IIAI Open Conference Publication Series
Volume title
Volume number, Article ID
DOI: https://doi.org/10.52731/XXXXXX

GAN-based Detailed Clothing Generation System

Ryozo Masukawa *
ee187111@meiji.ac.jp

Shosuke Haji *
hajisho@cs.meiji.ac.jp

Masane Fuchi *
gshvsuhdtvzg@gmail.com

Kazuki Yamaji *
yamajikazuki@cs.meiji.ac.jp

Taiga Matsui †
t.matsui@air-closet.com

Keita Ishikawa †
k.ishikawa@air-closet.com

Tomohiro Takagi *
takagi@cs.meiji.ac.jp

Abstract

The implementation of Generative Adversarial Networks (GAN) in the fashion domain has been researched for various applications such as virtual-try-on, fashion item recommendation, and design generation. In this paper, we propose a GAN-based fashion design generation system that reduces the workload of the labor-intensive design creation task. Our system consists of two generative models: one that produces images of fashion items without any clothing patterns using conditioned StyleGAN2-ADA, and one that is a style transfer model reflecting the fine texture of the fashion item. The system also allows users to edit images of garments by manipulating the latent code of the generator. We demonstrate through qualitative and quantitative experiments that the proposed system trained on a dataset of real clothing inventory images can generate realistic and diverse images that reflect the input conditions in detail.

Keywords

Keywords: Generative Adversarial Networks, Swapping Autoencoder, Style Transfer, Fashion Design Generation

* Department of Computer Science, Meiji University, Kanagawa, Japan
† airCloset, Inc.

1 Introduction

Generally speaking, one of the main tasks in the fashion industry is to analyze the future fashion demands and produce novel fashion designs that fit with the trend forecast result. However, because it requires designers to forecast the unseen future, creating designs of fashionable garments is a labor-intensive and time-consuming task. We design our system to reduce the workload of producing novel fashion designs. Our system utilizes GAN-based generative models to produce detailed garment images conditioned by the input fashion attributes that fashion experts consider trendy, and designers can discover novel design ideas by browsing images generated from them.

Since our aim here is to include diverse garments, we use generative models rather than simple Information Retrieval (IR) models. For example, IR models can only retrieve clothing images that match items already in the database and when designers have to edit the details of a garment, the model cannot apply such modification. In contrast, GAN models are able to condition the images they generate in detail, and therefore, we use GANs to enable versatile edits on clothing images.

2 Related Works

Our proposed system consists of three key modules: a Generative Adversarial Networks (GANs) [1] module, a texture synthesis module using a neural style transfer model, and an image edit module that manipulates latent codes. We introduce several related works in this section. As far as we know, our proposed system is the first that can generate high-resolution fashion item images by combining conditional-StyleGAN2-ADA [2], style transfer model [3], and latent space editing module [4].

2.1 Application of Generative Adversarial Networks in fashion

Generative Adversarial Networks (GANs) [1] have been improved to the point that they can generate realistic high-resolution images, and research on GAN applications for the fashion domain such as virtual try-on [5] and fashion item recommendation ([6][7]), is now being conducted extensively. At the same time, detailed fashion image editing methods such as TailorGAN [8], which utilizes GAN to edit the lengths of sleeves and collars of garments have been proposed.

One of the most promising methods is StyleGAN [9], which improves PGGAN [10] by using Adaptive Instance Normalization (AdaIN) [11] to generate realistic high-resolution images. StyleGAN image generation tasks can be applied to fashion image generation techniques such as style transfer-based virtual try-on [12] and fashion outfit generation [13]. StyleGAN2 [14] improves the original StyleGAN's training stability by refining AdaIN and introducing a lazy path length regularization method. StyleGAN2-ADA [2] can utilize data augmentation, which enables StyleGAN2 [14] generation to be trained from limited data. Our proposed system using StyleGAN2-ADA [2] generates images conditioned by using the principle of Conditional GAN [15].

