

Fashion Designer Application Based on Generative Adversarial Network

Ziming Liu *

Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, China

* Corresponding Author Email: zliuda@connect.ust.hk

Abstract. Embodying the design from scratches has been a common problem for all designers. In some situation, clothes designers have trouble creating works that match their idea the most. Creating a fine work from a simple sketch is a tedious process of endless try-and-error. Designers spend days, weeks, or even months drawing drafts and discard them, because they need to draw the picture that best embodies the idea in their head. This study proposes a deep learning method that can generate clothes images from human sketch input, explores the effect of different image to sketch translators, different Generative Adversarial Network (GAN) architecture, loss functions and regularization methods. Many previous works have demonstrated strong results on conditioned image generation, and this study shows that similar settings can be transferred to help designers to embody their designs. The result shows that the model can generate clothes images with high fidelity while faithful to the input sketch.

Keywords: GAN, Cross-domain adversarial loss, image translation, conditioned image generation

1. Introduction

GAN can be used for generating various kinds of artworks, for example, fashion designs. The goal is to develop a GAN model that can generate clothes image from a few human sketches. Such model is of great value for designers, which is providing visualization for their design idea according to the rough sketches they drew. It is not a rare situation that designers draw dozens of sketches but screwed it, since they do not match their idea. Providing visualizations based on their sketches will give them great inspirations and reduce a lot of designing time.

In this project, the goal is to solve questions above by utilizing GAN to build an architecture that produces designer clothing image from human's simple sketches, involving cross-domain loss between sketches and real images. In the end, the model can generate designer clothes from sketches with high fidelity to the original image dataset and resembles the sketches at the same time.

2. Related works

2.1. Generating fashion clothes image

Previous investigation on fashion clothes image generation can obtain favorable result even using very simple network architecture. Deepstyle [1] is a typical example of utilizing simple network architecture and generated decent fashion image. It uses a simple DCGAN model with 5 transpose convolutional layers in the generator and 5 convolutional layers in the discriminator, and the result is good enough. The dataset used in Deepstyle are pictures from fashion books, cropped after using object detection to retrieve only the clothes. However, the dataset contains a lot of noise in the background, which lowers the image quality. The generated pictures are of random styles, but the user of these images and the designers often want constrained clothes style that is close to their designing idea. Random styled clothes will provide no benefit but interference to their inspiration. Therefore, the goal of this study is to generate clothes images conditioned on the request of the designer, moreover, reduce the noise and display only the clothes as much as possible.

2.2. The determination of the number of network layers

Fashion GAN[2] has achieved remarkable result on generating fashion clothes with condition. The objective is to generate different clothes on the model in the given image, with the constraint from the input sentence. During train time, it first inputs a clothes image of a person and a sentence description of a new desired outfit, and the first GAN can generate a segmentation map S and discriminate it with the real segmentation map of the input image. The second GAN renders the new image with the guidance from the segmentation map generated in the previous step. This way of generating different image style could reach the great result, however, semantic input cannot carry the idea of the designer as accurate as sketch. Therefore, this study will generate clothes images that can meet the idea of the designer as much as possible.

In Figure 1, it shows that given an input clothes image of a person and a sentence description of a new desired outfit, the model first generates a segmentation map S using the generator from the first GAN. It renders the new image with another GAN with the guidance from the segmentation map generated in the previous step.

