EFFICIENT MULTI-COLLECTION STYLE TRANSFER USING GAN[ID: 81]

FAN BU [FANBU@SEAS.UPENN.EDU],
RUNZHI ZHOU [ZRZ@SEAS.UPENN.EDU],
ZHAOZHENG SHEN [SHENZHZH@SEAS.UPENN.EDU],

ABSTRACT. In this project, we propose a new model that can make style transfer from single style image, and allow to transfer into multiple different styles in a single model. We apply patch permutation and adversarial gated networks for multi-collection single-style image style transfer. From the experiment, we noticed that there are mainly two issues with the transferred image. First of all, some content of the image are lost after transferring to a style, and we will consider to construct a convolutional network for classification. On the other hand, the transferred image have many grids after training a large number of epochs. If we have time, we will try to do patch permutation on transferred images before feeding into the discriminative network or edit loss to figure out the reason of this issue.

1. INTRODUCTION

Generative Adversarial Networks (GANs) are a powerful class of neural networks that are used for unsupervised learning. GAN is made up of two networks called generator and discriminator. The role of the discriminator is to discriminate real from fake signals. The aim of the generator is to fool the discriminator. The generator is normally combined with the discriminator to form an adversarial network. It is through training the adversarial network that the generator learns how to produce fake signals that can trick the discriminator.

Style transfer is a big application of GAN, which enables amateurs to produce fantastic and versatile images in seconds, so that it has many practical applications in cartoon generation, oil painting re-rendering and art education, etc. Style transfer is a useful image synthesis technique that can re-render given image into another artistic style while preserving its content information. For example, when a user takes a picture of a beautiful landscape, he might hope to re-render it on canvas such that it appears to have been painted by an artist, so it is possible to imagine how the artist might render the scene.

However, most methods for style transfer rely on sufficient amount of style images to train the model, and traditional style transfer also has 2 drawbacks. First, traditional GAN can only train one style/author's work, which means that every time you want to do a style transfer of a certain writer's work, you need to retrain a model, which is very inefficient. Second, traditional style transfer requires many content images and many style images at the same time.

We will solve these two problems in this project. In our project, we apply Gated-GAN[1] to transfer multiple styles in a single model, and implement the patch permutation and patch discriminator like the implementation in the P2-GAN [6], which efficiently learn the style from a single style image. We build a model that can make style transfer from single style image, and the styles are controlled by switching different gated-transformer module.

1.1. **Contributions.**

Apply patch permutation and adversarial gated networks for Multi-Collection Single-Style Image Style Transfer.

Analysis the problems in experiment, including the transferred image have many grids and some content of the image are lost(black or white holes).

2. BACKGROUND

The style transfer model in our model can make style transfer from single style image, and the styles are controlled by switching different gated-transformer module. There is a generator $G(\cdot)$ and a discriminator $D(\cdot)$ in our model.

The input of the generator are $\{x_i\}_{i=1}^{N}$, which contains N content images, and the output of $G(\cdot)$ is $N \times M$ fake images $\{G(x_{ij})\}_{i=1,j=1}^{N,M}$ generated by the generator network, $G(x_{ij})$ is the generator image that corresponds to input content image x_i and style $j \in \{1, 2, ..., M\}$, where M is the number of styles.

In the discriminator network $D(\cdot)$, the inputs contains two parts: $\{G(x_{ij})\}_{i=1,j=1}^{N,M}$ is the fake images we generated, and $\{\phi_k(y_m)\}_{k=1,m=1}^{K,M}$ is the style images after patch permutaion, where $\{y_m\}_{m=1}^{M}$ represents M raw style images, and patch permutation breaks the style image into K patches, and we will introduce in more detail latter.