2.2 Texture Synthesis

Although neural style transfer models that reflects the texture of one image onto another [16] have been studied frequently, this is not practical due to the difficulty of collecting vast

amounts of paired image data that is required to train such models. Swapping autoencoder [3], in contrast, can be trained without a paired image dataset and can encode the structure and texture of one image into two different latent codes. It can also effectively conflate the texture information of one target image with another input image by utilizing a Patch Discriminator, that judges whether a randomly cropped patch of the synthesized image belongs to the target texture image.

2.3 Deep Image Manipulation with StyleGAN latent code modification

Various image manipulation methods that modify the disentangled latent code of StyleGAN [9] have been proposed. For example, Collins et al. [17], developed a method that implements local image editing by means of spherical k-means clustering [18] applied to the weights of each progressive Affine transfer blocks of StyleGAN. Another method, GANSpace [19], uses PCA-based data sampling to detect interpretable image editing directions. However, such methods requires time-consuming optimization [17] or data sampling [19]. An unsupervised latent space editing method named SeFa [4] that requires neither training on a generator nor data sampling has recently been introduced. SeFa utilizes the eigenvectors of arbitrary GAN convolutional generator block weights and dramatically reduces the image editing time for manipulation on a latent code.

3 Dataset

Each module in our proposed system is trained on a dataset that contains 62k clothing images and corresponding condition labels (seven categories, ten silhouette labels, 12 length labels, 16 colors, and 20 patterns) collected from the inventory of airCloset, Inc from October 2018 to April 2022. The dataset is not publicly available.

4 Method

The overall architecture of our proposed system is shown in Figure. 1. Our proposed system consists of three major modules. The first is a "Multiple Conditional StyleGAN2-ADA Module" that utilizes StyleGAN2-ADA [2] to generate the clothing images with no patterns on the basis of four conditions: category, silhouette, length, and color (Sec. 4.1). The second is a "Swapping Autoencoder Module" that utilizes a swapping autoencoder [3] to reflect the input clothing pattern into the garment image generated by the first module (Sec. 4.2). The final module is the "SeFa Color Editing Module", which applies unsupervised latent Semantics Factorization (SeFa [4]) to revise the color of the output synthesized image from the second module to match the input color label(Sec. 4.3).

4.1 StyleGAN2-ADA For Clothing Image Generation With No Patterns

From a latent code $\boldsymbol{z}$ in the latent space $\mathscr{Z}$, the mapping network of the original StyleGAN2 [14] generator f ($f : \mathscr{Z} \to \mathscr{W}$) projects $\boldsymbol{z}$ into the intermediate latent code $\boldsymbol{w} \in \mathscr{W}$. When conditionally trained, f has a functionality to project $\boldsymbol{z}$ into $\boldsymbol{w}$ with the embedded condition vector $\boldsymbol{y}$, as

$$\boldsymbol{w} = f(\boldsymbol{z}, \boldsymbol{y}), \quad \boldsymbol{y} = g(c), \tag{1}$$

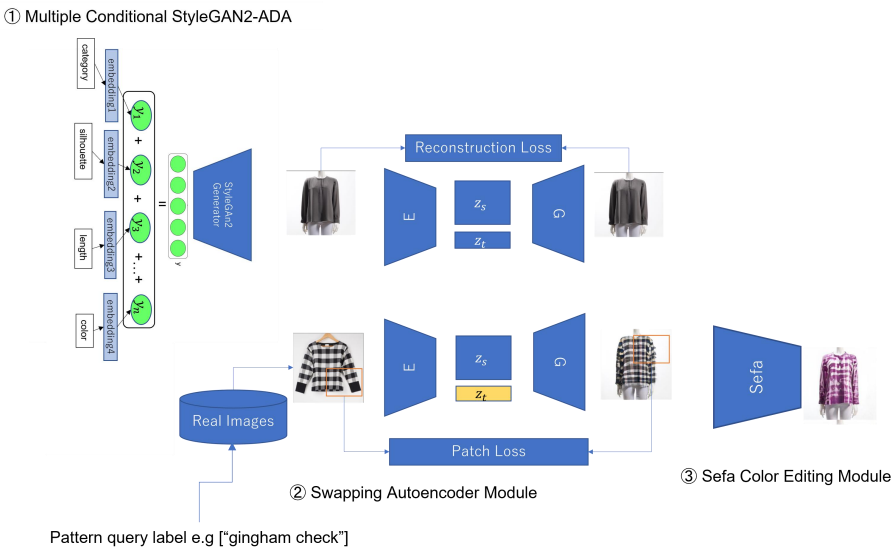


Figure 1: Overall architecture of proposed system.

