CSGAN: Content with Style Generating Using Generative Adversarial Networks

Hongyuan Du
Department of Electrical and Computer Engineering
UC San Diego
hdu@ucsd.edu

Abstract

In this work, we propose CSGAN, a novel model to associate Generative Adversarial Networks (GANs) with style transfer techniques. The idea is that we expect GANs with enhanced capacity can generate not only target contents, but also specific styles, which can be very useful in large scale artwork generating. We propose a two-step approach to learn a mapping $G : z \to Y$, such that the distribution of images Y created from low-dimensional latent variables z by generator G is both indistinguishable from the distribution of content C and style S. In addition to providing new tools for art creation, we have observed interesting results that by training only a few layers of the network we can capture the distribution of styles, which could help to understanding the structure of GANs. Our project is completely open source. Codes are available at `https://github.com/dashidhy/CSGAN`.

1 Introduction

After it was proposed in 2014, Generative Adversarial Network [8] never stops demonstrating its power in handling computer vision tasks. This framework and its variants have become the state-of-the-art method in many areas related to learning a generative model, such as image generating [8, 17, 15, 12, 3], image-to-image translation [24, 10, 20], super resolution [14], video synthesis and rendering [19], etc., which have achieved stunning results compared to traditional methods.

This paper focuses on a new application of GANs in image style transfer. Style transfer is a subset of image-to-image translation problems in which the goal is to convert images from a given domain X to another Y, e.g., grayscale image to RGB, photo to artistic painting, stick figure to detailed picture. We usually have to learn the mapping $f : X \to Y$ between the two domains, and this can be done by training a deep neural network. CycleGAN proposed in [24] is a breakthrough in this area, in which the pair of generators are trained to maintain cycle consistency between the two domains X and Y to approximate the mapping f and f^{-1}.

A lot of works like CycleGAN have been done for image-to-image translation and have achieved impressive results, but there is a question: what if we want to create new images in domain Y? It's straightforward to think about concatenating two networks: the first for generating, and the second for translation, but this would be time and computation consuming. Is it possible that we train only one network that can do the two things at the same time? The answer is yes. In this paper, we present a new strategy for training a GAN – specifically, a generator G – to help it directly learn a mapping $G : z \to Y$, where $G = f \cdot g$ with $g : z \to X$. We test our method in style transfer tasks, by firstly train a traditional GAN for generating images with content C, and then fine-tune the generator on a target style image S. Our experiments are quite promising, that the generator can create 64x64 and 112x112 images with style in less than 2 minutes fine-tuning, shown in Figure 1. We also notice that by fine-tuning only the last few layers of the generator we could generate equivalent images as fine-tuning the whole network, which provides further understanding of the hierarchical structure of GANs.

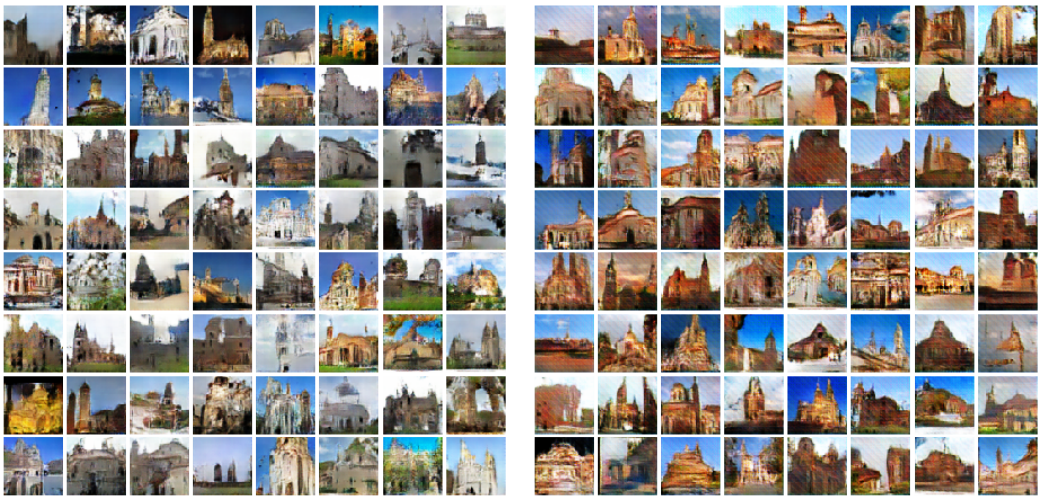

Figure 1: CSGAN outputs. **Left:** Sampled outputs from a DCGAN pretrained on church-outdoor class of the LSUN dataset. **Right:** Sampled images created by the generator fine-tuned on Edvard Munch's *The Scream*.