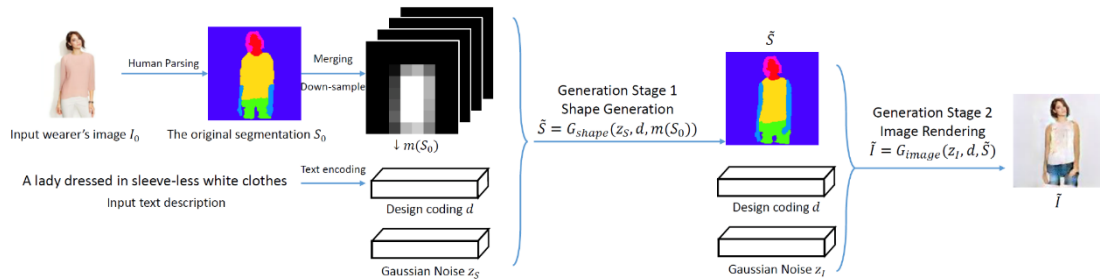


Figure 1. Fashion GAN framework

2.3. Training a mapping from edge to photo

Pix2pix [3] is one of the most advanced application of generating image from scratches, with a minor difference that it starts with edges of photos rather than human sketches. The idea is to use conditional to construct an edge to photo mapping. The data is presented in pairs of {edge, photo} tuples. The tuples of edge and the generated photo are discriminated as fake, and the tuples of edge and real photo are identified as real. This paper has achieved the result on generating photo from edge map. However, this is not applicable in our project. The first reason is that edge map and sketch are different in terms of the number of details of the object. The edge maps in this study can portray the object with too many details than human sketches would do. This study uses the edge map image to sketch (or edge) translator that can best imitate hand-drawn sketches. The second reason is that in real situation it is not likely to have thousands of sketches for training. In this architecture, every photo needs to pair with an edge map. When Pix2pix is applied to hand-drawn sketches, pairing each real photo with a hand-drawn sketch is unrealistic. A handful of hand-drawn sketches is needed to generate large amount of photo that is close to the sketches.

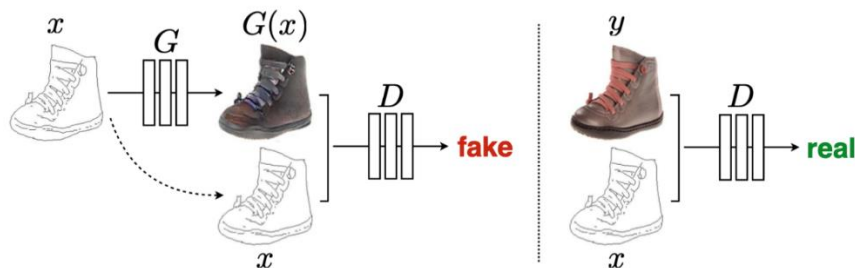


Figure 2. Conditional GAN in Pix2pix

In Figure 2, it trains a conditional GAN to map from edge to photo. The discriminator, D can classify between generated fake and real {edge, photo} tuples. The generator G can learn to fool the

discriminator. The difference between conditional and unconditional GAN is that both the generator and discriminator can observe the input edge map

2.4. Cross domain adversarial loss

One of the main questions in our project is how to relate a sketch and real photos together. Traditionally, training loss must be constructed between data that are from the same domain, but sketches and real photos differ a lot in contents. This study applies cross domain adversarial loss, proposed in Sketch Your Own GAN [4]. As shown in Figure 4, the network first generates a fake image, and then it will go through two discriminators: the first one distinguishes between the sketch image translated by an image-sketch translator F and the hand-drawn sketch, and the second one distinguishes the generated image and real image. The sketch discriminator aims to update the generator so that the shape of the generated image matches the sketch, while the image discriminator aims to preserve the image quality, i.e., closer to the real image in the dataset. Sketch GAN has achieved very significant result on sketch to image generation, but the network it uses requires very expensive computational resources. The images are of high resolution and the pretrained generator is StyleGAN2 [5] which needs weeks to do pretrain. Therefore, this study only uses the cross domain adversarial loss for reference without its network architecture.

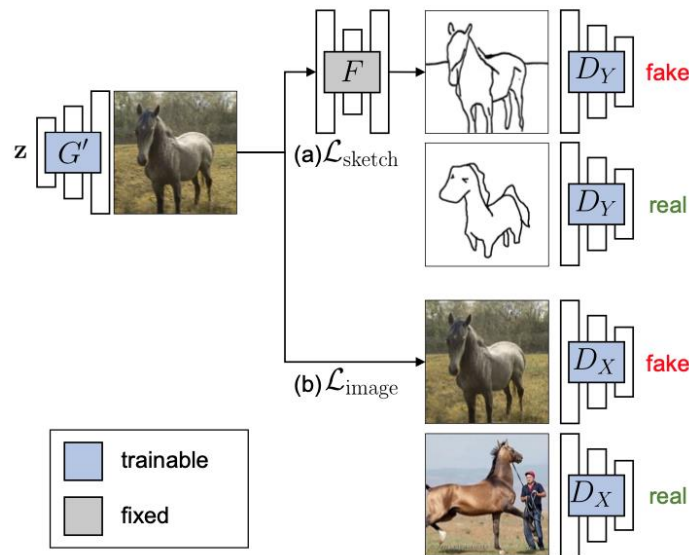


Figure 3. Sketch GAN -- Cross domain adversarial loss

Figure 3 shows the training of cross domain adversarial loss consists of two major components. (a) L_{sketch} is the sketch discriminator D_Y that can classify between the fake and user sketches. A pre-trained mapping network F is used to translate the output of our model $G(z; \theta')$ to a fake sketch. (b) L_{image} is the image discriminator D_X that can classify between the fake and real images. Real images are sampled from the training set of the original model $G(z; \theta)$.

