Neural Style Transfer

GR 5242 Final Project

Zixiao Wang (zw2513), Xingyu Chen (xc2457), Sitong Liu (sl4460), Yue Zhang (yz3383)
Columbia University

Abstract

Two articles by Gatys et al. present some ideas that combining the features of content image and style image through convolutional neural networks could create artistic style images. In this article, we will introduce the specific algorithm of neural style transfer (NST), and share our thoughts as well as the problems we met. Firstly, We will introduce the mathematical foundation of NST, which includes convolution neural networks and loss functions. Then, we will present how the results would change with respect to training iterations, different random seeds and relative weight of content style loss in the loss function. Finally, we will share some further ideas about how to develop our model in the future. All our works have been uploaded to GitHub at

`https://github.com/xc2457/GR5242-final-project`

1 Introduction

Recently, neutral style transfer (NST) has become a popular research topic in the field of machine learning. Before neural style transfer was invented, humans had the skills to create unique visual experiences by constructing complex interactions between the content and style of images to form different artistic styles.[1] However, for artistic style, people have their own opinions, and there are some arts that are probably not clearly defined in the art world. Thus, it is even harder to define how to change the style of one image into another. For programmers, this vague definition is a nightmare, how to turn an inexplicable thing into an executable program became a problem that has puzzled many researchers in the field of image style transfer.

Before neural networks, image style transfer programs had a common idea: analyze the image of a certain style, establish a mathematical or statistical model for that style, and then change the image to be migrated so that it can better conform to the established model. The results of this model perform well, but there is a big disadvantage: a program can only do a certain style or a certain scene. Therefore, the practical application based on traditional style transfer research is very limited.

The seminal work of Gatys changed this situation. Before that, it was impossible for the program to imitate any picture. The research paper published by Leon A. Gatys demonstrated the power of Convolutional Neural Networks (CNN) in creating artistic imagery by separating features and recombining images of content and style.[1] This process of using CNN to fuse the semantic content of a picture with different styles is called Neural Style Transfer.

2 Methods

2.1 Convolutional Neural Network (CNN)

In recent years, deep learning has been extensively used in different fields, for example, visual recognition. For visual perception, leveraging on the rapid growth of annotated data volume and the

powerful advancement of the graphics processor unit, the research of convolutional neural networks has risen rapidly and achieved the latest results in various tasks. [2]

2.2 Visual Geometry Group 19 (VGG19)

In this project, we used VGG19 (Visual Geometry Group 19) for NST. VGG is able to detect advanced features in the image since it contains 19 weight layers and the weights are pre-trained on millions of images. Compared with the other neural network model, VGG19 used multiple 3*3 convolution kernels instead of 5*5 or 7*7 convolution kernels. The main purpose of this is to ensure that the model not only keeps the conditions of the same perceptual field but also improves the depth of the network, which could improve the effectiveness of the neural network to a certain extent.

In our project, VGG is the key part to construct the Neural Style Transfer. Specifically, when we want to apply the style image ($\vec{a}$) to the content image ($\vec{p}$), we will use gradient descent algorithm to update the content of the target image ($\vec{x}$) to ensure that the target image has similar features to the style picture and similar content to the content picture. In this way, the generated target image is the NST image we want to get. Since we want the target image ($\vec{x}$) to have the similar features of the content image ($\vec{p}$) and the style image ($\vec{a}$), we calculate the loss of the generated image ($\vec{x}$) to the corresponding content ($\vec{p}$) and style ($\vec{a}$) image finally.[3] Thus, in the next part, we defined the content loss and style loss to implement our algorithm.

