Hybrid Generative Adversarial Network

Wu Jiahao
Sun Yat-sen University
18364094

Zhan Dandan
Sun Yat-sen University
18364113

Abstract

We do a preliminary exploration on the possibility of combining a series of generative adversarial networks to achieve good performance. Our aim is to construct a hybrid model that can be robust enough to generate images that look realistic, and we use shoes as our dataset. To meet this goal, we also try different forms of tensor flows, different loss function, different training tricks and finally adopt the combination that works best.

1. Introduction

1.1 The Shoe Industry

With the development of economy, the demand of shoes is greater than ever before. How to design shoes that can meet the needs of people has become one of the key problems. So we want to find a hybrid network to generate shoes, hoping that they can bring the shoes designers some inspirations and therefore speed up the process of shoe design.

1.2 Procedural Image Generation

Recently, due to the rapid development of computer vision, procedural image generation has made great progress. Many generative models have been proposed and quite a few of them have been very successful at generating images. Now it is possible for computers to generate images that look realistic.

1.3 Our Hybrid Generative Adversarial Network

In this paper, we propose a universal GAN, which can generate images according to the required condition. GAN [1] is one of the powerful models that work well in generating images. Our work is based on this model. If we want the generator to generate an image of the specified type, we need to be able to tell the generator the type we want. We can do this with a Conditional GAN [3]. On the other hand, we can encourage our model to learn some interpretable and meaningful representations with an InfoGAN [2]. As a result, we decide to combine these two models, that is, we adopt the input format of the generator of InfoGAN [2] which include a continuous input noise vector z, the label of the target image as well as the input latent code, and the input format of the discriminator of CGAN [3] which include an image and a label. We also introduce self-attention mechanism [8] and variation tricks [7] into our model and use WGAN-GP loss [4] as our GAN loss.

Our contributions are as follows:

- We propose a universal hybrid GAN, which can generate images according to the required condition and perform well.
- We apply our model on our sorted shoes dataset and this can provide inspirations for shoes designers.
- By experiments, we prove the possibility of combining several models into a hybrid model to achieve good performance.

2. Related Work

2.1 Generative Adversarial Networks

In 2014, Goodfellow et. al proposed a system for generating images based on a Generator network, G, and a Discriminator network, D. At each training iteration, G will generate a set of images. These images will be sent to D. What D should do is to determine whether each of G's images are real or not. In order not to be recognized by D, G should try to generate the images as realistic as possible. On the other hand, D should improve its ability to tell real from fake so that it won't be fooled by G. As both G and D train against each other, the generated images should become more and more realistic. But at that time, GAN was fully connected, and it was easy to crash. The images it generated were low-resolution, slightly distorted and of poor quality. Even so, it did open a new era, and since then various improved GAN versions have emerged. GAN has become more and more powerful, and has been applied in more and more fields.

2.2 Conditional Generative Adversarial Nets

In 2014, Conditional Generative Adversarial Nets [3], a functional extension of GAN, was proposed. It can

control the categories of the generated images. The meaning of CGAN is that it can use one model and one mixed data set to produce many kinds of images, that is, it can complete the functions of multiple models. In order to enable our model to generate multiple categories of shoes, we also provide condition for our model, where both G and D have condition as part of the input. However, when CGAN was proposed, it was fully connected and could only generate low-resolution images. A lot of parameters will be needed if we want to generate high-resolution images. Therefore, we do not use its backbone but use convolutional architecture instead.

2.3 Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

In 2015, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks [6] was proposed, which applied convolution neural network to GAN. The quality of images generated by DCGAN has improved and, more importantly, the sharing of parameters in convolutional neural networks makes the GAN model lighter and thus produces higher resolution images. Experiments also found some explainable property of the latent code. So we adopt the structure of convolution. But so far the quality of the images is still not good enough, the training is still not stable enough, and the control of the relationship between input and output is still at a preliminary stage of experiment and observation.

2.4 InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

In 2016, the proposed InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets [2] took an important step towards exploring the relationship between the input and output of GAN. It offers an information code to the original input of generator and hopes that discriminator can restore the code from the image generated by generator. In order to achieve this goal, the discriminator should supervise the generator to encode the information code into the image as a visual information so that the discriminator can decode this visual information into original code.