3. Related Work

Most early works on style transfer were pixel-level approaches that relied on low-level visual features. Since Gatys for the first time incorporated convolutional neural networks (CNNs) into the computations of style loss and content loss, CNNs based approaches have become the mainstream of style transfer and achieved convincing results. These methods can be divided into three categories: a) online learning methods[2] [4], patch-swap methods[3][5], and offline learning methods. Generative Adversarial Network (GAN) is a widely adopted framework toward this task for its better representation ability on local style. However, when the number of style images is only one, traditional training methods of GANs will not work due to single sample can not represent a distribution, a novel Patch Permutation GAN (P2-GAN) network[6] was proposed to overcome this problem. In P2-GAN, a patch permutation module and a patch discriminator are proposed, together with an improved encoder-decoder generator for more efficient computation on both offline and online stages. In addition, Gated-GAN[1] introduced the gated transformer module to integrate multiple styles within a single generated network. The generative network consists of three modules: an encoder, a gated transformer, and a decoder. Images are generated to different styles through branches in the gated transformer module. The discriminative network uses adversarial loss to distinguish fake image and real image, and the auxiliary classifier supervises the discriminative network to classify the style categories.

4. Approach

The network architecture is shown in Figure 1, it contains three parts: generator, discriminator and patch permutation. The generative network contains an encode, a gated transformer and a decoder, and it generates a particular style image using gated transformer. Patch permutation breaks one single style image into different patches, which is then feed into the discriminative network. The discriminative network uses adversarial loss to distinguish between stylized images and real images, and use an auxiliary classifier to classify the style categories. In this section, we separate the architecture into three parts and explain each part in each subsection.

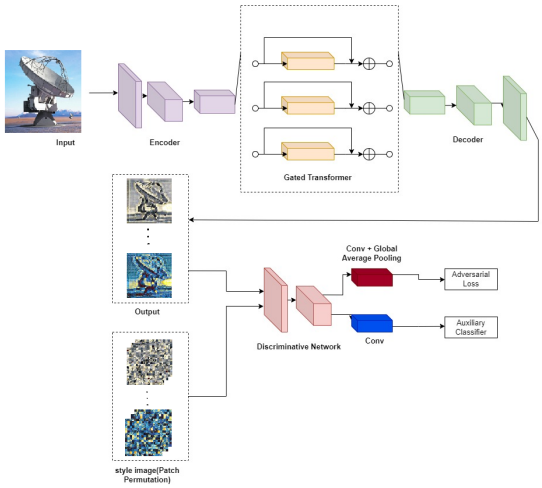

FIGURE 1. Architecture of our network: a generative network, patch permutation and a discriminative network.

4.1. **Generator.**

The generative network implicitly learns the target style from adversarial loss, aiming to fool the discriminator. The whole framework has three modules: an encoder, a gated transformer and a decoder. The encoder consists of a series of convolutional layers that transform input image into feature space $Enc(\cdot)$. After the encoder, a series of residual networks become the transformer: $T(\cdot)$. The input of residual layer in gated function T is the feature maps from the last layer of encoder module $Enc(x)$. The output of the gated function is the activation $T(Enc(x))$. Then, a series of fractionally strided convolutional networks decode the transformed feature into output images $G(x) = Dec(T(Enc(x)))$. To stabilize training, we introduce the auto-encoder reconstruction loss.

4.1.1. *Anto-Encoder.*

To reduce the space of possible mapping functions, we introduce the auto-encoder reconstruction loss. In our model, the auto-encoder is obtained by directly connecting the encoder and decoder modules. The auto-encoder reconstruction loss:

$$L_R = E_{x \in X}[||Dec(Enc(x)) - x||_1]$$

4.1.2. *Gated Generated Network.*

Our gated generative network aims to output images $G(x, m)$ by assigning specific style m. The encoder and decoder networks are shared by different styles, while the gated-transformer transforms the input from encoded space into different styles by switching trigger to different branches: $G(x, m) = Dec(T(Enc(x), m)$. As described in Figure 1, In each branch, we employ the residual network as the transfer module.

Since the only difference among different styles is the transformer, all other parts including encoder, decoder and generator are exactly the same, the network only has to save the extra transformer module parameters for each style.

4.2. **Patch permutation.**

Since we only have one style image, we considered to break the style image into many patches, and learn stroke style from the patches, and we use patch permutation method for the style image. In addition, it can also discard the structure information of style image and can avoid content interference from the style images.

Firstly, patches with size $n \times n$ are cropped from style image y_m at its random position. For any patch p_j , we have:

$$p_j(u, v) = y_m[(W_y - n)r + u, (H_y - n)r + v]$$

where u and v are the horizontal and vertical coordinates, r is a random variable with uniform distribution $U(0, 1)$, and W_y and H_y is the width and height of image y_m.