2 Related Work

Generative Adversarial Networks (GANs) By minimizing an adversarial loss, GANs provide a unprecedented solution for neural networks to approximate a probability distribution. In information theory, the adversarial loss can be interpreted as the f-divergence [16] between two distributions (Jensen-Shannon [8], Person χ^2 [15], Earth-Mover [1]). In our work, we suppose the distribution of images with target style should not be "far away" from the distribution of training data. Difficulty should lies in learning the contents, not the colors or textures. Thus, we should first feed the network with an enormous amount of content images, and then fine-tuning will be easy. Our experimental results confirm our assumption.

Neural Style Transfer Gatys et al. [6, 7] propose that the statistics of Gram matrix in multiple layers of a convlutional neural network (CNN) is a stable, multi-scale representation of the input images' styles. They use a CNN pretrained on ImageNet [5] as a "loss network". Style transfer is realized by first evaluate the difference of Gram matrix between input images and the style target as a loss, and then directly backprop the gradients to optimize the values of inputs' pixels. Johnson et al. [11] improve the method by inserting another CNN between inputs and the loss network for approximating a transfer function, so that just by forwarding we can rapidly process new images. Our idea is inspired by Johnson's method. We replace the transfer CNN by a pretrained GAN generator and do backpropagation to fine-tune it. Fed with batches of low-dimensional noises, the enhanced generator can create a large bunch of new images with desired contents and target styles, so we call this technique *Large Scale Style Transfer*.

Understanding GANs Although GANs have achieved amazing results in many areas of computer vision, there is a lack of research in understanding their structures. Recently, [2] presents a strategy for visualizing and understanding GANs on a per-neuron level. The authors find that contents in a image generated by GAN are controlled by specific neurons in the generator, and by activating/ablating those neurons they can control the appearance/vanishment of target contents. Different from [2], our work investigates GANs on a per-layer level. We find that just by fine-tuning the last few layers of the generator we can finish style transfer and get equivalent results as by fine-tuning the whole network, which implies that deeper layers tend to control the style (texture, color, ...) of the image, and shallow layers control more on the contents.

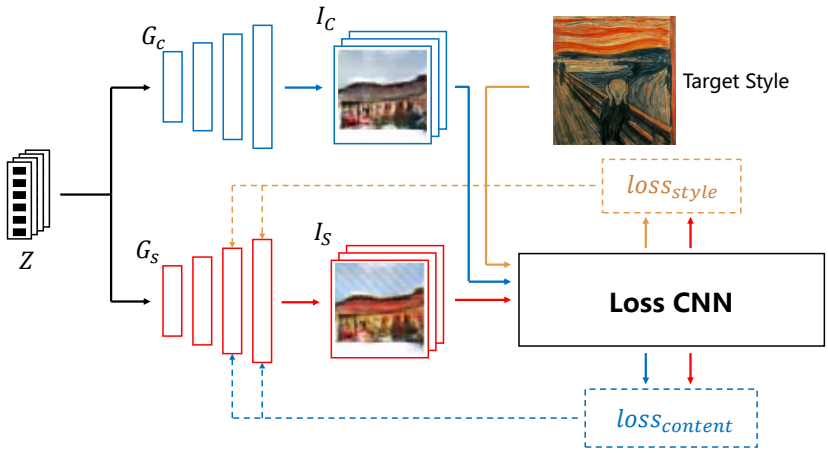

Figure 2: The fine-tuning step of our CSGAN system. What is not shown in this figure is a *Total Variation Loss* [11] $loss_{TV}$ for encouraging spatial smoothness of output images. In our experiment, we find whether implementing $loss_{TV}$ won't affect the result too much.

3 Method

Our CSGAN system is trained by a two-step strategy. The first is training a regular deep convolutional GAN on a large dataset containing desired contents. This step encourages the generator to learn a mapping $g : z \to X$, where z is latent noises sampled from a prior distribution (uniform, Gaussian, ...), and X is the domain of contents.

