2. Also, bi-GRU does not show obvious advantage on training speed. So, we adopt bi-LSTM in DAMSM in all our experiments.

4.1.2 Hyper-parameters in DAMSM

We carefully tune each hyper-parameter in DAMSM while keeping other ones unchanged. Among all of the factors, it is worth mentioning some of them, i.e., max number of tokens forming the description, dimension of the text embeddings produced by the bi-LSTM, the balancing parameters γ_1, γ_2, γ_3, and batch size.

Max number of tokens

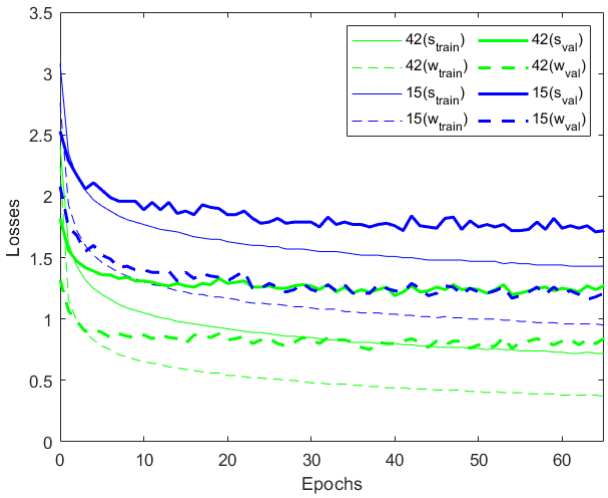


Figure 4.3: The comparison of 42 tokens and 15 tokens on the training set and partial validation set of FashionGen

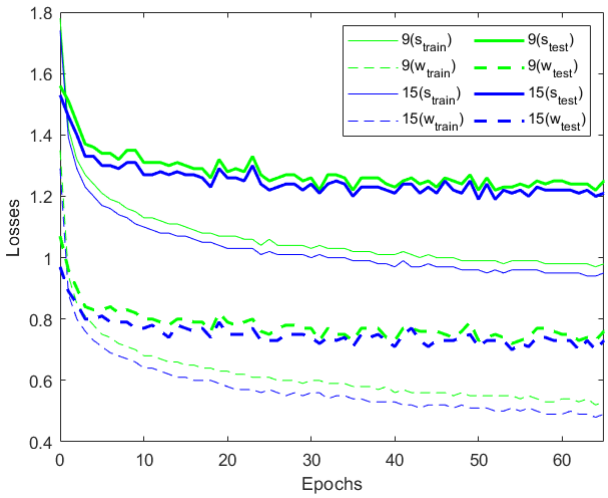


Figure 4.4: The comparison of 9 tokens and 15 tokens on the training set and test set of Fashion Synthesis

The lengths of descriptions varies, so do that of the sequences of tokens. However, when fed into the bi-LSTM in DAMSM in batches, each sequence must be of same length L, i.e., max number of tokens for each description within

each batch. In this case, zero-padding (a predefined token '<end>' is used for padding) is necessary for short sequences and sampling is necessary for long sequences. For COCO and CUB, L is set to be 15 and 18 respectively in the experiments of AttnGAN. However, since FashionGen tends to have very long descriptions, sticking to common values of L may be inappropriate. So, we compare the losses when $L = 42$ and $L = 15$, demonstrated in Figure 4.3. We try $L = 42$, because sequences with lengths less than or equal to 42 account for around 80% of all samples in FashionGen. For Fashion Synthesis, the 80% point is $L = 9$ and we also try $L = 15$, as can be seen in Figure 4.4.

It reveals that for both datasets, larger L comes with smaller losses. Thus, we use $L = 42$ for FashionGen and $L = 15$ for Fashion Synthesis.

Dimension of the text embeddings

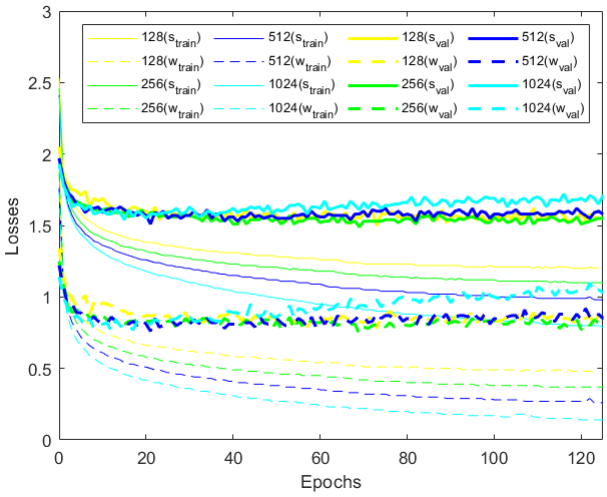


Figure 4.5: The comparison of different dimensions of text embeddings on the training set and partial validation set of FashionGen

The dimension of the text embeddings is also the size of two concatenated hidden states in the bi-LSTM. We experiment on several cases, i.e., 128, 256, 512, and 1024 on FashionGen. The results are depicted in Figure 4.5.

It can be seen that on the training set larger dimension corresponds to smaller losses. However, large-dimension embeddings are prone to overfitting on the validation set due to too much flexibility of the model. In this case, we should use 256 as the dimension of text embeddigns.

Balancing parameters

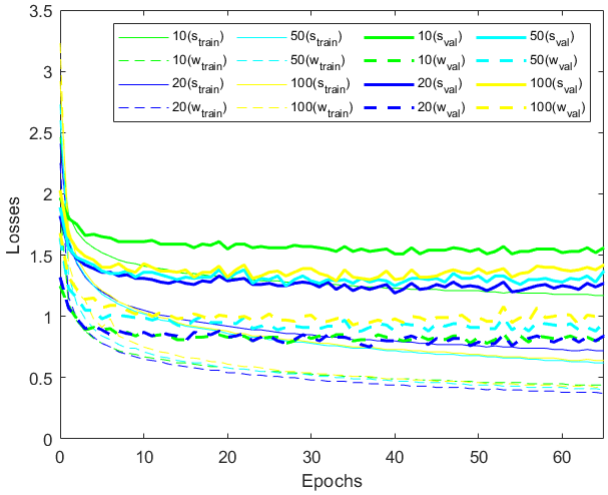


Figure 4.6: The comparison of different γ_3 in DAMSM on the training set and partial validation set of FashionGen

There are three special hyper-parameters in DAMSM, namely, γ_1, γ_2, and γ_3. When tuning these balancing parameters, we find that γ_1 and γ_2 have no obvious impact on the losses, while a good γ_3 can lower the loss at the sentence level on the validation set, as illustrated by Figure 4.6.

For FashionGen, we specify $\gamma_1 = 4$, $\gamma_2 = 5$ and $\gamma_3 = 20$. For Fashion Synthesis, we use $\gamma_1 = 5$, $\gamma_2 = 5$ and $\gamma_3 = 10$.

Batch size

It is a common practice to use a large batch size in lots of deep learning based applications, since it is usually of benefit for rapid convergence of the model parameters and better performance. Following this experience, we set the batch size as large as possible when pretraining DAMSM on FashionGen, according to the available memory of the GPU. The DAMSM losses are still large after we finish the comparison of the aforementioned components or different values of a specific parameter. Though it is weird, the visualization of the text embeddings seems to be able to learn different features of categories in the validation set of FashionGen, as shown in Figure 4.9 in the next subsection. We have to continue to train the attentional generative network on FashionGen with this pretrained DAMSM, since we cannot find a way to further decrease the loss.