3. Data

The data used in this study is from Deep Fashion[6], and Fashion Synthesis Benchmark. It consists of 78,979 images of fashion models, associated with segmentation. This study used first 75000 images in the first pretrain step and reserved the remaining images as unseen data from which I selected images as input and evaluation set. This study changed the background of the images from black to white, so that the aliasing problem in the segmentation mask can be alleviated to some extent. Since our input hand-drawn sketch will be the whole human figure, this study preserves all the human segmentation instead of only using the clothes segmentation.

4. Methods

4.1. Training procedure

To generate clothes image from the single or few sketches, we will use the training procedure described in Figure 4. The architecture uses the Cross-Domain Adversarial learning proposed by Zhu et al. in Sketch Your Own GAN as backbone [4]. This study first uses a pretrained generator to generate a fake image and produces a sketch from it. The first discriminator D_{sketch} can distinguish the real input human sketch and the generated fake sketch. The second discriminator can discriminate the generated fake image and the real image from the dataset. In this way we can guarantee that the shape of the generated images matches the sketch, while preserving the image quality and fidelity.

Figure 5 shows the simple workflow of the GAN. We first generate a fake image from noise using a generator, then go through a sketch filter to produce a fake sketch. The first discriminator D_{sketch} can discriminate real human sketch input from the sketch of the generated fake image. The second discriminator D_{image} can discriminate real image in the dataset from the generated fake image.

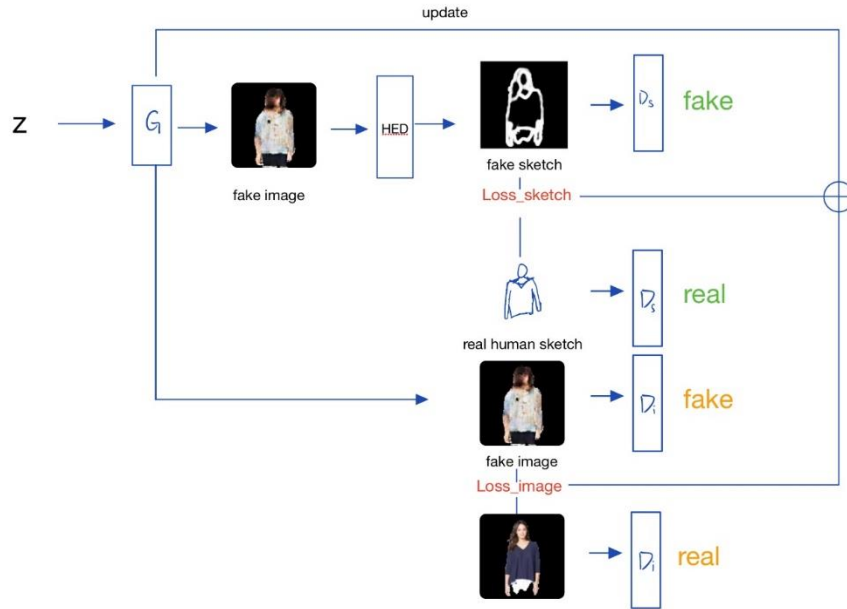


Figure 4. Training procedure

4.2. Objective

$x \sim p_{data}(x)$ denotes the set of real training images, $y \sim p_{data}(y)$ denotes the set of human sketches, and z is noise. F denotes the translator that converts the image to sketch, in this case it is HED. The binary cross entropy loss of the sketch part is as Equation 1:

$$L_{sketch} = E_{y \sim p_{data}(y)} \log(D_{sketch}(y)) + E_{z \sim p_{data}(z)} \log\left(1 - \left(D_{sketch}\left(F(G(z))\right)\right)\right) \quad (1)$$