The next step is designed for learning the style transformation. As shown in Figure 2, we duplicate the pretrained generator into two copies G_C and G_S. We set G_C to evaluation mode (do not requires gradients) for providing mini batches of contrast $I_C \in X$ to evaluate content loss. G_S is fine-tuned for style transfer. During each iteration of fine-tuning, we sample a mini batch of latent variable z and feed it to G_C and G_S simultaneously to create a content-style pair I_C and I_S. Similar to [7, 11], we send the pair together with a style image to the loss network to compute the losses, and then backprop the gradients to G_S for optimization. This step enhances G_S to learn the transfer function $f : X \to Y$ where Y is the domain of desired style, so finally we obtain the mapping $G = f \cdot g : z \to Y$. The effect of optimizing the whole network or only the last few layers of G_S is discussed in detail in Section 5.

4 Implementation

We trained an LSGAN [15] and a DCGAN [17] with inverted order of batchnorm and rectifier layers on the *church-outdoor* class of LSUN [22] dataset. Training images were central cropped and resized to 64x64 for DCGAN, 112x112 for LSGAN (same as the output size of generator), and then pixel values were rescaled to $[-1, 1]$. No other preprocessing or data augmentation was used. Network parameters were initialized by default settings in PyTorch 0.4.0. The LSGAN was trained for 24 epochs and the DCGAN was trained for 9 epochs, both with a batch size of 128. The initial learning rate of our Adam optimizer was set to 2e-4, with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and was decreased by a multiplier of 0.9 for generator and 0.85 for discriminator every 1000 iterations. When training LSGAN, the multipliers were increased to 0.95 and 0.9 after 16 epochs. Least square loss in [15] was used for both of the GANs. Some tricks used for stabilizing the training process will be discussed in Section 6.3.

For style transfer, we used SqueezeNet [9] as our loss network. The outputs of layer 1, 4, 6, and 7 were used for evaluating style loss and layer 3 for content loss. Different weights were given to each of the losses for best performance on different target styles. For fine-tuning, we used a standard Adam optimizer with learning rate of 1e-4 and a batch size of 128. Under these settings, the generator could be well enhanced in about 100 iterations (less than 2 minutes with a Nvidia GTX 1080 Ti GPU) to get ideal outputs.

5 Results

Images in Figure 1 are generated from a DCGAN, each with resolution of 64x64. We also generate more beautiful 112x112 images by LSGAN, shown in Figure 5 and Figure 6. (You can zoom the figures for detail.)

Understanding Layers of the Generator

It has been shown in lots of literatures that CNN designed for classification tasks could be considered as a hierarchical feature extractor [13, 23, 21], that shallow layers detect basic patterns like colors, edges, textures, etc., and deep layers focus on superior features, such as human eyes or wheels of cars. We supposed the generator of a deep convolutional GAN also has this kind of characteristic and designed the following experiments.

We monitored relative changes of convolution layers in a DCGAN during fine-tuning. The result is shown in Figure 3. You can find that deeper layers experienced much more variation. This implies we can imagine the generator roughly as an inverse of a CNN classifier, that its shallow layers decide the content to generate, and deeper layers control more on the images' style. Further, we can surmise that if there is a strong independence between the controlling of contents and styles, we could do style transfer just by fine-tuning the last few layers instead of the whole network. We also verified this gussing. Images generated by a partly fine-tuned generator are show in Figure 4 and Figure 6.

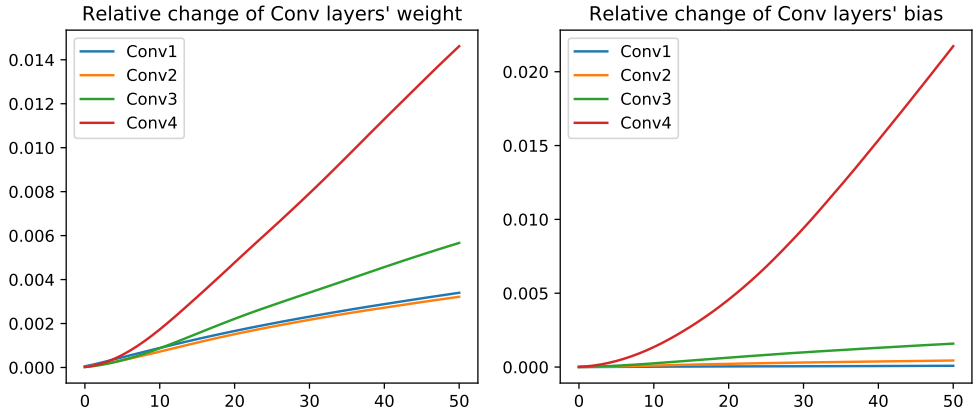

Figure 3: Parameter variations.