where g is a embedding layer and c is a condition label. In order to reflect all four conditions (clothing category, silhouette, length, color), StyleGAN2 needs to be able to apply multiple conditions. Therefore, we modified the mapping network f in the StyleGAN2 [14] generator so that it can represent an arbitrary number of conditions. As our aim is to reflect the combination of four conditions $\boldsymbol{c} = \{c_{cate}, c_{sil}, c_{len}, c_{color}\}$, we define linear embedding layers for each conditions $g_c, c \in \boldsymbol{c}$. Therefore, our refined mapping network architecture is

$$\boldsymbol{w} = f(\boldsymbol{z}, \boldsymbol{y}), \boldsymbol{y} = [g_{cate}(c_{cate}), g_{sil}(c_{sil}), g_{len}(c_{len}), g_{color}(c_{color})], \tag{2}$$

where $c_{cate}, c_{sil}, c_{len}, c_{color}$ are condition labels for clothing category, silhouette, length, and color, respectively. The final output generated image I is defined by using the synthesis layer of StyleGAN2-ADA (G_s), as

$$I = G_s(\boldsymbol{w}). \tag{3}$$

4.2 Clothing Pattern Synthesis Module

In the StyleGAN2-ADA module discussed above, we generated images constrained by relatively simple conditions (category, silhouette, length, color). However, due to the prodigious diversity of clothing patterns, we cannot train to represent all clothing patterns using only StyleGAN2-ADA. We therefore apply the swapping autoencoder [3] proposed by Park et al. to synthesize clothing pattern from real images rather than generating them from scratch. The model utilizes an encoder decoder network where the encoder (Enc) outputs a structure tensor $(\boldsymbol{z_s})$ and a texture vector $(\boldsymbol{z_t})$ from a given image I to split the information of the structure and texture disparately, as

$$Enc(I) = \boldsymbol{z_s}, \boldsymbol{z_t}. \tag{4}$$

At the same time, the residual StyleGAN2 [14]-based decoder/generator of the swapping autoencoder (G_{swapAE}) reconstructs the input image (I_{rec}) from the two tensors ($\boldsymbol{z_s}$, $\boldsymbol{z_t}$) as

$$I_{rec} = G_{swapAE}(\boldsymbol{z_s}, \boldsymbol{z_t}). \tag{5}$$

This architecture allows one image (I_A) to mix the texture of another image (I_B). Concretely speaking, the mixed image I_{mix} can be obtained by swapping the texture vector of I_A ($\boldsymbol{z_t^A}$) with the texture vector of I_B ($\boldsymbol{z_t^B}$) while preserving the structure tensor $\boldsymbol{z_s^A}$ of the original image as

$$I_{mix} = G_{swapAE}(\boldsymbol{z_s^A}, \boldsymbol{z_t^B}). \tag{6}$$

In our system, when generating garment images with patterns, we use swapping autoencoder to synthesize the clothing pattern conditioned by the input with the image generated by the first StyleGAN2-ADA module (I). Strictly speaking, we retrieve the real image (I_{tgt}) that matches the pattern label from the dataset and obtain the texture vector ($\boldsymbol{z_t^{tgt}}$), then swap the texture vector of the input image I with the texture vector of I_{tgt} as

$$I_{patterned} = G_{swapAE}(\boldsymbol{z_s}, \boldsymbol{z_t^{tgt}}), \tag{7}$$

where $I_{patterned}$ is the output texture synthesized image.