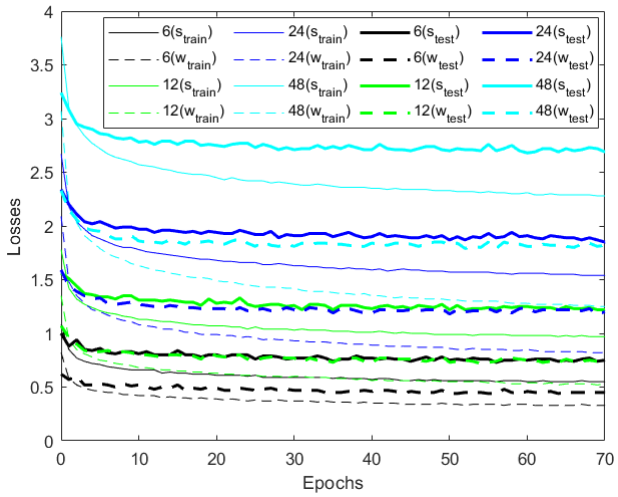


Figure 4.7: The comparison of different batch sizes on the training set and test set of Fashion Synthesis

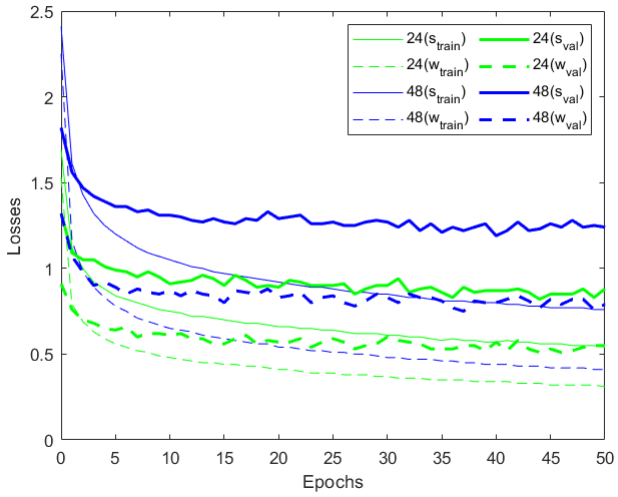


Figure 4.8: The comparison of different batch sizes on the training set and partial validation set of FashionGen

Later in our experiments, we find it is actually inappropriate to use large batch size to pretrain DAMSM, when we are in the period of pretraining DAMSM on Fashion Synthesis. To demonstrate this, we experiment on different batch sizes using Fashion Synthesis and plot the learning curves demonstrated in Figure 4.7. We go back to train DAMSM on FashionGen with half of batch

size (i.e., 24), as shown in Figure 4.8. Due to the limited time, we end up with the previously trained models (DAMSM, AttnGAN and StackGAN) on FashionGen. We only adopt a smaller batch size on Fashion Synthesis for pretraining DAMSM and its generated embeddings are further used for the two models on Fashion Synthesis. Although small batch size can lower the losses significantly, it involves a trade-off between losses and training time. It is acceptable to use a small batch size to pretrain DAMSM on Fashion Synthesis. However, it is already very slow to pretrain DAMSM on FashionGen with a relatively large batch size. In this case, it is not practical to use a small batch size.

Visualization of the text embeddings

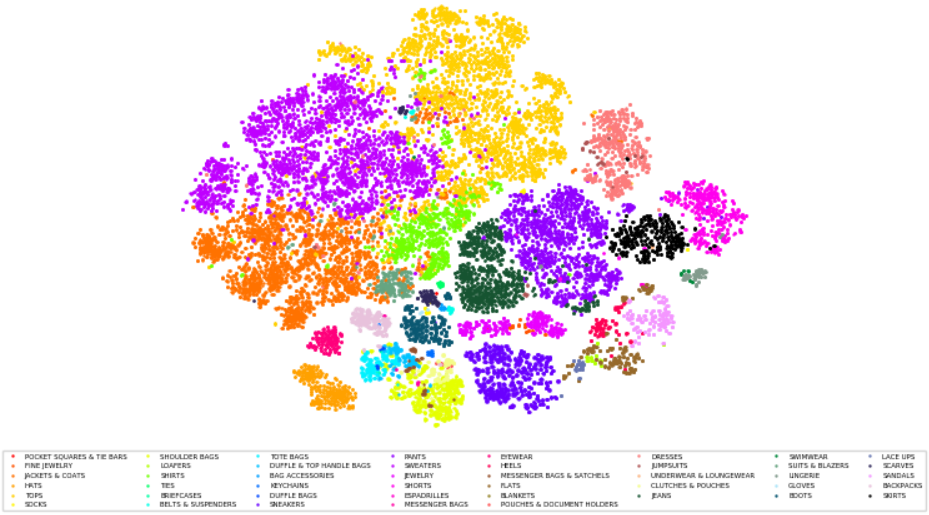


Figure 4.9: The visualization of the text embeddings on the validation set of FashionGen

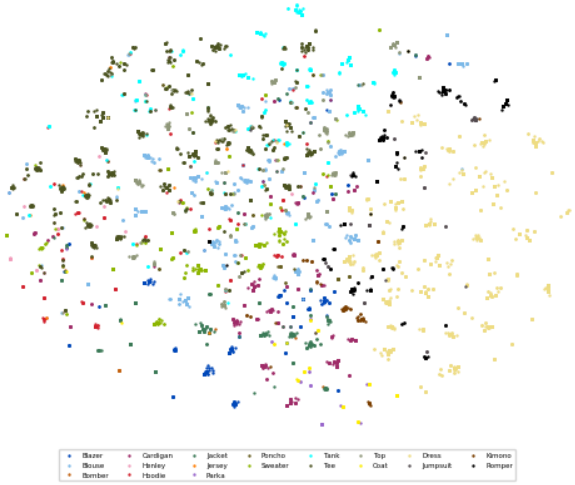


Figure 4.10: The visualization of the text embeddings on the test set of Fashion Synthesis

There are a variety of approaches to evaluate the quality of word embeddings, either extrinsic evaluation or intrinsic evaluation [37], which are beyond our research. However, we would like to employ t-SNE [38] to implicitly evaluate the text embeddings, which is a popular way to visualize high-dimensional data. It is a non-linear dimensionality reduction algorithm and thus capable of capturing not only the local structure, but also the global structure of data. We use its implementation in Scikit-Learn with the text embeddings. From Figure 4.9, we can see that the categories in the validation set of FashionGen can be reflected by the clusters of text embeddings, though some data points are in disorder. In Figure 4.10, the data points are more sparse and more overlapped, which is possibly due to the less obvious features of categories indicated by the descriptions of Fashion Synthesis.

4.2 The results on the attentional generative network

With the pretrained DAMSM on FashionGen or Fashion Synthesis, we can train the attentional generative network to generate images of different sizes, i.e., 256×256 for FashionGen, 128×128 for Fashion Synthesis. We will first present intermediate results when tweaking hyper-parameters in this model on each dataset and show the final quantitative results. Then, some synthesized

samples on the validation set of FashionGen and test set of Fashion Synthesis are provided for qualitative evaluation.

4.2.1 Hyper-parameters in the attentional generative network

On the premise of not changing the compositions of network, the attentional generative network involves the following hyper-parameters: learning rates of generator and discriminator, batch size, number of epochs and λ in Equation 3.19. In this part, we use FID as introduced in subsection 3.4.2 to evaluate the quality of generated images and determine each parameter above. Instead of exhibiting the details of them all, we elaborate on the special parameter λ.

When generating 128×128 images, F_2^{attn}, F_2, G_2 and D_2 in Figure 3.2 are skipped. The left parts are the model for tweaking those parameters on Fashion Synthesis and generating images. We compare three cases: $\lambda = 1$, $\lambda = 5$, and $\lambda = 10$ after determining other parameters. The FID curves are plotted in Figure 4.11. As for FashionGen, since it is really slow to train all three stages, we cannot afford the time to tune each hyper-parameter by using the complete model. We decide to follow the method in their experiments [8], that is, we construct the first-two-stage model (i.e., F_2^{attn}, F_2, G_2 and D_2 in Figure 3.2 are skipped) for tuning parameters and then directly use those values in the complete model for generating 256×256 images. We also compare the above cases to find the best value of λ on FashionGen, as shown in Figure 4.12.