After T^2 times random cropping, all the patches $p_j, j = \{1, ..., T^2\}$, can be reorganized into a new image: $\phi(y_m)$, where $p_j \in R^{n \times n}, \phi(y_m) \in R^{nT \times nT}$ is a single permutation from the original style image y_m.

We also repeatedly doing such process such permutation K times, therefore get K different permutations. In this case, we can transform one single style image into K different images.

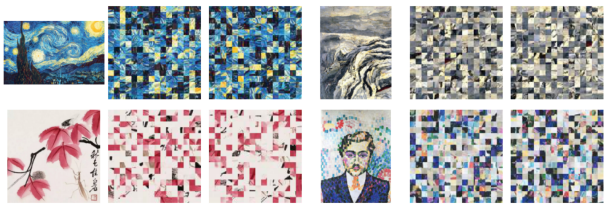

FIGURE 2. Patch permutation result.

Figure 2 is the result of our patch permutation, the next two pictures followed by the original style images are the patch permutation results.

4.3. **Discriminator.**

We use the patch discriminator to build our discriminator network, and we use it to calculate the adversarial loss, and also add an auxiliary classifier after the patch discriminator.

4.3.1. *Adversarial Loss.*

GANs train both the generator $G(\cdot)$ and the discriminator $D(\cdot)$ by applying an adversarial loss $L_{GAN}(G, D)$:

$$L_{GAN}(G, D) = E_{y \in Y}[(D(y) - 1)^2] + E_{x \in X}[D(G(x))^2]$$

The generator $G(\cdot)$ tries to generate an image that looks similar in style to a style image, while the discriminator $D(\cdot)$ aims to distinguish between them.

4.3.2. *Auxiliary Classifier for Multiple Styles.*

If we only use the adversarial loss, the model tends to confuse and mix multiples styles together. Therefore, we need a supervision to separate categories of styles.

An auxiliary classifier C is added in the consideration of leveraging the side information directly:

$$L_G^{Gated} = \lambda_{CLS} E_{x \in X}[H(u(c), C(G(x, c))] + L_{GAN}$$

, where $u(\cdot)$ is the vectorizing operator defined with M classes, and $C(G(x, c))$ is the probability distribution over these classes given by the auxiliary classifier.

4.3.3. *Patch Discriminator.*

The operation of patch permutation in previous subsection will inevitably generate discontinuous textures on junction pixels between adjacent patches. This will incur severe artifacts in the transferred image by traditional discriminators. To address this problem, we use a discriminator $D(\cdot)$ that satisfies:

$$k_l = s_l, n = \prod_{l=1}^{L} k_l$$

, where n means that the patch size is $n \times n$ and k_l, s_l are the convolution kernel and stride at convolutional layer l.

(a) stride = 2, kernel_size = 3 (b) stride = kernel_size = 3

FIGURE 3. Patch discriminator result.

As shown in figure 3, figure (a) is problematic due to the discontinuous texture across adjacent patches, while figure (b) enable discriminator on patch permuted images due to the kernel size equals to the stride in each layer. In this way, we can ensure that any convolution computation in each layer will be performed on one inner patch, and the sliding kernel will never have chance to stretch across different patches. This means our network will only feed forward complete textures from $\phi_i(y_m)$ although discontinuous textures do exist in it.

4.3.4. *TV Loss.*

To smooth the generated image $G(x, c)$, we add the TV loss to make use of the total variation loss.

5. EXPERIMENTAL RESULTS

5.1. **Setup.**

Our network proposed is trained on PASCAL VOC 2007 dataset. 9963 of images in the dataset are used as the content images to train our model. We will train the model to transfer images into 4 different styles, including Mountain No.2 by Jay DeFeo, The Starry Night by Van Gogh, FengYeHanChan by Qi Baishi.etc. For each style image, we use patch permutation to create 20 images that each consists 16x16 patches of size 8x8. We evaluate the quality of style transfer of our model by generating 4 style transferred images for each of the 17 test images. As different settings and parameters can have huge influence on the performance of the network, we did multiple sets of experiments that compare different settings of network to see how number of epochs trained, average pooling , padding of convolution layers and λTV will influence the output image. These comparisons help us to find the best setting of the network and also to understand what is not working well.

In addition, we use the structure of Gated-Gan($https://github.com/colemiller94/gatedgan$) to train our model, and add patch permutation and patch discriminator based on it.