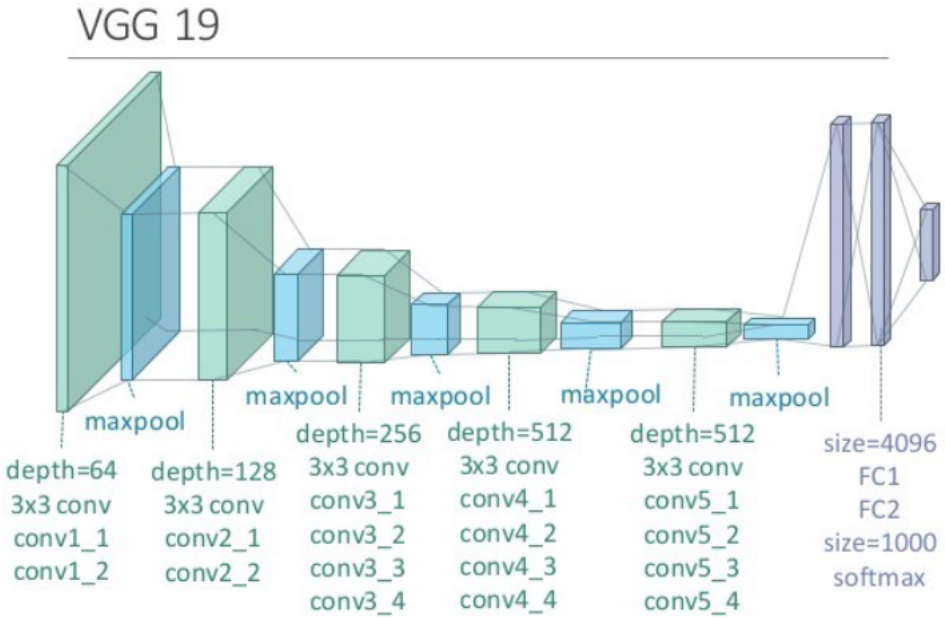

Figure 1: Micro-architecture of VGG19 [4]

2.3 Loss Function

Content Loss Assuming a convolutional layer contains N_l filters, we can get N_l feature maps. We assume that the size of the feature map is M_l (which is the height by the width of the image). Thus, we can store the data of the l layers through a matrix.

$$F^l \in R^{N_l \times M_l}$$

- $F^l_{i,j}$ represents the activation of the i -th filter at the j position in the layer l.

Therefore, after generating a convolution layer l, a content image $\vec{p}$, and a generated picture $\vec{x}$.(Initial is Gaussian distributed), will get their corresponding feature representation: P^l and F^l, then the corresponding content loss would define as follows [5] :

$$L_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{ij} \left(F^l_{ij} - P^l_{ij}\right)^2$$

The derivative of this loss with respect to the activation in layer l equals [5]:

$$\frac{\partial \mathcal{L}_{content}}{\partial F_{ij}'} = \begin{cases} \left(F^l - P^l\right)_{ij} & if F_{ij}^l > 0 \\ 0 & if F_{ij}^l < 0 \end{cases}$$

Style Loss The representation of style here we use Gram matrix:

$$G^l \in R^{N_l \times N_l}$$

- G_{ij}^l is the inner product between the vectored feature maps i and j in the layer [6]:

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l$$

- The Gram matrix calculates the correlation between two features, which means we can use this matrix to represent the style of the image.

Assuming a style image $\vec{a}$, a generated picture $\vec{x}$, after generating a convolution layer l, we can get its corresponding feature representation: $\vec{A}$ and $\vec{G}$, then the corresponding style loss in this layer is [6]:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2$$

Moreover, the total style loss is [6]:

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

The derivative of E^l with respect to the activation in layer l can be computed analytically [6]:

$$\frac{\partial E_l}{\partial F_{ij}^l} = \begin{cases} \frac{1}{N_l^2 M_l^2} \left(\left(F^l\right)^{\mathrm{T}} \left(G^l - A^l\right)\right)_{ji} & if F_{ij}^l > 0 \\ 0 & if F_{ij}^l < 0 \end{cases}$$

Total Loss Initialize an output image by white noise, and then modify the style and content constraints of this result through the NST by Loss Function [6]:

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

3 Results

3.1 Initial Ideas and Preparation

The first challenge we met through the process was that generating new pictures using our model was really time-consuming. We spent a lot of time testing the best parameters to use. Initially we tried some parameters as the reference paper suggested. For example, use higher layers of the network as the content representation. [5] We first chose the convolution layer in block 5 as our content representation. It turned out that the output was too abstract and can hardly present detailed features from the content image and hence, we decided to use a layer in a different block within VGG19 model. Also, the weights for content loss and style loss will greatly influence the output of our model and we worked with different weight ratios.