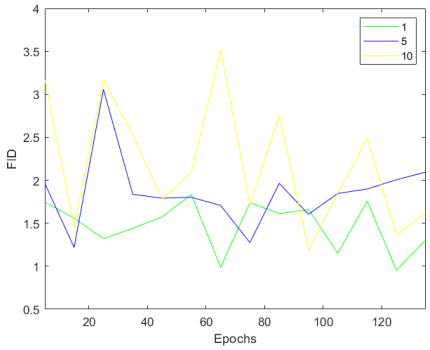


Figure 4.11: The comparison of different λ on the test set of Fashion Synthesis

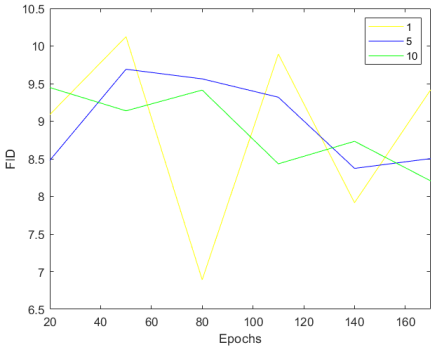


Figure 4.12: The comparison of different λ on the validation set of FashionGen

In both figures, the curves oscillate, which is a typical behavior when we tune each parameter in the attentional generative model. It indicates the training of GANs is not stable. According to the current situations, we choose $\lambda = 1$ for Fashion Synthesis and it has the minimum FID, 0.953 at epoch 125, reported in Table 4.1. We choose $\lambda = 10$ for FashionGen, though it does not get the minimum value of FID, it shows relatively steady decrease of FID as the number of epochs increases. With all the tuned parameters and the complete model, we obtain the final numerical result of AttnGAN on the validation set of FashionGen, i.e., FID=1.511, recorded in Table 4.1. It can be noticed that the FID values in Figure 4.12 is much higher than the final reported value. It is reasonable because they are computed using the intermediate 128×128 images and the statistics of 256×256 images of the validation set of FashionGen. Indeed, this gap reflects the quality improvement of the images from the second stage to the final stage.

Datasets / Algorithms	FashionGen (256×256)	Fashion Synthesis (128×128)
AttnGAN	1.511	0.953
StackGAN	2.266	0.434

Table 4.1: The Fréchet Inception Distances of difference cases on both the validation set of FashionGen (256×256) and the test set of FashionSynthesis (128×128)

4.2.2 Synthesized images

In [8], by manipulating the weight matrix $\beta^{N\times T}$ (N is the number of sub-regions in the generated image, T is the number of tokens) in Equation 3.17, the attention learned by the attention model $\mathbf{F}_i^{attn}$ in Equation 3.15 can be visualized. Specifically, each word has N corresponding weights, ($\beta_{0,i}$, $\beta_{1,i}$, ... $\beta_{N-1,i}$). They are transformed from $N \times 1$ to $\sqrt{N} \times \sqrt{N} \times 3$, becoming a map. A threshold is set to turn the weights corresponding to the less-relevant sub-regions into zero. Then, the preserved weights are added up as a score for that word and the map is upsampled with Gaussian filters to have the same size as the generated image. Finally, the generated image and the map are stacked, showing the effect of attention. We only visualize the maps for the words having top-5 highest scores. In addition, multiple synthesized images are displayed given the same description to examine the diversity.

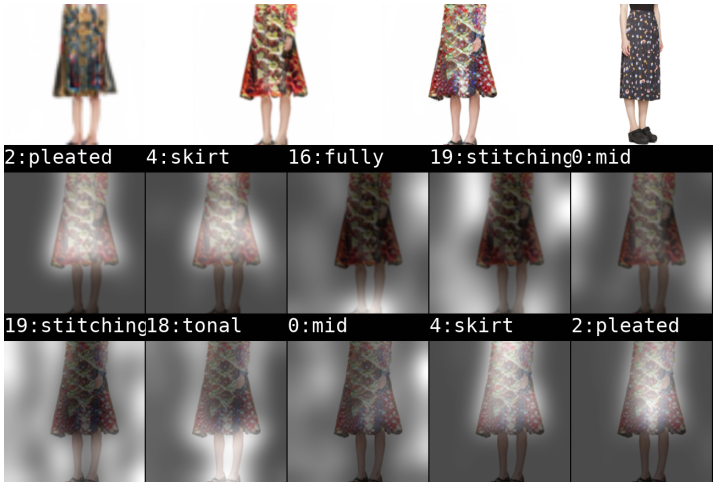
FashionGen

As displayed in Figure 4.13, for each map example from FashionGen, the first row gives the three generated images from the attentional generative network, followed by the corresponding real image in the validation set. The first two generated images are bilinearly upsampled to be of the same size as the third one for better visualization. The second row and last row illustrate top-5 words attended by $\mathbf{F}_1^{attn}$ and $\mathbf{F}_2^{attn}$, respectively. The original description is also provided in the caption of each subfigure. More examples can be found in Figure A.1 in the Appendix.

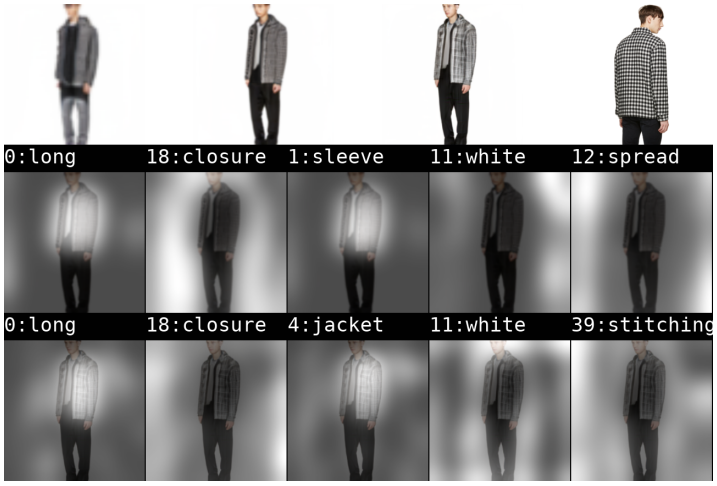
From Figure 4.13a to 4.13g, we can see that the quality of generated images becomes better as the stage increases, which verifies the effectiveness of the multi-stage attentional generative network. Among these examples, some appear to be novel designs, e.g., Figure 4.13a and 4.13b; some are indistinguishable from the corresponding real images, e.g., Figure 4.13c and 4.13d. They are not only close to photo-realistic quality, but also succeed in matching original descriptions with fine-grained textures, colors, shapes, etc. Since the gender is not specified in the description, the synthesized image is not restricted by the gender in the real image, e.g., Figure 4.13g.

Despite some compelling results, the model fails to attend the words with most relevant sub-regions in the synthesized images. For example, the attention area of 'sleeve' is as broad as that of 'long' and 'jacket' in Figure 4.13e. Besides, though the overall image is acceptable, the attention areas of 'collar', 'shoulders' and 'grey' are wrong in Figure A.1b, A.1e and A.1f, respectively. On the other hand, some distorted and malformed images are produced, e.g.,

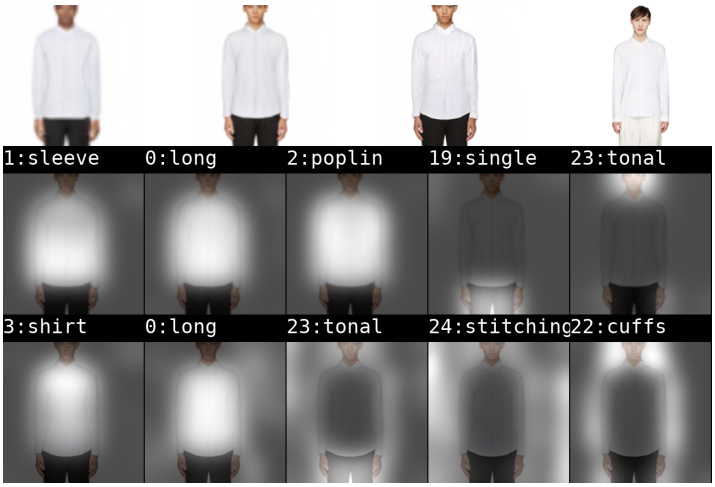
Figure 4.13h and 4.13i. This usually happens to accessories and shoes with much fewer training samples.



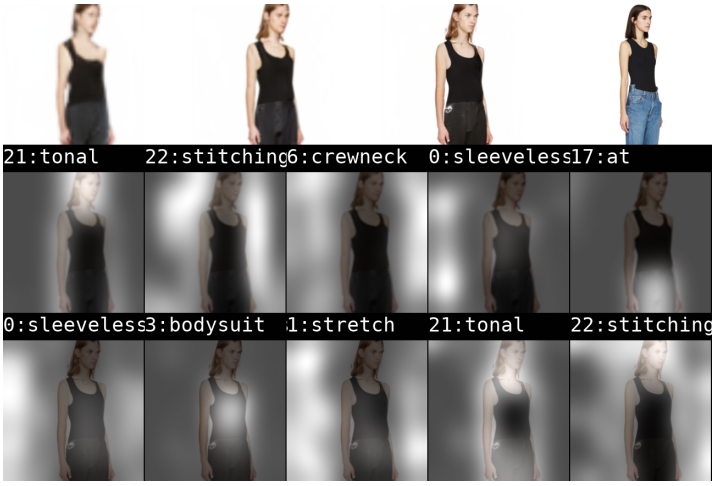
(a) Description: Mid-length pleated satin skirt in black featuring multicolor floral pattern. Concealed zip closure at side. Fully lined. Tonal stitching.