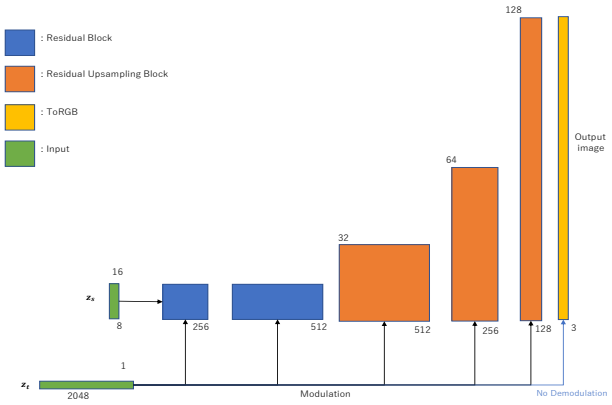


Figure 2: Swapping autoencoder generator.

4.3 SeFa Module For Color Matching

The texture synthesis applied to the generated image in the second module assumes that all combinations of color and clothing patterns are inside the real dataset. In reality, since our dataset do not include all the combinations, it cannot represent novel combinations. Therefore, if a pair of color and clothing pattern that does not exist in the dataset becomes the novel trend, we cannot generate garment images consistent with the current trend if we only use the swapping autoencoder. To resolve this problem, we utilize a closed-form factorization method proposed by Shen et al. [4] that discovers latent interpretable directions in the

GAN generator to edit the color of the output image from the second swapping autoencoder module. We found from a prior experiment (discussed in Sec. 5.3), that applying SeFa on the weights of the Convolutional Neural Network (CNN) layers in the ToRGB block of the swapping autoencoder shown in Figure. 2 enables us to edit the color with minimal effect on the image structure or texture.

SeFa [4] regards each transformation step in the CNN-based generator G as an affine transformation. The first transformation step of one CNN-based generator (G_1) with the d-dimensional input latent code ($\boldsymbol{z}$) can be formulated as

$$G_1(\boldsymbol{z}) \triangleq \mathbf{A}\boldsymbol{z} + \boldsymbol{b}, \tag{8}$$

where $\mathbf{A} \in \mathbb{R}^{m \times d}$, $\boldsymbol{b} \in \mathbb{R}^m$ represent the weight and bias of G_1, respectively. The image editing using the modified latent code with one semantic direction vector $\boldsymbol{n} \in \mathbb{R}^d$ can be defined as

$$edit(G(\boldsymbol{z})) = G(\boldsymbol{z} + \alpha\boldsymbol{n}), \tag{9}$$

where α is the strength of interpretable direction $\boldsymbol{n}$.

From Equations. (8) and (9), we can define the editing in the first transformation step in the generator (G_1) as

$$edit(G_1(\boldsymbol{z})) = G_1(\boldsymbol{z}) + \alpha\mathbf{A}\boldsymbol{n}. \tag{10}$$

Therefore, the manipulation of the latent code can be completed independently of the original instance $G_1(\boldsymbol{z})$ simply by adding $\alpha\mathbf{A}\boldsymbol{n}$ after the first transformation G_1. By solving the following optimization problem, where the solutions are eigenvectors of the matrix $\mathbf{A}^\top\mathbf{A}$,

$$\mathbf{N}^* = \underset{\{\mathbf{N}\in\mathbb{R}^{d\times k}:\boldsymbol{n_i}^\top\boldsymbol{n_i}=1\forall i=1,\ldots,k\}}{\arg\max} ||\mathbf{A}\boldsymbol{n}||_2^2, \tag{11}$$

we can retrieve the optimal k interpretable directions $\mathbf{N}^* = [\boldsymbol{n}_1^*, \boldsymbol{n}_2^*, \ldots, \boldsymbol{n}_k^*]$.