2.5 Generative Adversarial Text to Image Synthesis

In 2016, Generative Adversarial Text to Image Synthesis [5] was proposed, too. Generative Adversarial Text to Image Synthesis is a generalization of CGAN in convolution neural network. It explored the possibility of CGAN and found that CGAN did adapt to various types of condition input including language embedding. It extends the condition input to the same size of images after embedding and then concatenate it into the feature-map. We also use this trick. But the quality of the generated images is not good enough, and the training is unstable. We need to find some suitable tricks to improve the quality of the generated images and the robustness of our model.

2.6 Improved Training of Wasserstein GANs

In 2017, Improved Training of Wasserstein GANs [4] was proposed. And it greatly improves the quality of the generated images and enable the training to be stable. So we adopt the loss function of it.

2.7 Progressive Growing of GANs for Improved Quality, Stability, and Variation

In 2017, Progressive Growing of GANS For Improved Quality, Stability, And Variation [7] was proposed. It proposed a novel training way to generate high resolution images progressively. Besides a variety of tricks were also introduced, among which we introduce one called minibatch standard deviation to avoid the collapse of our model.

2.8 Self-Attention Generative Adversarial Networks

In 2018, Self-Attention Generative Adversarial Networks [8] was proposed. By introducing attention mechanism into G and D, it solves the problem that the images generated by G are too smooth and the texture is not clear, thus greatly improving the ability of G to generate detail texture. By experiments we find that this mechanism is indeed beneficial to improve the performance of our model, so we add it to our discriminator.

2.9 Dataset

Our dataset is the UT-Zap50K dataset, which was collected by researchers at the University of Texas. This dataset contains 50025 images which were divided into 4 major categories – “boots”, “sandals”, “slippers” and “shoes”. Each category also includes different types of shoes. And we modify this dataset according to our requirement.

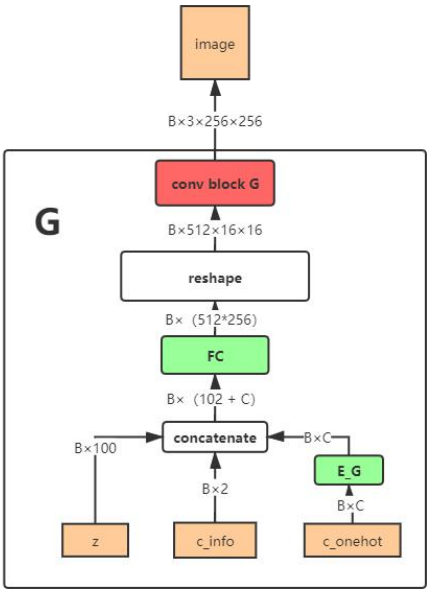


Figure 1.1. Generator structure. Our generator has 3 inputs and one output. The FC layer is a simply matrix multiply operation without activation function.

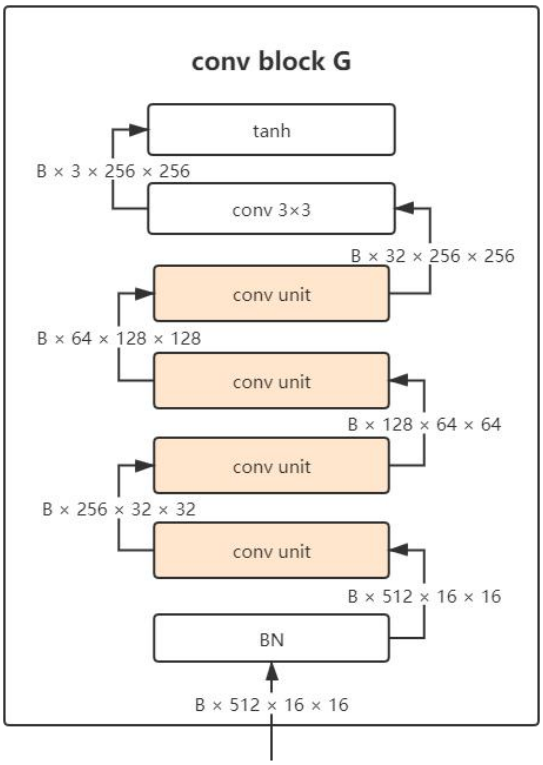


Figure 1.2. Conv block of Generator. We do not use fraction stride convolution to upsample but use nearest neighbor interpolation instead. The architecture of *conv unit* in conv block G is shown in Figure 1.3.