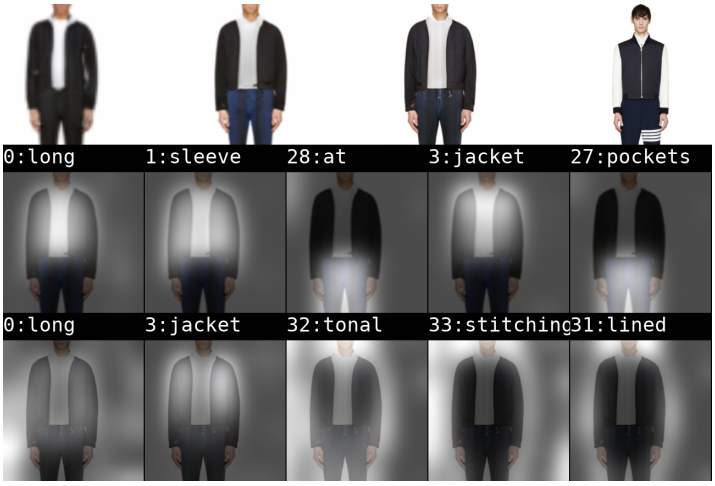
(b) Description: Long sleeve wool-blend jacket featuring check pattern in black and white. Spread collar. Button fastening concealing zip closure at front. Two-pocket styling. Elbow patch at sleeves. Adjustable cinch strap at cuffs. Fully lined. Silver-tone hardware. Tonal stitching.



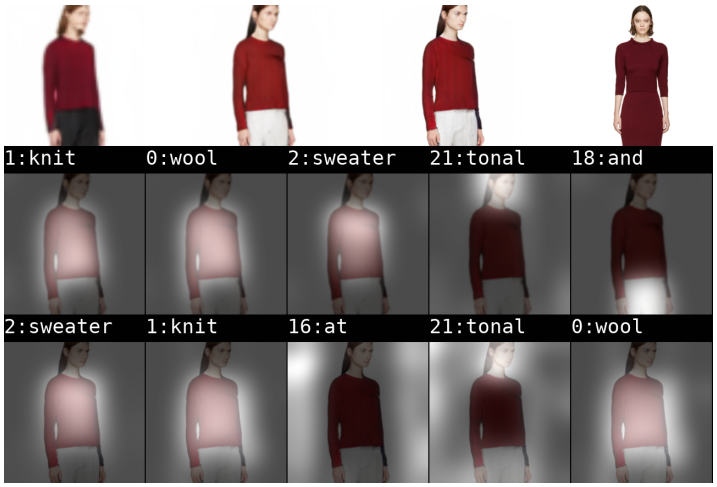
(c) Description: Long sleeve poplin shirt in white. Rubber studs at spread collar with bonded cotton overlay. Button closure at front. Single-button barrel cuffs. Tonal stitching.



(d) Description: Sleeveless stretch-jersey bodysuit in black. Crewneck collar. Press-stud closure at bottom. Signature stitching in white at nape of neck. Tonal stitching.

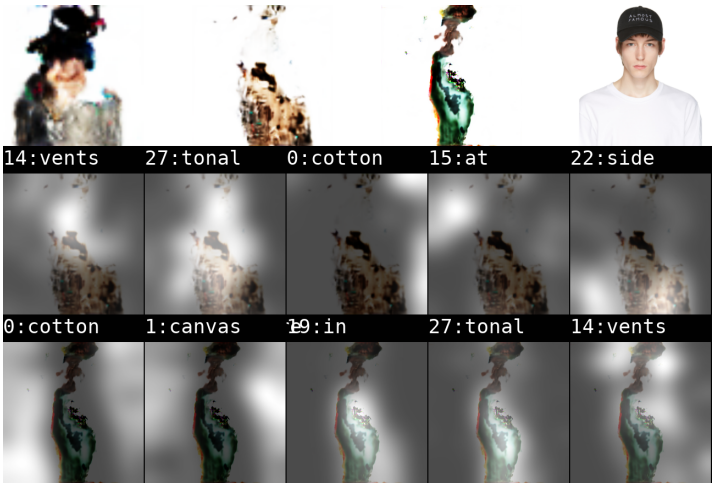


(e) Description: Long sleeve bomber jacket in deep navy. Ribbed knit stand collar, cuffs, and hem. Zip closure and welt pockets at front. Contrasting leather sleeves in white. Welt pockets at interior. Fully lined. Tonal stitching.

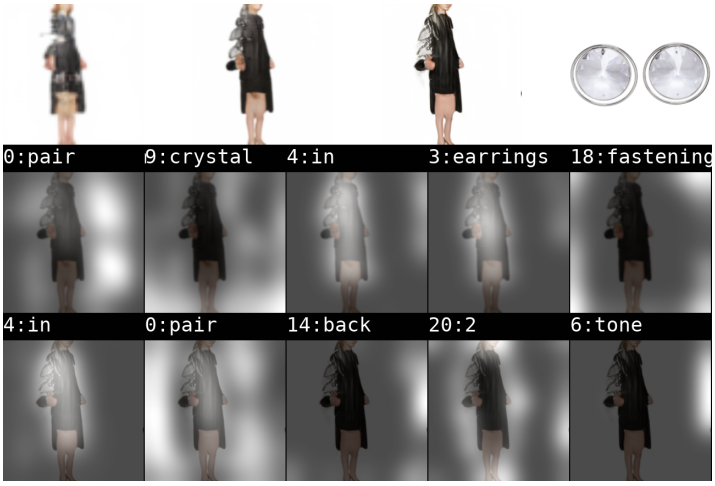


(f) Description: Wool knit sweater in bordeaux red. Cropped raglan sleeves. Padded raised crewneck collar. Ottoman knit panels at sleeves and at hem. Tonal stitching.

(g) Description: Long sleeve leather biker jacket in black. Wrinkled effect throughout. Round silver-tone studs at epaulet shoulders, front, and back. Notched lapel collar with press-stud fastening. Off-set zip closure. Zip pockets at front. Zippered sleeve cuffs. Adjustable belt at front waist. Fully lined. Tonal stitching.



(h) Description: Cotton canvas cap in black. Text embroidered in white at face. Curved brim. Eyelet vents at crown. Graphic embroidered in white at side. Cinch strap at back. Tonal stitching.



(i) Description: Pair of drop earrings in silver-tone brass. Transparent crystal accent. Logo engraved at back face. Post-stud fastening. Approx. 2" diameter.

Figure 4.13: Some map examples of synthesized images with AttnGAN on the validation set of FashionGen.

Given the same text from the validation set of FashionGen, six images are generated, as displayed in Figure 4.14. It can be seen that the designs usually do not change much, which is possibly because the descriptions are very specific. Nevertheless, the images varies in angles, providing chances to

view the whole body, half body and profile, etc.

(a) Description: Slim-fit jeans in black Japanese denim. Fading and whiskering throughout. Five-pocket styling. Zip-fly. Tonal stitching.

(b) Description: Knit tank top in navy. Heart print throughout in red. Ribbed crewneck collar and armscye in black. Asymmetrical scalloped hem. Tonal stitching.

(c) Description: Long sleeve denim jacket in 'light stonewash' blue. Fading and distressing throughout. Spread collar. Button closure at front. Flap and welt pockets at body. Logo flag in red at chest. Single-button barrel cuffs. Buttoned cinch tabs at back hem. Antique copper-tone hardware. Contrast stitching in yellow and tan.

(d) Description: Low-top buffed nappa leather sneakers in white. Round toe. Tonal lace-up closure. Eyelet vents at inner side. Padded collar. Heel tab in yellow featuring signature smiley embossed in black. Textured rubber sole in off-white. Tonal stitching.