Another problem we found in this project was that it was hard to evaluate the generated results. Using loss function could quantify the results mathematically, but it did not mean that the outputs were artistically acceptable. In other words, the performance of our generated results was very subjective and relied on different personal reviews. In this way, we evaluated them based on our understanding of the style pictures.

For style pictures, we used the following two pictures (Cubism and Post-Impression Styles):

Figure 1: Left: Nude in an Armchair; Right: The Starry Night

For content pictures, we used the following three images:

Figure 2: Content Images

3.2 Finding Van Gogh in Shanghai

Random Seed Firstly, we tested whether there would be a difference between random seed 0 and 2019.

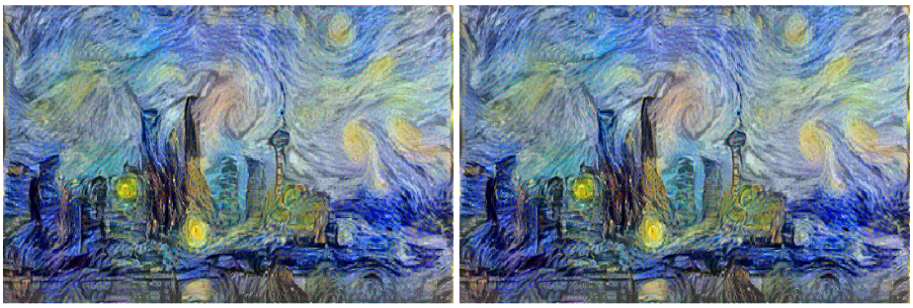

Figure 3-1: Generated Pictures (Left: seed=0; Right: seed=2019)

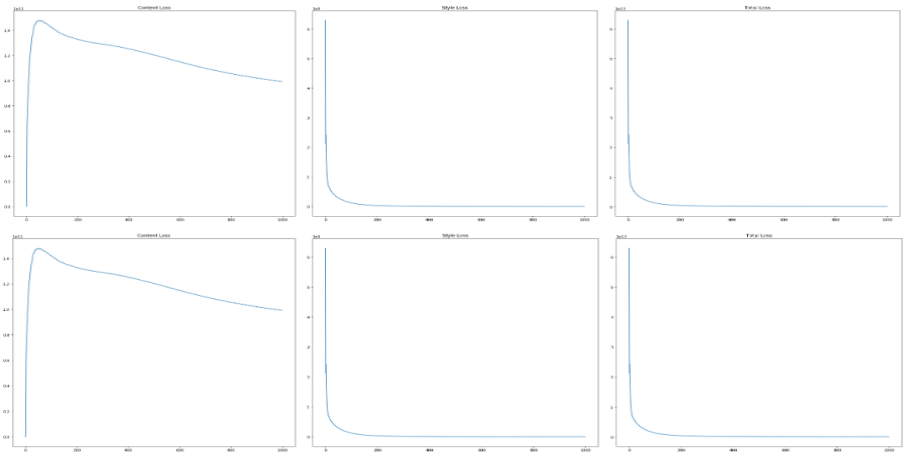

Figure 3-2: Loss Plots (Up: seed=0; Down: seed=2019)

From the above results, the output difference between seed 0 and seed 2019 are very subtle. In addition, based on the loss plots above, the trends of all three types of loss seem to be similar. In conclusion, we think that random seeds would not affect the results too much. Thus, we decided to use seed 0 for the following steps.