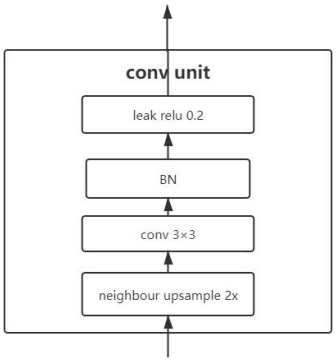


Figure 1.3. Conv unit

3. Hybrid-GAN

We propose a universal GAN for generating images fitting the required condition. It takes in several valuable ideas from several GAN models so we called it Hybrid-GAN. The structure of our final model is showed in [Figure 1.1. to 1.5.]. Experiments on shoes dataset proved that it is highly robust, light, fast, while handling with hard tasks well.

3.1 condition fitting

w.r.t. CGAN [3], in order to force the images generated to obey different given rules with only one generator, the network should be formed to receive the condition input and provide some supervision. We think of the ability to obey different conditions simultaneously

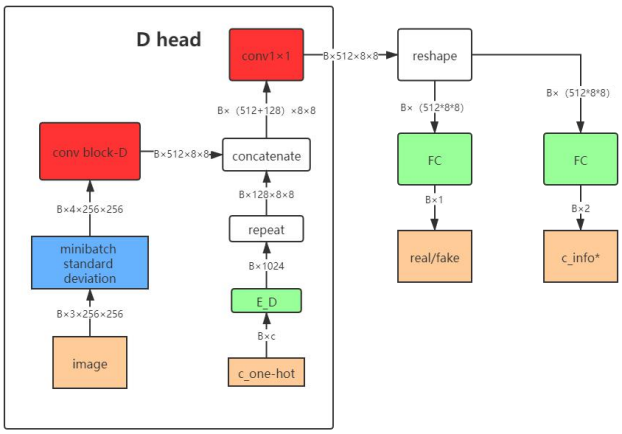


Figure 1.4. Discriminator structure. The *conv 1×1* denotes a pile of per pixel convolution, leakyRelu and dropout. The *minibatch standard deviation* operation are introduced from [7] without any change so we won't talk more about it. The two *FC* layer are simply matrix multiply without activation function. The *E_D* function denotes a two-layer fully connected function. *Conv block D is shown in* Figure 1.5.

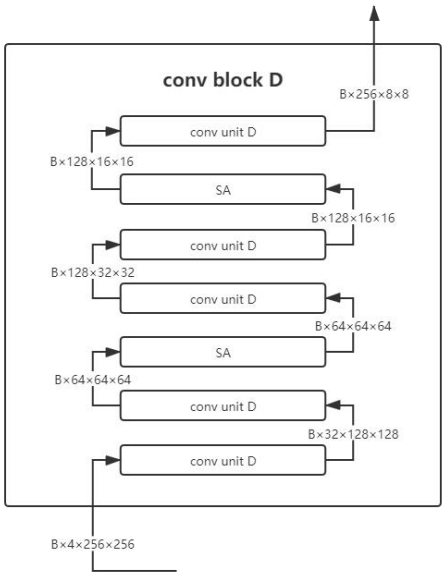


Figure 1.5. Discriminator conv block. Expect for the first unit, each conv unit D consists of a 3×3 convolution layer with strides 2, a leakyRelu activation, a dropout layer and a batch normalization layer. The difference between the first unit and other units is that the first unit doesn't have a BN layer. The SA layer is introduced from [8] without change, so no more description is needed.