Figure 4.14: Some images synthesized by AttnGAN given the same description from the validation set of FashionGen.

Fashion Synthesis

Similarly, for each map example from Fashion Synthesis shown in Figure 4.15, the first row gives the generated images from the first two stages of attentional

generative network, followed by the corresponding real image in the test set. The first generated images is bilinearly upsampled to be of the same size as the last one for better visualization. The second row illustrates top-5 words attended by the single attention model. More examples are put in Figure A.3 in the Appendix.

As can be seen from Figure 4.15, with the second stage, the quality of synthesized images is improved, which again demonstrates the model gradually learns useful information. Moreover, the words are attended more reasonably compared to the cases in FashionGen, such as 'sleeve', 'short', etc. Generally speaking, the generated images can reflect the meaning of the descriptions very well and only a minority of samples are distorted like Figure 4.15e. However, since the resolution of real images is low and the description is too general and unspecific, the synthesized images are restricted and lack of enough details inherently and they may be very different from imagination and thus undesired, e.g., Figure A.3e and A.3f.

(a) Description: the man is wearing a gray short-sleeved tee.

(b) Description: the lady wore a multicolor sleeveless dress.

(c) Description: the lady wore a purple long-sleeved top.

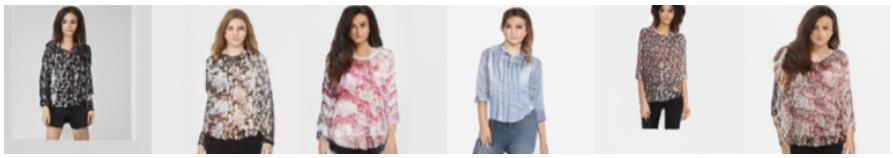
(d) Description: the woman was wearing a yellow long-sleeved blazer.

(e) Description: the lady was wearing a sweater with a multicolor long sleeve.

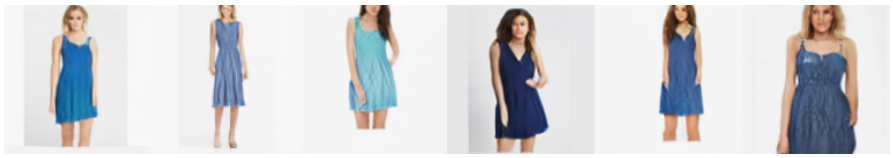
Figure 4.15: Some map examples of synthesized images with AttnGAN on the test set of Fashion Synthesis.

Given the same text from the test set of Fashion Synthesis, some images are generated and displayed in Figure 4.14. When the descriptions are not specific, the model can produce a variety of different designs. This also indicates that the trained AttnGAN is free of 'mode collapse'.

(a) Description: the lady is wearing a white long-sleeved blazer.



(b) Description: the lady wore a blouse with a multicolor long sleeve.



(c) Description: the lady wore a blue sleeveless dress.

(d) Description: ms. wearing a multi-color short-sleeved tee.

Figure 4.16: Some images synthesized by AttnGAN given the same description from the test set of Fashion Synthesis.

4.3 The results on the stacked generative adversarial network

There are much fewer hyper-parameters need to be determined in StackGAN compared to AttnGAN, since the text embeddings are given by DAMSM in AttnGAN. As shown in Figure 3.1, the model is trained stage-by-stage, that is, the Stage-I GAN needs to be first trained with the training text-image pairs, which is then fixed when training the Stage-II GAN. The synthesized images are also used for evaluation using FID to find the best hyper-parameters and models at different stages on each dataset. The numerical results are reported in the Table 4.1.

4.3.1 Synthesized images

FashionGen

Here, we display the synthesized images from StackGAN given the same descriptions as those in Figure 4.13 and A.1. For each example, the left image of size 64×64 is bilinearly upsampled to be of the same size as the right one for better visualization. From Figure 4.17, we can see that the quality of images is improved from the Stage-I GAN to the Stage-II GAN. It can also produce some novel designs. However, most synthesized images cannot match well with the descriptions with respect to color, shape, texture, etc. In most cases, they are less realistic than the corresponding ones in Figure 4.13. More examples are placed in Figure A.2 in the Appendix.

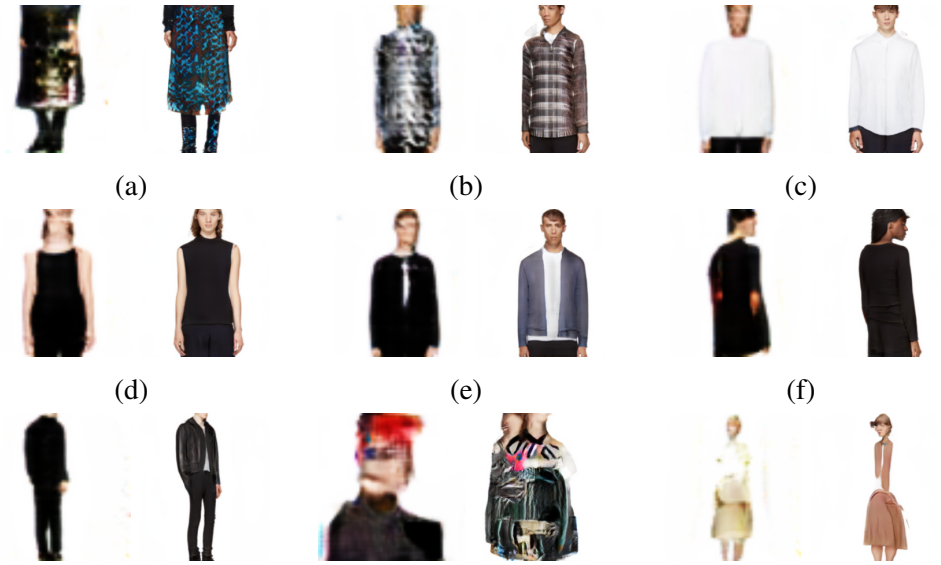


Figure 4.17: The examples of synthesized images with StackGAN on the validation set of FashionGen

Fashion Synthesis

Likewise, we display the synthesized images from StackGAN given the same descriptions as those in Figure 4.15 and A.3. In Figure 4.18, although the quality of images also get improved with the second stage, more images are less realistic and easier to be malformed than the corresponding ones in Figure 4.15. More examples can be found in Figure A.4 in the Appendix.

(a) (b) (c)

(d) (e)

Figure 4.18: The examples of synthesized images with StackGAN on the test set of Fashion Synthesis

Chapter 5

Discussions

In this chapter, we are going to analyze the results in Chapter 4. More specifically, the two adopted text-to-image synthesis models are compared and some problems in our experiments are discussed. Besides, the two fashion datasets are compared in terms of their respective advantages and disadvantages. All of these lead to some future work that can be done for improvement of text-to-image synthesis for fashion design.

5.1 Two text-to-image synthesis algorithms

In this project, both AttnGAN and StackGAN share the same text encoder that is provided by pretraining DAMSM in AttnGAN. Thus, they have the same text embeddings as input. Based on this, we attempt to compare their performance with respect to the network architectures. Both AttnGAN and StackGAN have several stages in their generative networks, which accounts for improvement of image quality inside the models. The numerical results in Table 4.1 and image examples shown in Chapter 4 and Appendix illustrate that AttnGAN is superior to StackGAN on FashionGen where the descriptions are much longer, demonstrating that the attention mechanism in AttnGAN plays a crucial role in synthesizing fine-grained sub-regions of images that align the corresponding descriptions at the word level. However, it achieves worse FID than StackGAN on Fashion Synthesis in our experiments, though the synthesized images look better. It is likely that FID may not always be consistent with human evalution.