Content Layer Representation When we got the results from above, in our personal opinion, the output images seem to include too many features of style image and hard to present content. So we decided to choose different content layers to find out a more comfortable output combining content and style images.

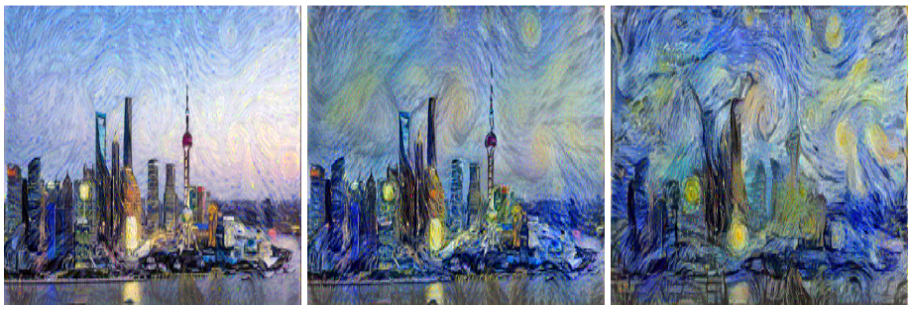

Figure 4: From left to right: 1: Block 1; 2: Block 3; 3: Block 5

Based on the outputs, we noticed that using layers in the block, which is closer to the input content image, will make the output more abstract. In contrast, using layers that are more close to the original input content image, it would be hard to recognize the style we tried to add. Thus, we decided to use the layer of intermediate block in our algorithm.

Relative Weight Furthermore, we want to figure out how the output images would be changed with respect to relative weight. Here, we choose content weight as 1 and style weights as 100, 1000 and 10000.

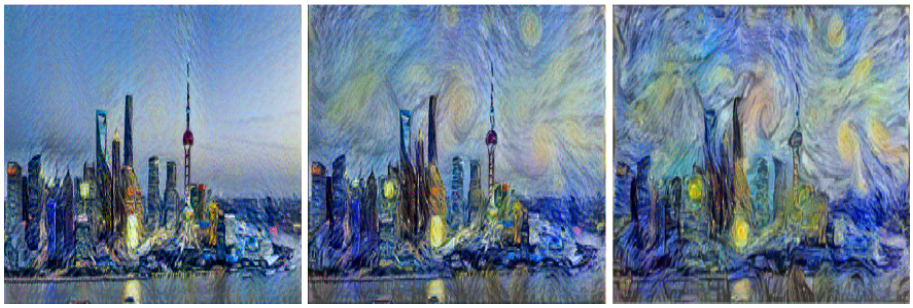

Figure 5: From left to right: 1: 1:100; 2: 1:1000; 3: 1:10000

From the results we can see as the style weight get larger, the picture looks artistically closer to The Starry Night. This is exactly as we expected since through the iteration process, the style loss would decrease more rapidly compared with the content loss. Similarly, if the weight ratio of content to style became larger, after reaching the maximum value, the content loss would decrease more fast, which would make the generated picture include more content features.

Iterations Finally, after we chose the convolutional layers and the relative weight, we used the content image Shanghai and the style image The Starry Night to generate more pictures through the whole iteration process. We expected to see the difference among all pictures in different iterations. We chose pictures of 100, 200, 300, . . . , 1000 iterations to see how the process went.