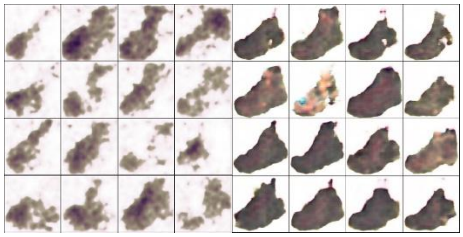
as the ability to handle with multitasking, which means, the generator should try to handle several different distributions at one time with the same parameters. But different classes will interfere with each other when training with the same parameters. The result is that even though the condition supervision is added, some images generated may look like "hybrid kinds" or may remain some features that do not belong to the class assigned, like the images showed in [Figure 7. (a)]. It's a tough job to generate completely different images using the same parameters. And the more different the distributions are, the harder the multitasking is.

To test our model's ability to handle with hard multitask, we use a shoes dataset called UT-Zap50K. We remove some images that do not meet our requirements, and divide the rest into 12 categories [Figure 2.]. Each category differs from each other greatly while holding some similarities as well. Therefore, it is hard but acceptable for the generator to train on such dataset.

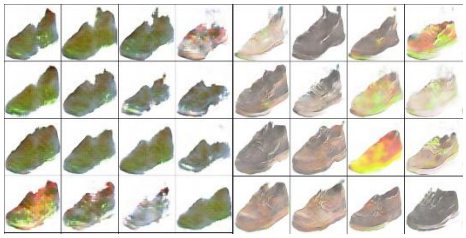
Figure 2. 12 kinds of shoes in dataset1, some are open, some are close, some are colorful, some are black and white, some are high heels, some are flat, they are different from each other greatly.

Our hybrid-GAN deals with the condition mechanism simply by following the idea in the original proposed conditional GAN [3]. We simply let the original one-hot label $c_{one-hot}$ go through a single full-connected layer $E_{G(\cdot)}$ and then merge the embedded condition $E_{G(c_{one-hot})}$ with other inputs to build a start vector V. As to the discriminator, we also embed the $c_{one-hot}$ with two fully connected layers $E_D(\cdot)$. We then repeat the $\mathcal{R}^{c1}$ shaped $E_D(c_{one-hot})$ to $\mathcal{R}^{c1 \times \mathcal{H} \times \mathcal{W}}$ and concatenate it with the upstream feature map $\mathcal{R}^{c2 \times \mathcal{H} \times \mathcal{W}}$[5].

w.r.t [5], we have also tried the trick to feed the discriminator with real images and mismatch embeddings to enhance the condition supervision. But our experiments showed that such training trick harms the stability of the training of our model. Our explanation is that at the original stage of training, the discriminator should help the generator to build an object, because only when the generator learns to generate some profile of an object can it learn to fit the object towards the right shape and texture [Figure 3.]. But the added loss term back propagates different task gradients so the generator learns neither condition task nor object building task. We have tried to reduce the impact by adjusting the weight of such loss term in the total loss expression with a small λ but only find that it just delays the point that our model collapses. And even without such trick, our model still can work well obeying the limitation of the given condition. So we do not adopt this trick in our model.



(a)Epoch 1 (b) Epoch 10



(c)Epoch 30 (d) epoch 100

Figure 3.

3.2 information encoder

InfoGAN [2] proposed a novel GAN that the discriminator should try to revert some parts of the input of the generator. We think of it as a kind of encode-decode process. In order to infer the original input code of the generator from the generated image, the discriminator should try to decode the information that the generator encodes into the image. On the other hand, the generator should try to encode the information as a distinguishable vision feature into the image. It's no more an adversarial mechanism, but a cooperation. And the loss to supervise this is a regression loss. The gradient backpropagated pushes the generator to encode better.

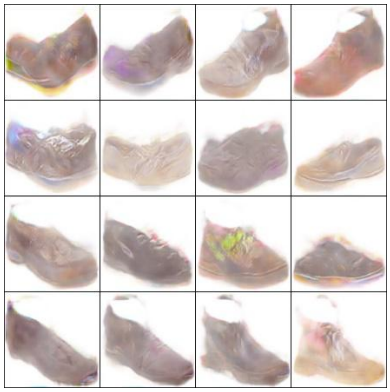
In our model we adopt this idea and implement this mechanism w.r.t [2] to try to control some vision forms of the generated images. We generate a uniform random vector c_{info} and then merge it with other input to build the start vector $V = (z, c_{class}, c_{info})$ of the generator while c_{class} denotes the $E_G(c_{one-hot})$ noted in 3.1 and z denotes a normal random vector. As to the discriminator we output the c^*_{info} with a fully-connected layer $Q(\cdot)$ behind all convolution layers $D_{head}(\cdot)$. We use MSE loss to evaluate the distance between c^*_{info} and c_{info}. We do not use the discrete code in [2] because the label form is already served as condition mechanism.