Compared to StackGAN, AttnGAN has much more hyper-parameters to be tweaked which have been discussed in section 4.1 and section 4.2. Among all of them, we find the choice of batch size used for pretraining DAMSM will affect the DAMSM loss at both the word level and sentence level significantly,

as shown in Figure 4.7 and 4.8. As opposed to the common practice, smaller batch sizes would drop the loss. This is because they are noisier and those explore the loss landscape better. It allows to jump from one local optimum to another. Another possible reason is that the DAMSM loss is calculated with M text-image pairs in Equation 3.12 and 3.14, i.e., batch size is equal to M. With smaller M, it is easier for T_i to match with $\tilde{X}_i$ rather than with the other $M-1$ description candidates and thus the loss will be reduced. As another important hyper-parameter in AttnGAN, λ in Equation 3.19 reflects how important the DAMSM loss is when training the attentional generative model. In our experiments, it turns out that λ of FashionGen is 10 times larger than that of Fashion Synthesis as mentioned in subsection 4.2.1, which implies that FashionGen is more complex with respect to the images and texts. As illustrated in Figure 4.14 and 4.16, we do not observe 'mode collapse' in AttnGAN. However, it can be noticed in Figure 4.11 and 4.12 that the FID learning curves on the validation/test dataset usually oscillate, indicating the instability of training GANs that sensitive to choices of parameters. Thus, the tuning of hyper-parameters and choices of models become less reliable. Although we have put much effort into tuning as many hyper-parameters as possible, it seems they are still not enough, since we leave the compositions of GANs unchanged where some potential parameters are involved, e.g., the dimensions of the hidden features in the generator and discriminator, the number of residual blocks, etc. In fact, we face the common problem of deep learning algorithms that the training usually takes a very long time because of large numbers of parameters and large amount of training data. In our experiments, both AttnGAN and StackGAN need to be trained from scratch. For example, the first-two-stage attentional generative network in AttnGAN with FashhionGen takes 10 days or so; the complete attentional generative network with FashionGen takes around 25 days.

Due to the limited time, we do not employ other models apart from DAMSM to produce text embeddings for comparison of their quality in the downstream text-to-image synthesis task. Moreover, we do not use extrinsic or intrinsic methods to evaluate the quality of learned word embeddings and text embeddings and only rely on the internal DAMSM loss to determine text encoder, which increases the risk that the input embeddings is not good enough for the GANs. As for the evaluation of the performance of GANs, we merely use FID, which sometimes is not consistent with human evaluation in our experiments. It calls for more focus and energy to be put into finding better evaluation metrics for image generation models.

Hence, there are still lots of aspects worth studying in the area of text-

to-image synthesis, e.g., devising effective regularization methods to stabilize the training of GANs and developing more advanced models to improve both image quality and alignment between text and synthesized image.

5.2 Two fashion datasets

In this project, FashionGen and Fashion Synthesis are used. We attempt to study the impact of using different training data on the performance of text-to-image synthesis. Since FID is only used to evaluate the quality of generated images based on the statistics of the same dataset, it cannot be used to compare the qualities of different datasets.

Models trained with FashionGen can synthesize more distinct images than with Fashion Synthesis, since FashionGen inherently has the advantage of higher resolution. In the cases where the descriptions fall in the most common categories in FashionGen, the synthesized images can be very close to the real ones or they are completely novel designs. When using AttnGAN, the details of designs are well attended in the generated images, e.g., textures, colors, shapes, etc. However, it is not uncommon to see some words are attended improperly, which is less disappointing for Fashion Synthesis. This is possibly due to the large amount of vocabulary in FashionGen, increasing the difficulty of learning the difference of words. Futhermore, some synthesized images are distorted and malformed for the less common categories. Hence, the imbalanced categories in FashionGen also degrades overall performance in our experiments. One possible solution is to set class weights or sample weights during training. Since the descriptions in FashionGen do not include gender information, as a future work, it can be integrated into the description for guidance.

In general, models trained with Fashion Synthesis can generate more matching samples. Since the descriptions are more unspecific than those in FashionGen, it allows to generate variant images conditioned on the same text. However, it will not support the detailed designs, since the synthesized images are inherently restricted by the low resolution and broad descriptions.

Chapter 6

Conclusions

The objective of this project was to train a neural network generative model to generate high-quality images of fashion items conditioned on the text. Two GAN-based algorithms, AttnGAN and StackGAN, were adopted and compared on two fashion datasets, FashionGen and Fashion Synthesis. For both AttnGAN and StackGAN, they can progressively improve the image quality through the stages in their generative networks. Through our experiments, AttnGAN exhibited more strength in synthesizing high-quality images for the most common categories in FashionGen, which contains much longer descriptions. This indicated the effectiveness of the attention mechanisms in AttnGAN. Although StackGAN achieved better FID than AttnGAN on Fashion Synthesis, the generated images looked less realistic, showing inconsistency between quantitative evaluation and human evaluation. In general, models trained with FashionGen can synthesize more distinct images, while models trained with Fashion Synthesis can generate more matching samples. Although some intermediate results gave some useful experience, there existed many inadequacies in our work with respect to dataset usage, tuning hyperparameters, model selection and evaluation.

Bibliography

[1] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[2] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[3] Mehdi Mirza and Simon Osindero. “Conditional generative adversarial nets”. In: *arXiv preprint arXiv:1411.1784* (2014).

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

[5] Xinchen Yan et al. “Attribute2image: Conditional image generation from visual attributes”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 776–791.

[6] Jun-Yan Zhu et al. “Unpaired image-to-image translation using cycle-consistent adversarial networks”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2223–2232.

[7] Han Zhang et al. “Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5907–5915.

[8] Tao Xu et al. “Attngan: Fine-grained text to image generation with attentional generative adversarial networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1316–1324.

[9] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. “A Simple but Tough-to-Beat Baseline for Sentence Embeddings”. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017. URL: `https://openreview.net/forum?id=SyK00v5xx`.

[10] Tomas Mikolov et al. “Distributed Representations of Words and Phrases and Their Compositionality”. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS’13. Lake Tahoe, Nevada: Curran Associates Inc., 2013, pp. 3111–3119. URL: http://dl.acm.org/citation.cfm?id=2999792.2999959.

[11] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. “Glove: Global vectors for word representation”. In: *In EMNLP*. 2014.

[12] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015. URL: http://arxiv.org/abs/1409.0473.

[13] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. “Sequence to Sequence Learning with Neural Networks”. In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS’14. Montreal, Canada: MIT Press, 2014, pp. 3104–3112. URL: http://dl.acm.org/citation.cfm?id=2969033.2969173.

[14] Scott E Reed et al. “Learning what and where to draw”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 217–225.

[15] Shikhar Sharma et al. “Chatpainter: Improving text to image generation using dialogue”. In: *arXiv preprint arXiv:1802.08216* (2018).

[16] Shengyu Zhang et al. “Text-to-image synthesis via visual-memory creative adversarial network”. In: *Pacific Rim Conference on Multimedia*. Springer. 2018, pp. 417–427.

[17] Scott Reed et al. “Learning deep representations of fine-grained visual descriptions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 49–58.

[18] Scott Reed et al. “Generative adversarial text to image synthesis”. In: *arXiv preprint arXiv:1605.05396* (2016).

[19] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks”. In: *arXiv preprint arXiv:1511.06434* (2015).

[20] Han Zhang et al. “Stackgan++: Realistic image synthesis with stacked generative adversarial networks”. In: *arXiv preprint arXiv:1710.10916* (2017).

[21] Zizhao Zhang, Yuanpu Xie, and Lin Yang. “Photographic text-to-image synthesis with a hierarchically-nested adversarial network”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6199–6208.

[22] Catherine Wah et al. “The caltech-ucsd birds-200-2011 dataset”. In: (2011).

[23] Tsung-Yi Lin et al. “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[24] Liqian Ma et al. “Pose guided person image generation”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 406–416.

[25] Patrick Esser, Ekaterina Sutter, and Björn Ommer. “A variational u-net for conditional appearance and shape generation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8857–8866.

[26] Xintong Han et al. “Viton: An image-based virtual try-on network”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7543–7552.

[27] Shizhan Zhu et al. “Be your own prada: Fashion synthesis with structural coherence”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 1680–1688.

[28] Mehmet Günel, Erkut Erdem, and Aykut Erdem. “Language Guided Fashion Image Manipulation with Feature-wise Transformations”. In: *arXiv preprint arXiv:1808.04000* (2018).

[29] Tim Salimans et al. “Improved techniques for training gans”. In: *Advances in neural information processing systems*. 2016, pp. 2234–2242.

[30] Takeru Miyato et al. “Spectral normalization for generative adversarial networks”. In: *arXiv preprint arXiv:1802.05957* (2018).

[31] Martin Heusel et al. “GANs Trained by a Two Time-Scale Update Rule Converge to a Nash Equilibrium”. In: *CoRR* abs/1706.08500 (2017). arXiv: 1706.08500. URL: http://arxiv.org/abs/1706.08500.