Equation 1 means that the generator is trained to generate images that can be translated to the sketch that most approximate the input sketch. However, a single discriminator that guarantee the similarity of the generated sketch is not enough. The ultimate output is the generated image, so a regularization loss on the authenticity of the generated image is needed, as shown in Equation 2:

$$L_{image} = E_{x \sim p_{data}(x)} \log(D_{image}(y)) + E_{z \sim p_{data}(z)} \log\left(1 - D_{image}(G(z))\right) \quad (2)$$

Therefore, the total loss is shown as Equation 3:

$$L_{total} = L_{sketch} + \lambda L_{image} \quad (3)$$

Where the λ is the regularization strength on the authenticity of generated image and suggested by Zhu et al (4) in their paper, it should be 0.7. This parameter will be further adjusted since our generated

image need fidelity. Based on the pretrained generator $G(z; \theta)$, where θ is the pretrained weight, we want to learn a new set of weight $G(z; \theta')$ with the following minimax objective, as presented in Equation 4:

$$\theta' = \arg \min_{\theta'} \max_{D_{\text{sketch}}, D_{\text{image}}} L_{\text{total}} \quad (4)$$

4.3. Network Architecture

W-DCGAN with gradient penalty is adopted to do pretrain on the image data, in this way we can get a generator that can generate any clothes images at the beginning of the training procedures. In Sketch Your Own GAN, StyleGAN2 was used as the pretrained generator, but it requires much more computational resources than we can afford: training 25,000 images of resolution 256*256 requires more than 32 days with 1 GPU. It can be predicted that if we are able to use StyleGAN2 as our generator, the image quality will be significantly improved.

For the image to sketch translator, we tried the pretrained Photosketch (7), Canny Edge Detector and Holistically-Nested Edge Detection (HED) (8), and we found that HED works best during training which can detect edges thus depicts the clothes. The results of different image-sketch translators are shown in Figure 5.

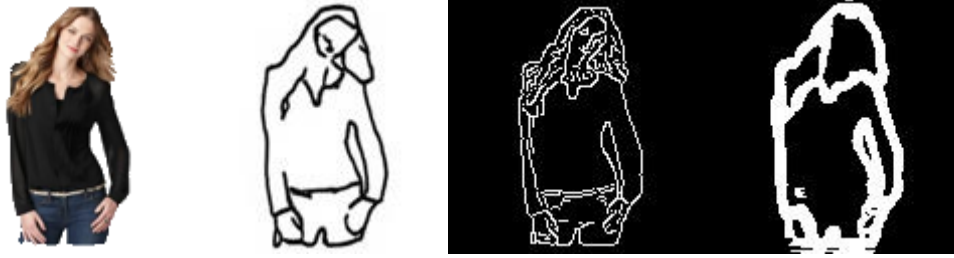


Figure 5. Different image-sketch translator: From left to right: original, Photosketch, Canny edge detector, HED.

While Photosketch has similar result as HED, its architecture is more complex resulting in a longer transforming time. Canny edge detector gives too many edge details compared to hand-drawn sketch. Overall, HED gives best image-sketch translation. It approximates hand-drawn sketch better than Photosketch and Canny edge detector.

5. Experiments and Results

5.1. Pretrained GAN

First step is to obtain a pretrained GAN to generate clothes images. We first tried to build up a pipeline to train StyleGAN2. As reported before, we observed the significant training difficulty in our experiments. On a single RTX2070 GPU, training a mini-batch consisting of two images would take more than 1 minute. We decided to give up StyleGAN2 and return to our previously implemented DCGAN.

In our previous work, we implemented a standard DCGAN model. However, a set of carefully chosen hyper-parameters is needed to keep a stable training process. In addition, the Sigmoid in the last layer of Discriminator also kills most of the gradients and causes a long training time. As an improvement to our previous work, we refer to the WGAN-GP [9]. We removed all Batchnorm layers and the Sigmoid layer in the Discriminator. And instead of binary cross entropy loss, we use Wasserstein distance with gradient penalty as loss function, which improves the gradient flow and leads to a faster convergence and better image quality. Also, the model does not suffer mode collapse that we have observed in previous implementation. The overall loss for the discriminator is:

$$L_D = 1/N \sum_{n=1}^N (D(G(z_n)) - D(x_n) + \lambda (\|\nabla D(\epsilon x_n + (1 - \epsilon)G(z_n))\|_2 - 1)^2) \quad (5)$$

where $z \sim N(0,1)$, $x \sim P_{\text{real}}$, $\epsilon \sim U(0,1)$.