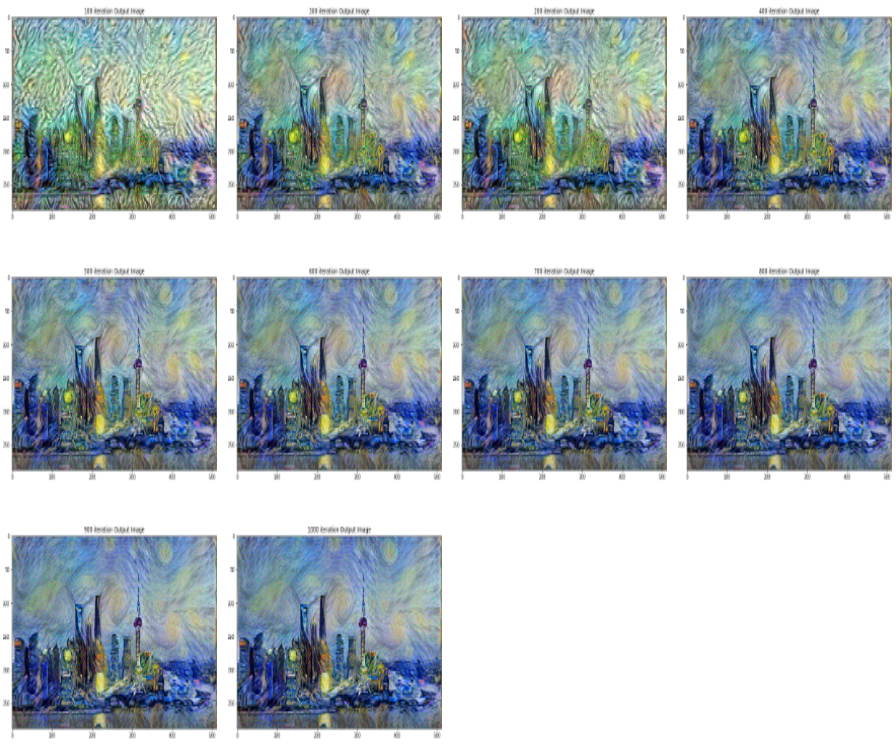

Figure 6: Images in different iterations

By looking at the results, we noticed that the total loss and style loss kept decreasing while content loss started to increase and then decreased until convergence. In other words, at the beginning of the process, more and more style features were added to the content picture and made more noises into the picture. As the process continued, both content and style features became clear in the generated image and finally reached a satisfying balance.

3.3 For More Examples

Here, we chose 3 different content images and 2 different style images to test our algorithm and based on the outputs, there is a significant style transfer among all three content pictures.