[32] Casper Kaae Sønderby et al. “Amortised MAP Inference for Image Super-resolution”. In: *CoRR* abs/1610.04490 (2016). arXiv: 1610.04490. URL: http://arxiv.org/abs/1610.04490.

[33] Negar Rostamzadeh et al. “Fashion-gen: The generative fashion dataset and challenge”. In: *arXiv preprint arXiv:1806.08317* (2018).

[34] Ziwei Liu et al. “DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations”. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[35] N. Rostamzadeh et al. “Fashion-Gen: The Generative Fashion Dataset and Challenge”. In: *ArXiv e-prints* (June 2018). arXiv: 1806.08317 [stat.ML].

[36] Shane Barratt and Rishi Sharma. “A note on the inception score”. In: *arXiv preprint arXiv:1801.01973* (2018).

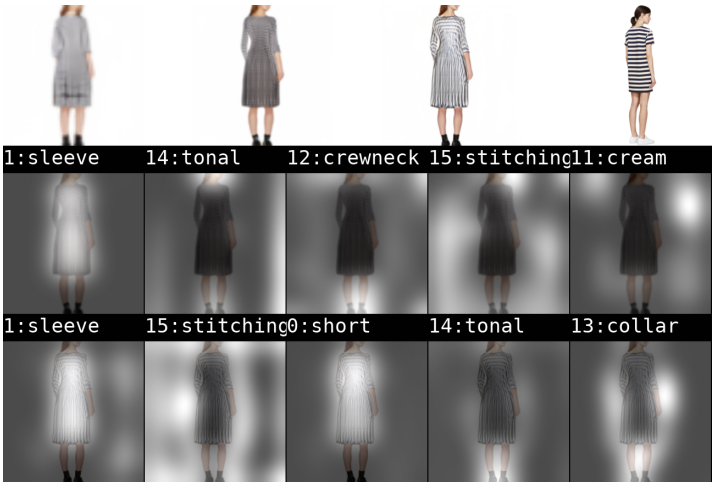
[37] Amir Bakarov. “A Survey of Word Embeddings Evaluation Methods”. In: *CoRR* abs/1801.09536 (2018). arXiv: 1801.09536. URL: http://arxiv.org/abs/1801.09536.

[38] Laurens van der Maaten and Geoffrey Hinton. “Visualizing Data using t-SNE”. In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605.

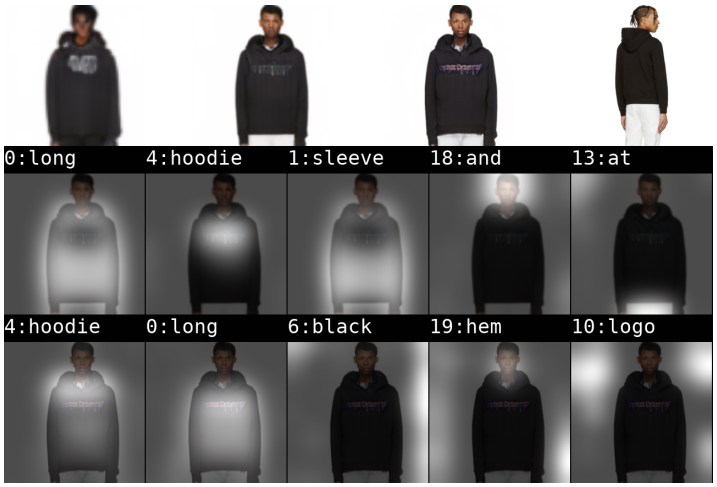
Appendix A

Extra Results

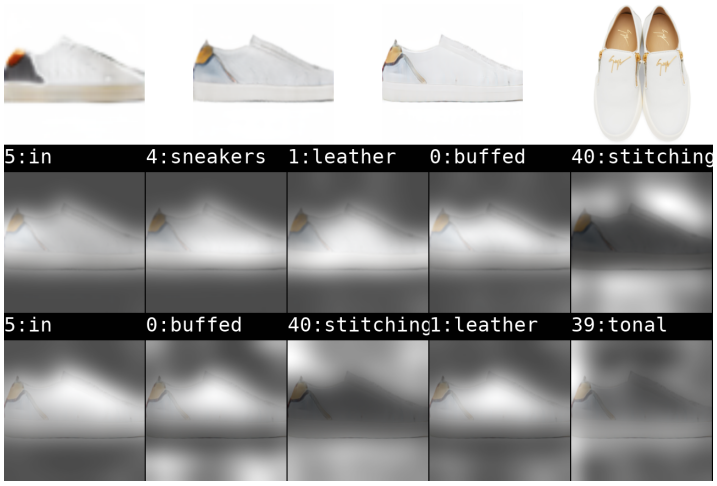
(a) Description: Short sleeve cotton jersey t-shirt in black. Rib knit crewneck collar. Multicolor graphic and text printed at chest and back. Tonal stitching.



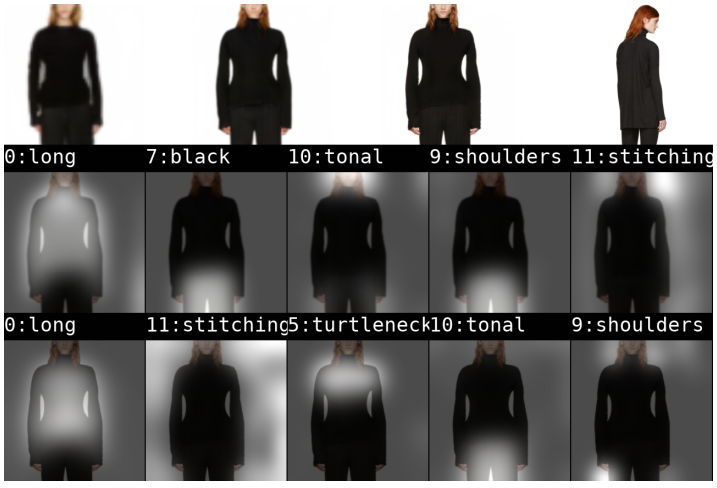
(b) Description: Short sleeve mid-weight cotton jersey dress striped in navy and cream. Crewneck collar. Tonal stitching.



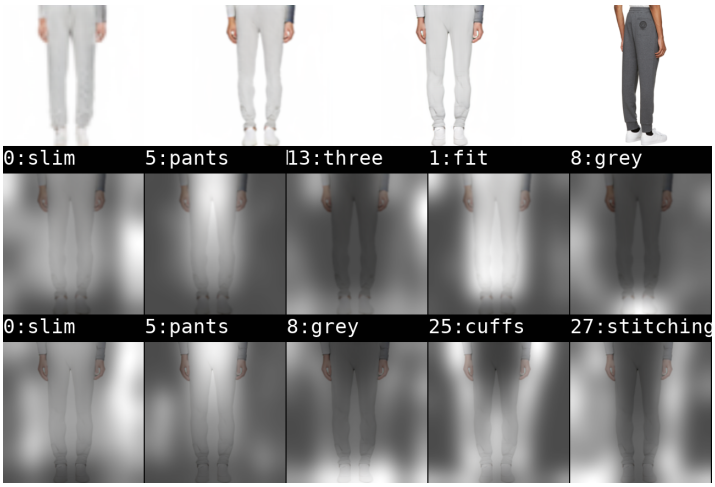
(c) Description: Long sleeve French terry hoodie in black. Drawstring at hood. Logo graphic printed at front. Rib knit cuffs and hem.



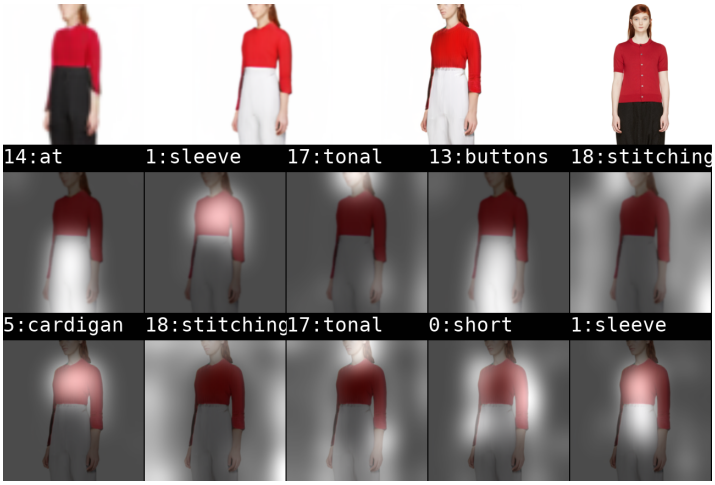
(d) Description: Buffed leather slip-on sneakers in white. Round toe. Signature zippered vent at eyerows. Logo plaque in gold-tone at padded bellows tongue. Leather lining in beige. Textured rubber midsole and treaded rubber sole in white. Gold-tone hardware. Tonal stitching.