We empirically found that G_{swapAE} controls the color of the generated images at the final ToRGB block (G_{ToRGB}); strictly speaking, the weights of the CNN layers in G_{ToRGB} ($\mathbf{A}_{ToRGB}$). Therefore, we can edit the color of the clothing patterns in applied images while maintaining the structure and texture by retrieving k optimal interpretable directions $\mathbf{N}^* = [\boldsymbol{n}_1^*, \boldsymbol{n}_2^*, \ldots, \boldsymbol{n}_k^*]$ from the eigenvectors of weight matrix $\mathbf{A}_{ToRGB}^\top\mathbf{A}_{ToRGB}$ using SeFa as

$$edit(G_{ToRGB}(\boldsymbol{z_t})) = G_{ToRGB}(\boldsymbol{z}_t^{tgt}) + \alpha\mathbf{A}_{ToRGB}\boldsymbol{n}^*, \tag{12}$$

where $\boldsymbol{n}^* \in \mathbf{N}$. The details of the experiment applying SeFa in each layers of the generator of the swapping autoencoder are discussed in Sec. 5.3.

5 Experiments

We conducted experiments on each module of the proposed system (Multiple Conditional StyleGAN2-ADA Module: Sec. 5.1, Swapping Autoencoder Module: Sec. 5.2, and SeFa Color Editing Module: Sec. 5.3) to investigate whether they satisfy the expected functionality to generate fashion item images. We also evaluated the overall garment generation quality (Sec. 5.4).

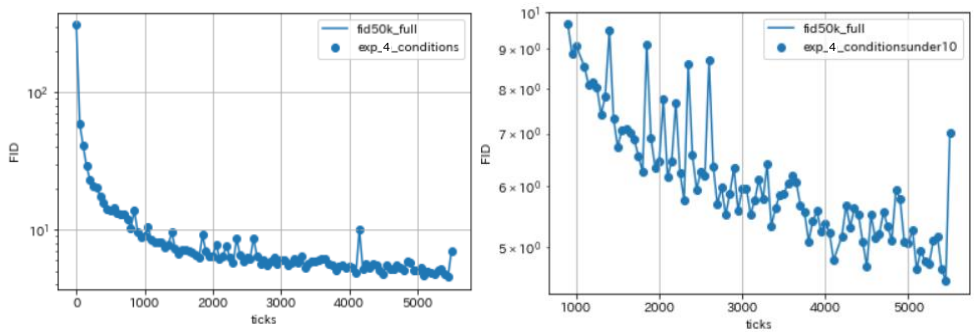
5.1 Quality of Multiple Conditional StyleGAN Generation

First, we trained the conditional StyleGAN2-ADA [2] module conditioned by the combination of four condition labels (category, silhouette, length, color) and evaluated the quality of the generated images both quantitatively and qualitatively.

For the quantitative evaluation, we compute the Fréchet Inception Distance (FID) [20] which measures the proximity between the distribution of generated images and real images. The result is listed in Table. 1 and the training curve is depicted in Figure. 3, where the further the training proceeds, the lower the FID becomes, i.e., the distribution of generated garment images is close to the real clothing image distribution. Figure. 4 shows sample generated images, where we can see that the majority of the generated clothing images consistently reflect the input combination of the four conditions. Moreover, the generated images with the identical combination of the four conditions in Figure. 4 differ from each other, which indicates that the model achieves diverse generation. Both the quantitative and qualitative evaluations demonstrate that the first StyleGAN2-ADA module can generate diverse fashion item images consistent with the given conditions.

Table 1: FID score of conditional StyleGAN2-ADA trained on AirCloset dataset.

	FID score
Conditional StyleGAN2-ADA	4.53



(a) : Overall Training Curve (b) : Final train curve

Figure 3: Training curves for AirCloset dataset.

Training details: We trained our conditional StyleGAN2-ADA module from 62k garment images with four condition labels for each image (provided by AirCloset.inc) using one NVIDIA RTX A6000 GPU for ten days. We used Adam [21] (learning rate:0.0025, $\beta_0 = 0$, $\beta_1 = 0.99$, $\varepsilon = 1e-8$) to optimize the model.

5.2 Quality of Clothing Texture Synthesis

Next, we evaluated the effectiveness of the second swapping autoencoder [3] module by qualitatively comparing it with conditional StyleGAN2-ADA module. As discussed in Sec. 4.1, since we modified the StyleGAN2-ADA module to be able to deal with an arbitrary number of conditions, we trained the StyleGAN2-ADA module with a combination of five conditions (category, silhouette, length, color, and clothing pattern) to compare it with the swapping autoencoder module. The result of this comparison is shown in Figure. 5. As we can see, compared with the swapping autoencoder, the conditional StyleGAN2-ADA cannot apply the given clothing pattern label in its generated fashion images. Moreover, the

Figure 4: StyleGAN2 generated images conditioned by 4 labels.