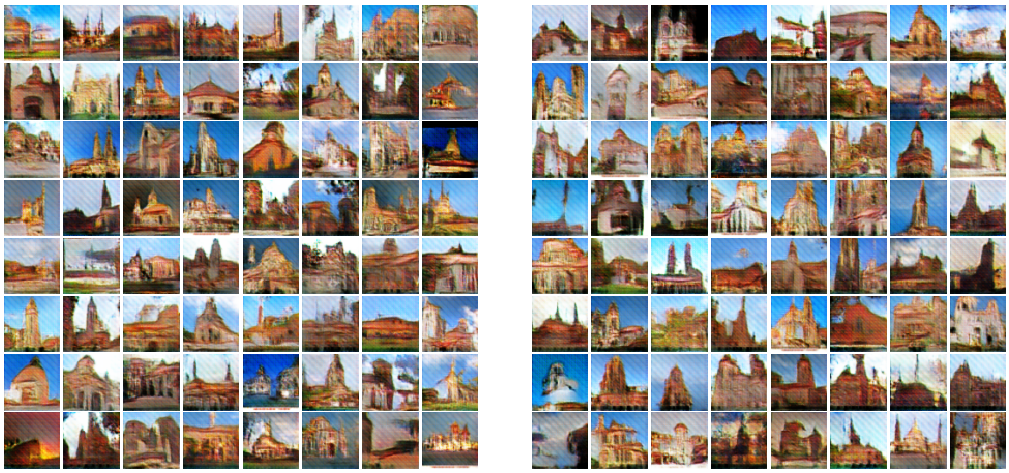

Figure 4: Left: After fine-tuning the whole DCGAN. **Right:** After only fine-tuning the final two convolution layers.