The loss for the generator is:

$$L_G = \frac{1}{N} \sum_{n=1}^N -D(G(z_n)) \quad (6)$$

To quantitatively evaluate the pretrained model, we use cleanFID [10] package to measure the Frechet Inception Distance(FID) score. At iteration 84600, the model reached its minimum FID score 25.805. We use this for future sketch modification. Table 1 and Figure 6 describes the FID and training result using BCE loss, table 2 and Figure 7 describes the FID and training result using WGP loss.

Table 1. The FID was calculated with 12800 images from the training dataset

Iteration	iter11200	iter27200	iter41600
FID	121.3	227.484	271.886

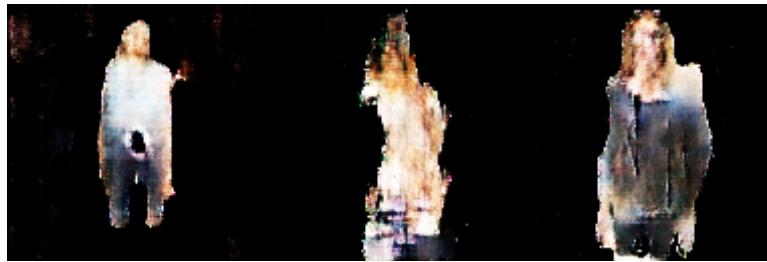


Figure 6. BCE Loss

Figure 6 shows the generated images at iteration 11200, 27200, 41600 respectively. The model collapsed at around iteration 30000, so we stopped training earlier.

Table 2. The FID was calculated with 12800 images from the training dataset.

Iteration	iter78500	iter84600	iter86600
FID	36.249	25.805	28.165



Figure 7. WGP Loss

Figure 7 shows the generated images at iteration 78500, 84600, 86600 respectively.

5.2. Training with sketches

5 images were selected from the DeepFashion data set that have similar human poses and clothes shapes, as shown in Figure 8, then transform them into sketches using HED as input sketches. Additional 300 similar images were hand-selected to serve as evaluation data set.



Figure 8. Input sketches

First approach was to train with sketch discriminator with only $\lambda = 0$. Using BCE as loss, the model degenerated significantly within 1000 iterations. This indicated a failure in building up an adversarial

learning process. Our analysis is that in the early stage of training when discriminator is not well trained, it provides meaningless gradients to the generator, however, due to the small sketch data set, the discriminator soon converges and provides no gradients. Model using WGP as loss also suffered degeneration problem but was less severe than using BCE. Also, the generated samples showed a successful sketch matching because generator displayed a preference in generating upper-body standing woman images and a relatively lower FID score. The result is demonstrated in Figure 9.



Figure 9. Left: BCE. Right: WGP. Generated results with $\lambda = 0$.

Second experiments applied sketch and image discriminator with $\lambda = 0.7$ as suggested in SketchGAN[4]. Using BCE for both discriminators also showed a poor image quality and failure in matching with the input sketches. When training with WGP loss for both discriminators, we noticed that a circle of fake contour occurred around the generated images causing the sketching network to generate more matching sketches though the actual generated images did not match with the inputs. On observing this, we decided to increase the λ factor in following experiments. The result is demonstrated in Figure 10.

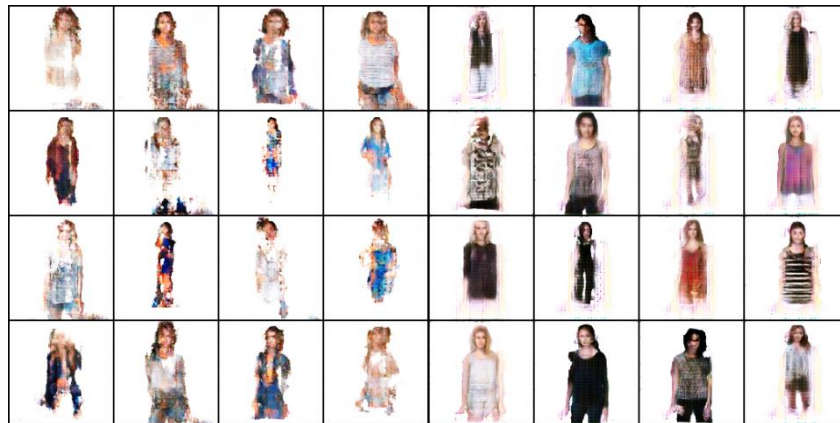


Figure 10. Left: BCE. Right: WGP. Generated results with $\lambda = 0.7$

In the third experiments, λ is increased to 3.0 and trained with WGP loss, which gave the best result. As an alternative approach, we also tried to use BCE loss for the sketch discriminator and WGP loss for the image discriminator, where the loss for the generator became:

$$L_G = 1/N \sum_{n=1}^N (-\log(D_{\text{sketch}}(G(z_n)) - D_{\text{image}}(z_n))) \quad (7)$$

However, the result images showed unexpected strips on the left side and a higher FID score. The result is demonstrated in Figure 11 and table 3.