5.2. **Number of epochs.** In this experiment, we trained multiple model with all the same setting except number of epochs. The models' λTV equals to 0.000002. The result of a input image is shown below.

From the result of transferred images, we can notice that as more epochs trained ,the transferred images have more details of original image. On the other hand, we see some white holes in the transferred image. These holes will dwindle with more epochs trained. One thing that is not expected and not desired is that as more epochs trained ,the transferred images tend to be like images after patch permutation, that consist of grids.

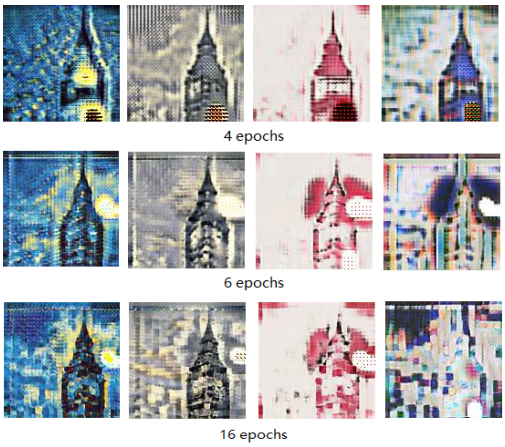

FIGURE 4. transferred image result of different number of epochs ran

5.3. **Importance of total variation loss λTV.** In this experiment, we compare the result of model that one has λTV = 0.000002 and one has λTV = 0.000003.All the other settings of the model are the same. from the result, we can observe

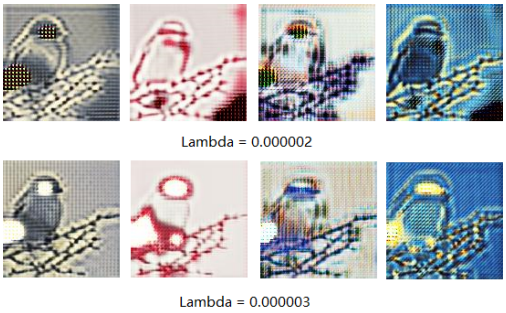

FIGURE 5. transferred image result of different λTV

that increase of λTV will make the image smoother, but a lot of information will be lost. This is because the increase of λTV will result in the decrease of

6. DISCUSSION

In this project, we proposed a GAN-based model that is able to transfer input images into multiple styles within a single model. Our model can learn styles from only one style image by creating multiple sampled style images using patch permutation. From the experiment and result, we noticed that there are mainly two issues with the transferred image. First of all, some content of the image are lost(black or white holes) after transferring to a style. In order to prevent this issue, we will consider to construct a convolutional network for classification such as VGG16. By compute and minimize the distance of corresponding feature maps of the original image and transferred image, we can prevent the transferred image from losing too much structural information. On the other hand, we noticed that after training a large number of epochs, the transferred image have many grids, just like the image after patch permutation. This indicates that the discriminative network learns the grid feature of the style image after patch permutation. If we have time, we will try to do patch permutation on transferred images before feeding into the discriminative network or edit loss to figure out the reason of this issue and solve it.

REFERENCES

[1] Xinyuan Chen et al. “Gated-GAN: Adversarial Gated Networks for Multi-Collection Style Transfer”. In: *IEEE Transactions on Image Processing* 28.2 (Feb. 2019), pp. 546–560. ISSN: 1941-0042. DOI: 10.1109/tip.2018.2869695. URL: http://dx.doi.org/10.1109/TIP.2018.2869695.

[2] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. “Image Style Transfer Using Convolutional Neural Networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[3] Chuan Li and Michael Wand. *Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis*. 2016. arXiv: 1601.04589 [cs.CV].

[4] Chuan Li and Michael Wand. “Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks”. In: *CoRR* abs/1604.04382 (2016). arXiv: 1604.04382. URL: http://arxiv.org/abs/1604.04382.

[5] Jing Liao et al. *Visual Attribute Transfer through Deep Image Analogy*. 2017. arXiv: 1705.01088 [cs.CV].

[6] Zhentan Zheng and Jianyi Liu. *P^2-GAN: Efficient Style Transfer Using Single Style Image*. 2020. arXiv: 2001.07466 [cs.CV].