Figure 5: Qualitative comparison of swapping autoencoder and StyleGAN2-ADA.

sleeves of the garment image generated by the conditional StyleGAN2-ADA (bottom right of Figure. 5(b)) are collapsed. In contrast, as shown in Figure. 5(a), swapping autoencoder can synthesize the clothing pattern while preserving both the structure and the texture. This result demonstrate that the swapping autoencoder outperforms the conditional StyleGAN2-ADA in terms of applying the clothing pattern to the generated garment image.

Training details: As in the first experiment (Sec. 5.1), we trained the conditional StyleGAN2-ADA model on a 62k clothing image dataset with a combination of five condition labels for each image using one NVIDIA RTX A6000 GPU for ten days. We optimized the conditional StyleGAN2-ADA by Adam [21](learning rate:0.0025, $\beta_0 = 0$, $\beta_1 = 0.99$, $\varepsilon = 1e-8$). As the swapping autoencoder, we trained it for one week using one NVIDIA RTX A6000 GPU and optimized it by Adam [21](learning rate : 0.002, $\beta_0 = 0$, $\beta_1 = 0.99$, $\varepsilon = 1e-8$).

Figure 6: Results of applying SeFa on swapping autoencoder output images.

Figure 7: Comparison of using different CNN weights for SeFa image editing.

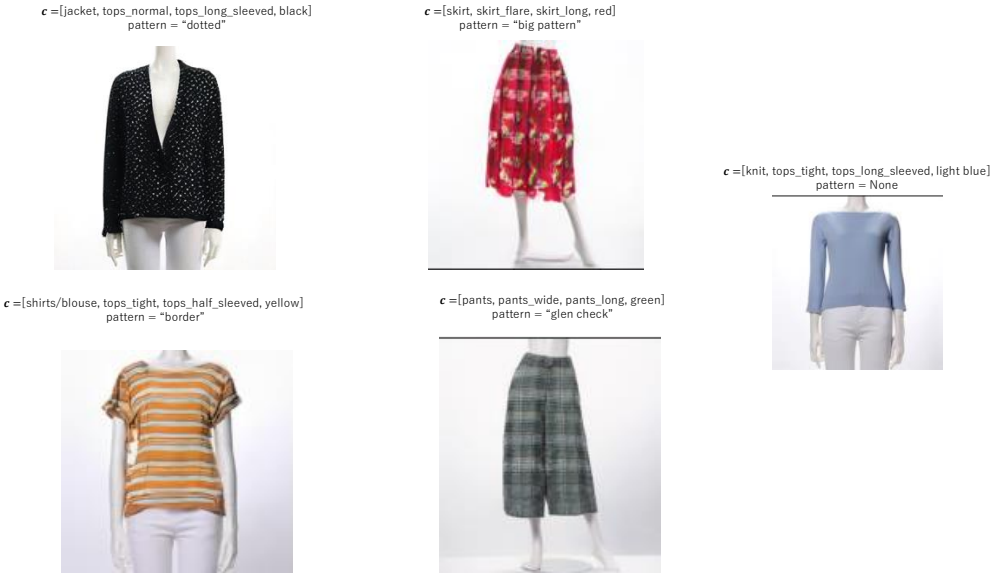


Figure 8: Overall quality of garment generation.

5.3 Quality of SeFa Color Editing

We next investigated the effectiveness of applying SeFa to the weight of CNN layers in the ToRGB block $\mathbf{A}_{ToRGB}$ of the swapping autoencoder generator. Figure. 6 shows the result of image editing by distilling five interpretable directions from the eigenvectors of $\mathbf{A}_{ToRGB}^{\top}\mathbf{A}_{ToRGB}$ where $\alpha_i (i \in [1,5])$ means the intensity of each optimal directions $\boldsymbol{n}_i^*$ in Equation. (12). These results demonstrate that we can edit the image color by manually adapting α_i while maintaining the structure and the texture of the generated clothing images thanks to the second module of our system.