Figure 11. Left: WGP+WGP. Right: BCE+WGP. Generated results with $\lambda = 3.0$

Table 3. Numeric Result under Training Configurations.

λ	Sketch Loss Fn	Image Loss Fn	FID
$\lambda = 0$	bce	NA	313.392
	wgp	NA	163.480
$\lambda = 0.7$	bce	bce	181.525
	wgp	wgp	136.506
$\lambda = 3.0$	wgp	wgp	92.933
	bce	wgp	118.185
Pretrained	NA	NA	134.382

Comparing the generated images before and after training with sketches, we observed that the overall image quality would slightly decrease however the generator would be more likely to produce images with similar shape as sketches than before. The FID scores of second step are higher than the ones in the pretrain step. The major reason is the size of evaluation data set is different. It is also obvious that the generator after the second stage training results in a lower FID score on the selected images, which indicates a successful sketch learning. The result is shown in Figure 12.



Figure 12. Comparison. Left: Before. Right: After.

6. Conclusion

The main objective of this project is to generate fashion clothes image with a few hand-drawn sketches as input. To achieve this, we utilized cross-domain adversarial loss to build connection between sketches and images, image-sketch translators to translate image into sketches during training and built a W-DCGAN with gradient penalty to make the network faster for training. We

also learned how to tune the GAN network when facing gradient vanishing and explosion, as well as modifying the data parameters under different situations to fix abnormal behavior of generated results. After reading a dozen of papers on different types of GANs, not only did we study the popular network architecture and training procedures of state-of-the-arts GANs, but also learned the lesson that state-of-the-arts GANs often requires huge computational resource than other types of deep learning framework, so we need careful examination before applying GAN to solve deep learning problems.

This project can be furthered applied to larger image datasets with higher image resolution, as well as using better pretrained generator such as StyleGAN2 to produce higher quality fashion clothes images with higher quality, provided with enough computational resource. Also, we can extend the design of clothes to more fields, such as designing cars and buildings, or even generating masterpiece drawings from hand-drawn sketches. We may also apply texture to further improve the network so that we can generate more realistic texture patterns on the clothes. We can foresee a wide range of application of this project.

References

- [1] Cheng, R. (2020). Generative Adversarial Network with Object Localization for Fashion Trend Prediction and Style Inspiration. <https://towardsdatascience.com/deepstyle-f8557ab9e7b>.
- [2] Zhu, S., Urtasun, R., Fidler, S., Lin, D., & Change Loy, C. (2017). Be your own prada: Fashion synthesis with structural coherence. In Proceedings of the IEEE international conference on computer vision (pp. 1680-1688).
- [3] Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1125-1134).
- [4] Wang, S. Y., Bau, D., & Zhu, J. Y. (2021). Sketch your own gan. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 14050-14060).
- [5] Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., & Aila, T. (2020). Analyzing and improving the image quality of stylegan. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 8110-8119).
- [6] Liu, Z., Luo, P., Qiu, S., Wang, X., & Tang, X. (2016). Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1096-1104).
- [7] Li, M., Lin, Z., Mech, R., Yumer, E., & Ramanan, D. (2019, January). Photo-sketching: Inferring contour drawings from images. In 2019 IEEE Winter Conference on Applications of Computer Vision (WACV) (pp. 1403-1412). IEEE.
- [8] Xie, S., & Tu, Z. (2015). Holistically-nested edge detection. In Proceedings of the IEEE international conference on computer vision (pp. 1395-1403).
- [9] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. C. (2017). Improved training of wasserstein gans. Advances in neural information processing systems, 30.
- [10] Parmar, G., Zhang, R., & Zhu, J. Y. (2021). On buggy resizing libraries and surprising subtleties in fid calculation. arXiv preprint arXiv:2104.11222.