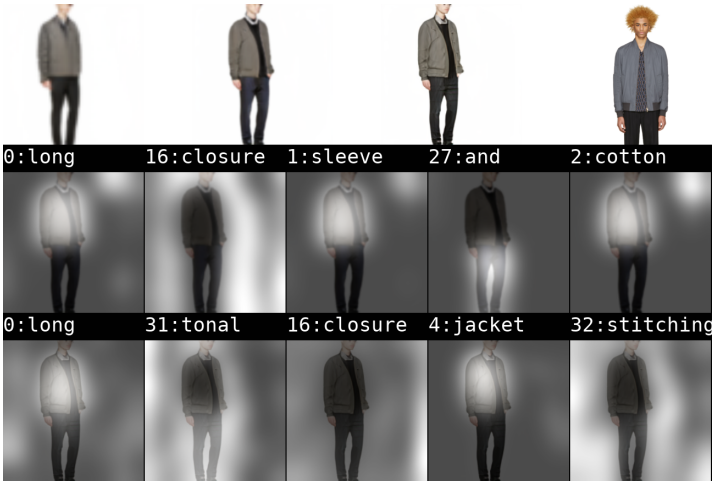
(e) Description: Long sleeve rib knit jersey turtleneck in black. Dropped shoulders. Tonal stitching.



(f) Description: Slim-fit cotton fleece lounge pants in heather grey. Drawstring at elasticized waistband. Three-pocket styling. Logo embroidered in black at back pocket. Rib knit cuffs. Tonal stitching.



(g) Description: Short sleeve ribbed knit wool cardigan in red. Crewneck collar. Pearlescent dark grey buttons at front closure. Tonal stitching.

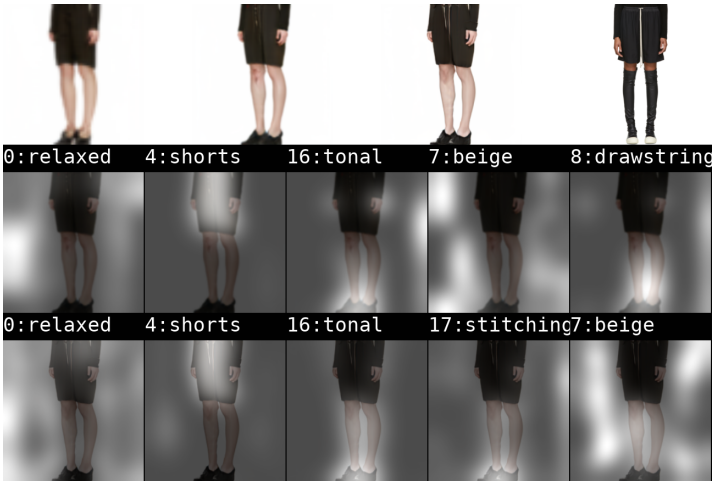


(h) Description: Long sleeve cotton bomber jacket in grey. Stand collar. Patch pocket at sleeves. Two-way zip closure at front. Welt pockets at waist. Rib knit sleeve cuffs and hem. Fully lined. Tonal stitching.

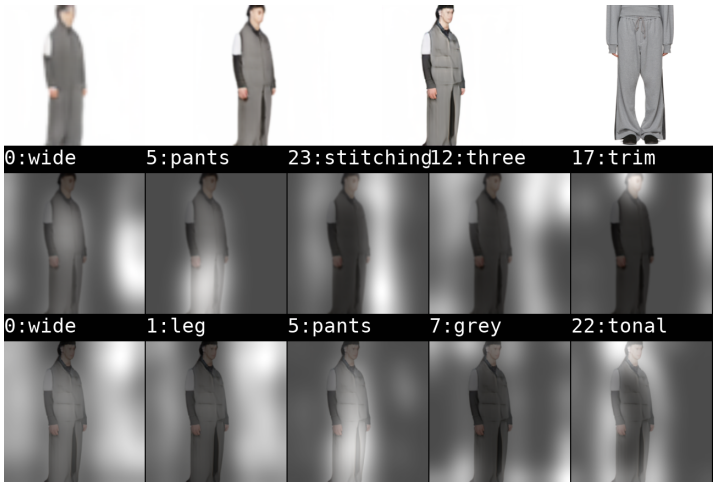
(i) Description: Grained calfskin tote in black. Twin rolled carry handles featuring press-stud fastener. Detachable and adjustable shoulder strap with lanyard clasp fastening. Logo stamp in gold-tone at face. Canvas panel featuring signature 'house' check pattern at sides. Press-stud fastening at main compartment. Patch pocket, zippered pocket, and leather logo patch at interior. Tonal textile lining. Bumper stud

(j) Description: Slim-fit jeans in 'dark paz' blue. Mid-rise. Fading throughout. Five-pocket styling. Logo embroidered in off-white at back pocket. Button-fly. Antiqued brass-tone hardware. Contrast stitching in tan. Approx. 6" leg opening.

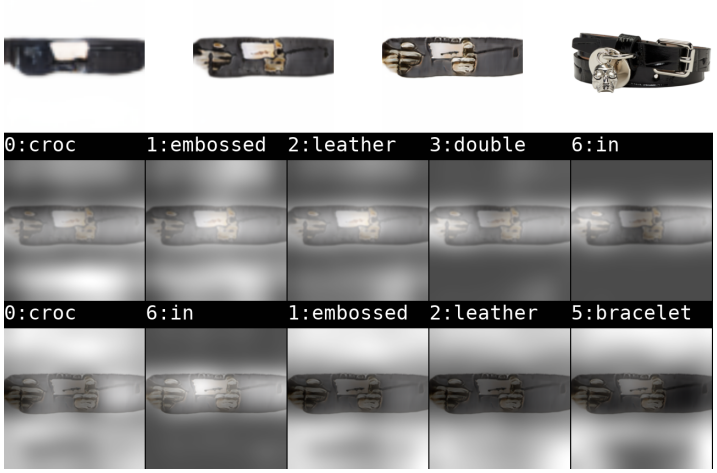
(k) Description: Long sleeve quilted down coat in matte navy. Stand collar. Removable hood with press-stud fastening and tonal bungee drawstring. Two-way zip closure at front. Patch pocket at exterior arm with signature logo detail. Zippered pockets at front. Tonal stitching.



(l) Description: Relaxed-fit viscose crepe shorts in black. Beige drawstring at elasticized poplin waistband. Two-pocket styling. Tonal stitching.



(m) Description: Wide-leg French terry lounge pants in grey. Drawstring at elasticized waistband. Three-pocket styling. Tonal denim trim at outseams. Zip-fly. Tonal stitching.



(n) Description: Croc-embossed leather double-wrap bracelet in black. Signature carved skull accent and logo-engraved disc at closure. Pin-buckle fastening. Silver-tone hardware. Approx. 16.75" length.

Figure A.1: More map examples of synthesized images with AttnGAN on the validation set of FashionGen.

Figure A.2: More examples of synthesized images with StackGAN on the validation set of FashionGen

(a) Description: the lady wears a pink long-sleeved blouse.

(b) Description: the lady was wearing a black sleeveless tank.

(c) Description: the lady wore a gray long-sleeved cardigan.

(d) Description: the lady is wearing a pink short-sleeved tee.

(e) Description: the lady wore a multicolor sleeveless dress.

(f) Description: the lady is wearing a multi-colored long-sleeved dress.

(g) Description: the lady is wearing a beige long-sleeved parka.

Figure A.3: More map examples of synthesized images with AttnGAN on the test set of Fashion Synthesis.

(a) (b) (c)

(d) (e) (f)

(g)

Figure A.4: More examples of synthesized images with StackGAN on the test set of Fashion Synthesis

TRITA -EECS-EX-2019:753