In our experiment we find the two components of c_{info} provide some kinds of control to some global aspect attributes, such as some subtle meaningful changes in the shape of the generated objects [Figure 4.]. This property can be used to help control such features of the generated shoes so that they can meet the shoes designers' demands.

Figure 4. This grid image is generated by varies two components of c_info along two axes from -1 to 1 evenly. It can be obviously observed that along height dimension the shoes become fatter while along width dimension the uppers of the shoes become lower.

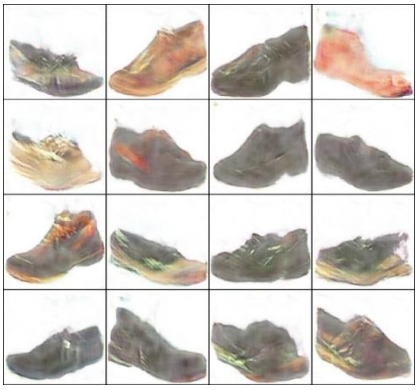
3.3 self-attention mechanism

SAGAN [8] introduces the self-attention mechanism into GAN and greatly improves the quality of the images generated. After experiments [Figure 5.], we find it does help to our model so we take it in.



(a) Trained on dataset1 without SA layer

Figure 5.



(b) Trained on dataset1 with SA layer

Figure 5.

But we find the self-attention layer charges heavily both in computation time and running memory [Figure 6.], especially in the generator. Therefore, we remove the self-attention layers in the generator but still observe significant improvement. Noted that the self-attention layer is exactly the same as that in [8].

3.4 variation tricks

One of the troubles that GAN may stick in is the mode collapse. To tackle this problem, [7] has proposed a trick called minibatch standard deviation, which simply computes the standard deviation for each feature in each spatial location over the minibatch and merges it to the feature map as an additional channel. We also take in this idea by simply doing the same operation to the input batch images of the discriminator.

3.5 loss function

We use WGAN-GP loss [4] as our GAN loss. Therefore, our loss expression are as follows, where $\text{gradientpenalty}(\cdot)$ denotes the gradient penalty term proposed in [4]. In addition, λ_{GP} should be exactly 10, and any smaller rate can lead to model collapse. After experiments, $\lambda_{infoloss}$ being 10 is good in our model and different weight could be set to offer different density of supervision for its robustness.

$$\mathcal{L}(\theta_G|z,c_{one-hot},c_{info},D,E_D) = -D(G(z,c_{one-hot},c_{info}),c_{one-hot})$$

$$\mathcal{L}(\theta_D|z,c_{one-hot},c_{info},D,E_D) = D(G(z,c_{one-hot},c_{info}),c_{one-hot}) - D(I_{groundtruth},c_{one-hot}) + \lambda_{GP}\text{gradientpenalty}(\theta_D)$$

$$\mathcal{L}(\theta_Q,\theta_{D_{head}},\theta_G|z,c_{one-hot},c_{info},D,E_D) = \lambda_{infoloss}\text{MSE}(Q(D_{head}(G(z,c_{one-hot},c_{info}),c_{one-hot})),c_{info})$$

4. Experiment

We conduct several experiments to finetune our parameters and model structure. Besides we also conduct some confrontation experiments to show our model's ability. Then we conduct some visual experiments to show the meaning of c_{info}

4.1 Dataset.

To evaluate our model's ability, we rearrange the dataset UT-Zap50K to a subset suitable for our GAN task with two versions of dataset: hard dataset1 and easy dataset2 to respectively test the robustness and the image quality of our model. Hard dataset1 consists of 12 kinds of shoes including boat shoes, boots, clogs&mules, flip flops, fast walker, flats, heels, loafers, oxfords, sandals, slippers and sneakers&athletic shoes. They are different from each other greatly and have their own significant features. Noted that all the shoes in the dataset have only one position with their head pointing towards left. We use horizontal flip to make it harder for our model when training the dataset1. Easy dataset2, as a subset of dataset1, consists of 5 classes which include boat shoes, boots, loafers, oxfords and sneakers&athletic shoes. They are less different from each other than dataset1. Results showed that our model works not bad in hard dataset1 while works well in dataset2 and can generate high quality images [Figure 7.].