Also, to determine the effectiveness of only using the weights of the ToRGB block for SeFa, we applied SeFa using the weight of all CNN layers in the swapping autoencoder generator. The results are shown in Figure. 7, where we can see in the middle row that, SeFa using weights of all the CNN layers and adapting α_i collapses the structure and the texture of the input generated images. On the other hand, only using the weights of the CNN layers at the ToRGB block of the Swapping Autoencoder generator (bottom of Figure. 7), can successfully edit only the color of the input generated images.

These results demonstrate that, when editing only the color of the input image using SeFa on the generator of swapping autoencoder, it is more effective to make use of the weights of the CNN layers in the ToRGB block.

5.4 Qualitative evaluation of the overall architecture

Finally, we qualitatively evaluated the capability of our proposed system to generate garment images by examining the output garment images shown in Figure. 8. In the clothing image on the bottom left of Figure. 8, we can clearly see a picture of yellow half-sleeved striped shirts, and on the bottom middle of Figure. 8, we see green checked wide-leg pants, both of which are consistent with the input condition labels. Therefore, we conclude that our proposed system can accurately reflect detailed conditions such as complex clothing

patterns, lengths, and silhouettes in its generated images.

6 Conclusion

In this paper, we propose a GAN-based fashion design generation system to reduce the workload of the labor-intensive novel fashion design creation. The experimental results and the sample images generated by our system demonstrate that it can accurately generate garment images consistent with the input conditions. However, in the real world, with its proliferation of photographs, characters, logos, etc., the garment patterns are too diverse for all of them to be reflected as the texture of the generated images. Therefore, our future work will focus on ways of improving generative models to that point that they are capable of generating diverse clothing images with complex patterns.

References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[2] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12104–12114. Curran Associates, Inc., 2020.

[3] Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7198–7211. Curran Associates, Inc., 2020.

[4] Yujun Shen and Bolei Zhou. Closed-form factorization of latent semantics in gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1532–1540, June 2021.

[5] Xintong Han, Zuxuan Wu, Zhe Wu, Ruichi Yu, and Larry S. Davis. VITON: an image-based virtual try-on network. *CoRR*, abs/1711.08447, 2017.

[6] Wang-Cheng Kang, Chen Fang, Zhaowen Wang, and Julian J. McAuley. Visually-aware fashion recommendation and design with generative image models. *CoRR*, abs/1711.02231, 2017.

[7] Sudhir Kumar and Mithun Das Gupta. c^{+}gan: Complementary fashion item recommendation. *CoRR*, abs/1906.05596, 2019.

[8] Lele Chen, Justin Tian, Guo Li, Cheng-Haw Wu, Erh-Kan King, Kuan-Ting Chen, Shao-Hang Hsieh, and Chenliang Xu. Tailorgan: Making user-defined fashion designs. *CoRR*, abs/2001.06427, 2020.

[9] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[10] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

[11] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[12] Kathleen M. Lewis, Srivatsan Varadharajan, and Ira Kemelmacher-Shlizerman. VOGUE: try-on by stylegan interpolation optimization. *CoRR*, abs/2101.02285, 2021.

[13] Gökhan Yildirim, Nikolay Jetchev, Roland Vollgraf, and Urs Bergmann. Generating high-resolution fashion model images wearing custom outfits. *CoRR*, abs/1908.08847, 2019.

[14] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[15] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[16] Wenqi Xian, Patsorn Sangkloy, Varun Agrawal, Amit Raj, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Texturegan: Controlling deep image synthesis with texture patches. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[17] Edo Collins, Raja Bala, Bob Price, and Sabine Susstrunk. Editing in style: Uncovering the local semantics of gans. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[18] Kurt Hornik, Ingo Feinerer, Martin Kober, and Christian Buchta. Spherical k-means clustering. *Journal of Statistical Software*, 50(10):1–22, 2012.

[19] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable GAN controls. *CoRR*, abs/2004.02546, 2020.

[20] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.