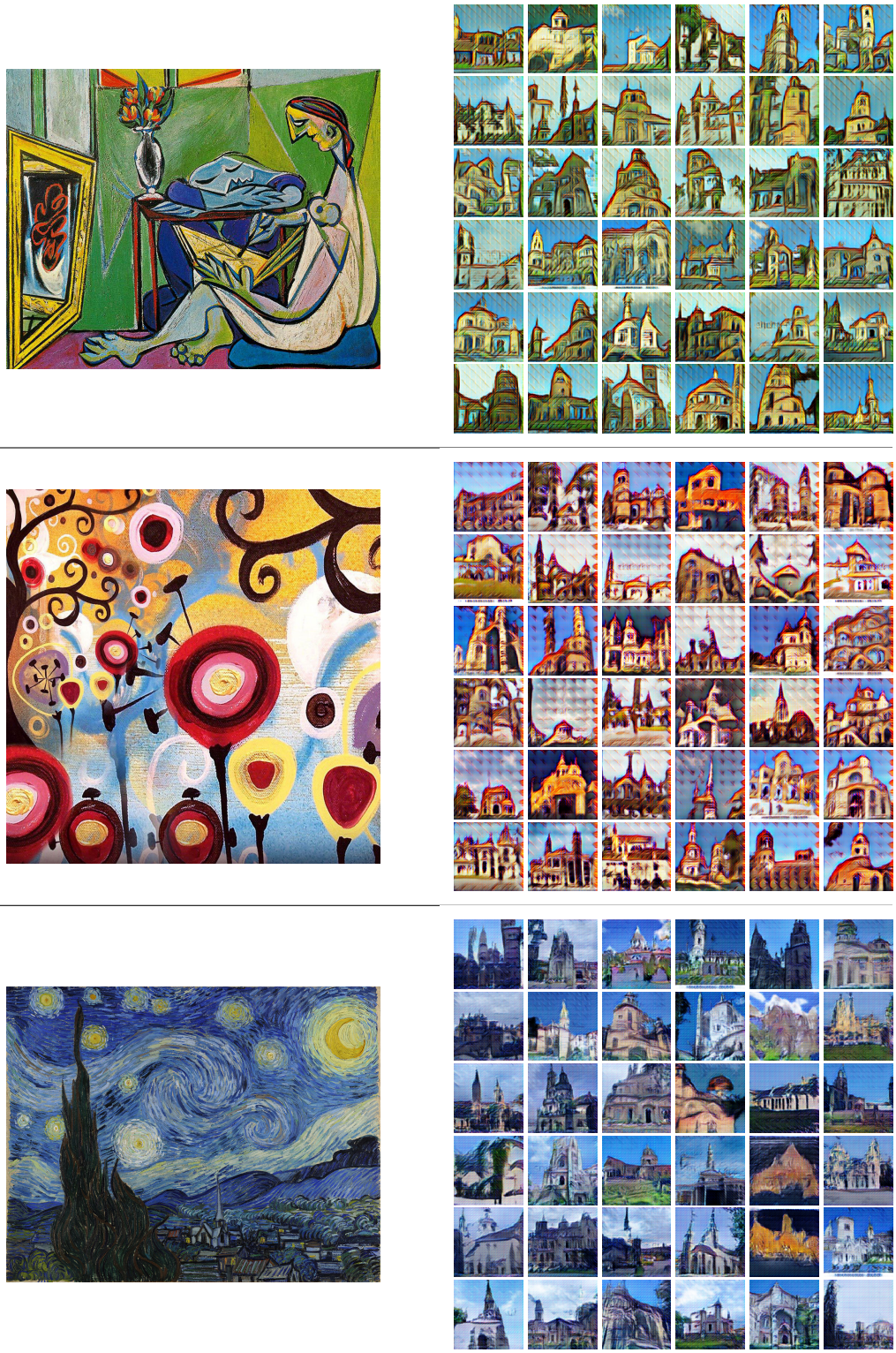

Figure 5: 112x112 images I. Whole generator was fine-tuned.

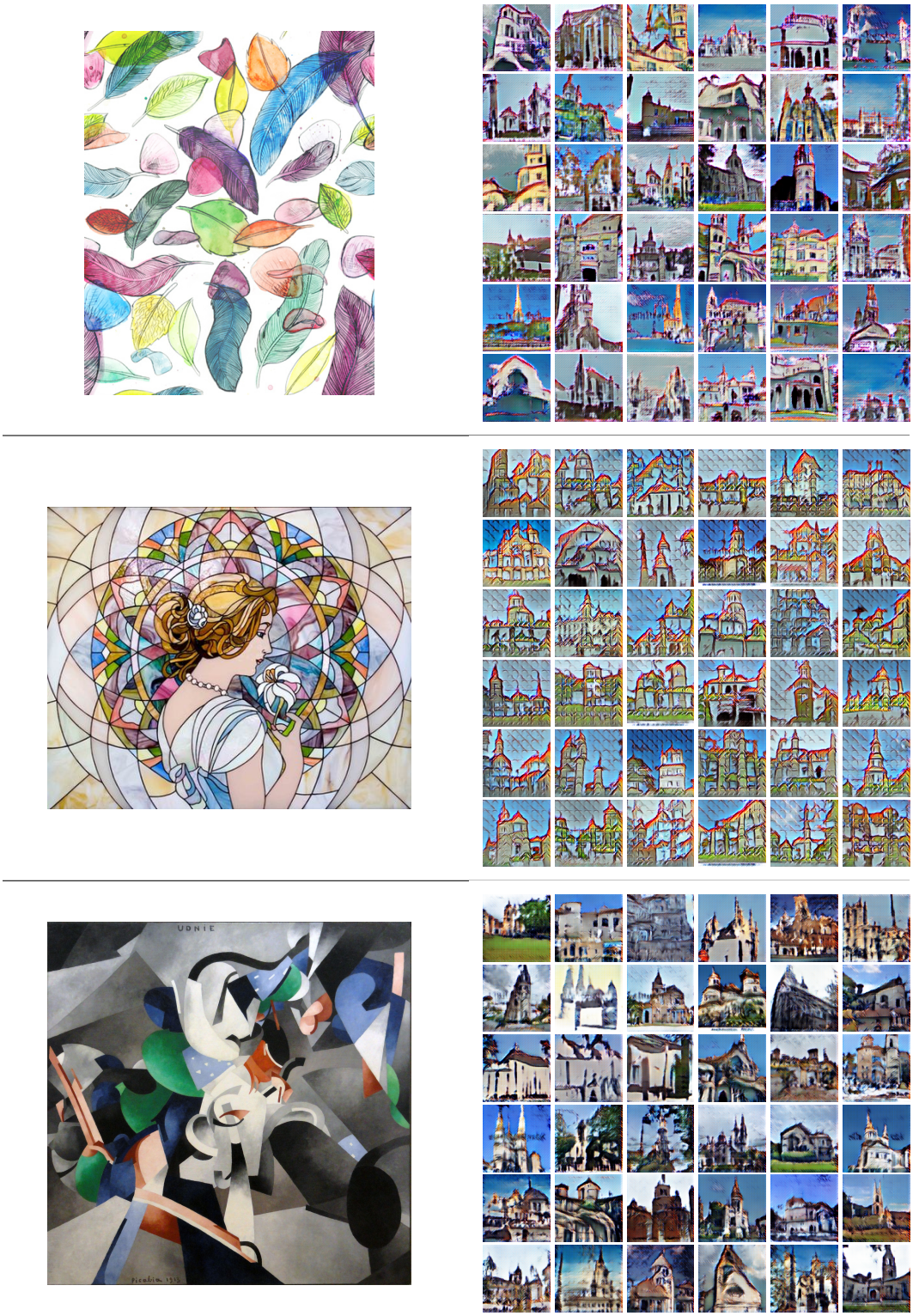

Figure 6: 112x112 images II. Only last three convolution layers of the generator were fine-tuned.