	Model1	Model2
SA layers on G	0	2
SA layers on D	2	2
Batch size per GPU	8	1
Running Memory	11651MiB	>16130MiB

Figure 6. We implement model1 with pytorch parallel function on 4 v100 GPU with batch size 32, which means each GPU operates a sub batch size of 8, and the main GPU0 is occupied nearly 11651MiB. But when we try to run our model2 with two SA layers in generator, even though we reduce

the batch size to 1, the total 16130MiB memory of v100 GPU is still not enough to run the generator.

(a) High heels in dataset1

(b) Dataset2

Figure 7. On hard dataset1 some high heels generated is affected by boots, but generally it works not too bad. And on easy dataset2 it works pretty well.

4.2 Training metric.

We use adam optimizer and set the learning rate of G and Q to be 6e-5 and D to be 4 times of G. Betas are all set to be (0.5, 0.999). Our final model is parallelly trained on 4 v100 GPUs with batch size 32 and totally 200 epochs. It takes only one day to train. Indeed, it looks pretty well after 100 epochs and the rest 100 epochs is only slightly fine-tuning.

(a) (b)

Figure 8. The final result. Some shoes even have lifelike reflective light

(c)

Figure 8. The final result. Some shoes even have lifelike reflective light

4.3 Some failed tries.

We have tried smaller λ_{GP} like 3, but the loss of G explodes to millions and billions while loss of D explodes to minus millions and billions. We have tried mismatch pairs for training condition mechanism, and model explodes like as mentioned above. Attempts like reducing the mismatch example loss weight to even 0.001 only delay the time point that model collapse for tens of epochs. We think such training trick may not suitable for our model. We tried to add single self-Attention layer in G but it can't learn to generate good-looking shoes but some shadow-like single color object [Figure 9.]. We also tried spectral norm w.r.t styleGAN [9] but the failed result looked the same as the single self-Attention try in G.

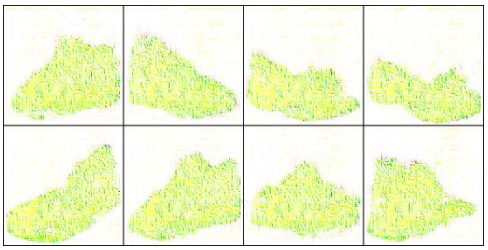


Figure 9. Some failed results

5. Conclusion

In this paper, we propose a hybrid model which combines a series of generative adversarial networks. We try different forms of tensor flows, different loss function and different training tricks to find the best combination. And the experimental results demonstrate that our final

model can generate images with good quality. But we just do a preliminary exploration. Probably our model can be further improved by combining some better models and tricks, such as styleGAN [9], progressive training [7] and so on. But in general, we prove the possibility of combining several models into a hybrid model to achieve good performance.

Reference

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[2] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[3] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[4] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA,* pp. 5769–5779, 2017.

[5] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text-to-image synthesis. In *ICML*, 2016.

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

[7] Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. *CoRR*, abs/1710.10196, 2017.

[8] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-Attention Generative Adversarial Networks. *arXiv e-prints, page arXiv:1805.08318*, May 2018.

[9] Tero Karras, Samuli Laine, and Timo Aila. A StyleBased Generator Architecture for Generative Adversarial Networks. *arXiv e-prints, page arXiv:1812.04948*, Dec 2018.

[10] Jaime Deverall, Jiwoo Lee, and Miguel Ayala. Using Generative Adversarial Networks to Design Shoes: The Preliminary Steps. http: // cs231n. stanford. edu / reports / 2017 / pdfs / 119. pdf. June 2017.