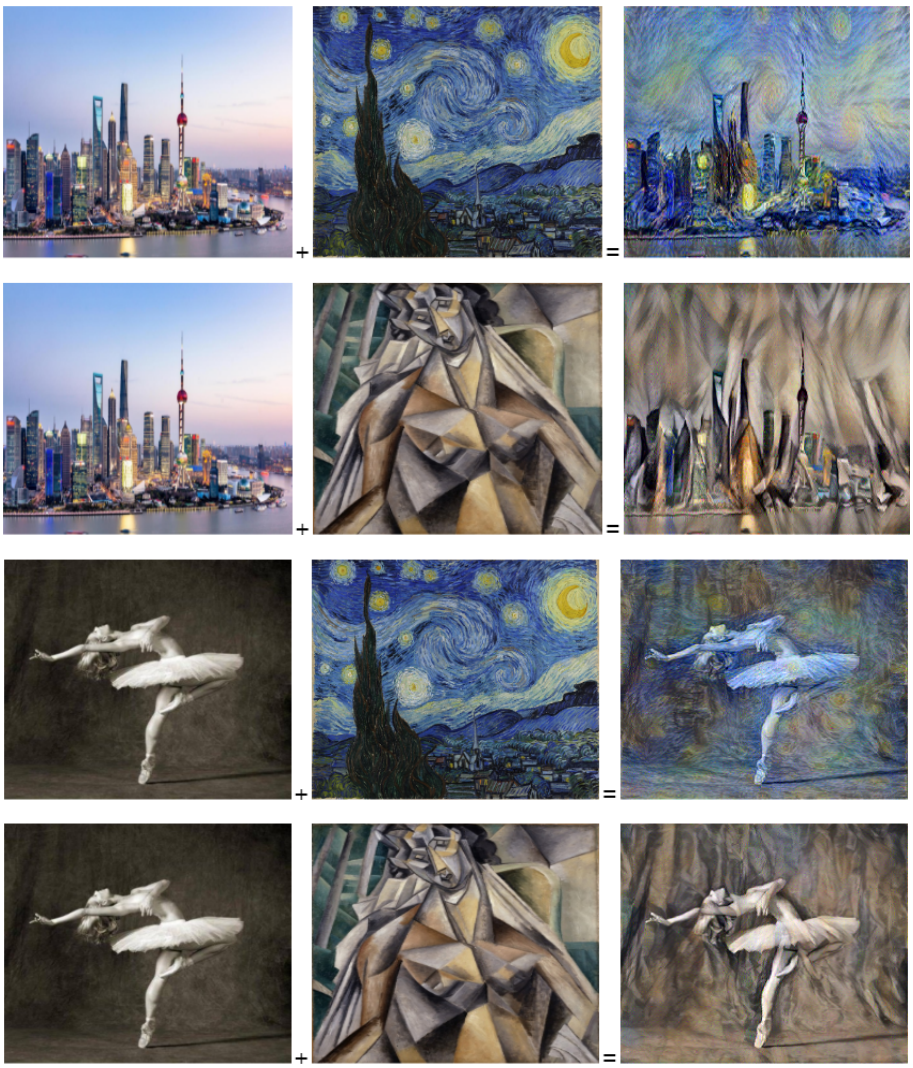

Figure 7: Output Samples

4 Conclusion and Discussions

4.1 Conclusion

The results of NST is very interesting but hard to be quantified. In other words, it is difficult to measure the style of a painting from a mathematical perspective. The conclusion we got is very subjective.

Based on our study, different parameters played an important role in our algorithm. For different content layer representation, choosing one within a block that more close to the original input content image would capture more detailed features of the content image; while one in higher layers would better transfer style from the style picture. As for the weight ratio, we chose the weight ratio 1:1000 but the reference paper also suggested much lower ratios such as 1:100000 to be used. These parameters require people to adjust according to their personal understanding of the style of paintings. In addition, the result could also depend on the style and content images people used. A city content picture might be a good fit for The Starry Night and the output of these two images could better present the features from both city content and The Starry Night style.

4.2 Discussion and Further Thoughts

So far our study mainly focused on the algorithm implementation and how the parameters in the algorithm would affect the output images of the style transfer process. As the results display, different types of content and style images would have different fit.

Through the process, we met different types of problems. When we first implemented the loss functions, we added many intermediate steps, which made the process very time-consuming. By removing some repeated calculations within our functions, the algorithm was improved. For further study, we noticed that there are some reference papers stating that there could be a total variation loss added into the total loss function. This loss is served as regularization, which would add spatial smoothness to image. There are also some applications such as NST for videos which require us to try in the future.

References

[1] Yongcheng Jing et al. *Song.Neural Style Transfer*. A Review. arXiv.org, August 2018. https://arxiv.org/pdf/1705.04058.pdf

[2] David Frossard. *VGG in Tensorflow* https://www.cs.toronto.edu/~frossard/post/vgg16/

[3] Ayush Singh. *How Do Neural Style Transfers Work?* https://hackernoon.com/how-do-neural-style-transfers-work-7bedaee0559a

[4] Park Chansung. *Transfer Learning in Tensorflow (VGG19 on CIFAR-10): Part 1*. https://towardsdatascience.com/transfer-learning-in-tensorflow-9e4f7eae3bb4

[5] Leon A Gatys et al. *Image Style Transfer Using Convolutional Neural Networks. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR, pages 2414 -2423. IEEE, 2016.)* https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Gatys_Image_Style_Transfer_CVPR_2016_paper.pdf

[6] Leon A Gatys et al. *A Neural Algorithm of Artistic Style. arXiv.org, August 2015*. https://arxiv.org/pdf/1508.06576.pdf