6 Discussion

6.1 Conclusion

We propose a novel framework for large scale style transfer tasks. By training GANs on a large dataset and then fine-tuning them, we can create a large number of entirely new images with specified contents and styles. Time spent on the fine-tuning process is nearly negligible compared to on the GAN training. We also demonstrate by experiments that the generator of GANs has hierarchical generative structures which first determine the content of image and then add style details to it. Our work can be very useful in the area of art creation, that just by manipulating low-dimensional parameters we can rapidly create new artworks in large quantity.

6.2 Limitation and Future Work

Though the results are promising, our work is not perfect. First, it seems that our generators suffer from learning subtle textures of target style images. *Starry Night* in Figure 5 and *The Feathers* in Figure 6 are two failure examples. The generators can only learn color patterns, and texture details are lost. Second, the quality of fine-tuning is to some extent depends on hyperparameters, sometimes leads to very bad images.

To break those limitations and make the framework more robust, we think our work could be improved in the following aspects. To deal with details of styles, we have observed that it partly depend on resolution. In our experiment, LSGAN that can generate 112x112 images is clearly better than DCGAN with 64x64 images. Thus, using high resolution GANs such as Progressive GAN [12] would be a nice trial. To make it more robust to hyperparameters, we believe that a more powerful loss network should be used. The SqueezeNet we used is a lightweight network with AlexNet-level accuracy, which is not quite strong in feature extracting. Deeper network like VGG [18] could be a better choice.

We also imagine some directions for extension:

- **Informative latent variables:** [4] presents a training strategy to encourage GANs learning a interpretable representation of its input variables. We believe this technique can also be used in our work for controlling what contents should present in the image and what the style is.
- **Large capacity networks:** Informative latent variables will rely on the generator to have enough capacity for learning a complex distribution consisting of several different contents and styles. [3] has proved the feasibility, but it may depend on a very complex training process.
- **End-to-end training:** Our train-and-fine-tune strategy may let the generator stick more on the content distribution, which leads to difficulties in learning desired styles, so we are thinking about if there is a end-to-end way to help the generator learn contents and styles simultaneously.

6.3 Some Tips for GAN Training

Though it's not quite related to our work, we would like to share some useful tricks to make GAN training more stable and controllable:

- **Layer order:** It is a question that what order of normalization and nonlinearity layers to use, BN-ReLU or ReLU-BN? We tried both of the orders, and here is a conclusion: first BatchNorm, then ReLU. When using BN-ReLU both in generator and discriminator, we observed much more stable decreasing of the GAN loss.
- **Soft label:** Instead of using hard labels 1 and 0, we use random numbers in [0.78, 1.12] as real label and [0.0, 0.24] for fake ones.
- **Conditional training:** Discriminator will dominate the training process if you don't control it, making the generator hard to optimize. A very simple trick is to set a threshold, and train the discriminator only if the loss is larger than it. In our experiment, we use 0.2.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning (ICML)*, 2017.

[2] D. Bau, J.-Y. Zhu, H. Strobelt, Z. Bolei, J. B. Tenenbaum, W. T. Freeman, and A. Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. *arXiv preprint arXiv:1811.10597*, 2018.

[3] A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations (ICLR)*, 2019.

[4] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2016.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[6] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2015.

[7] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2014.

[9] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[10] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[11] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision (ECCV)*, 2016.

[12] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations (ICLR)*, 2018.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2012.

[14] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and et al. Photo-realistic single image super-resolution using a generative adversarial network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[15] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley. Least squares generative adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[16] S. Nowozin, B. Cseke, and R. Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2016.

[17] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2016.

[18] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[19] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro. Video-to-video synthesis. In *Conference on Neural Information Processing System (NeurIPS)*, 2018.

[20] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[21] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[22] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[23] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision (ECCV)*, 2014.

[24] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.