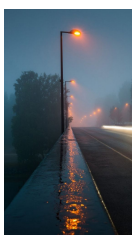


(a) set1: content

(b) set1: style

(c) set1: output

(d) set2: content

(e) set2: style

(f) set2: output

Figure 1: Example Set

Final Project: Neural Style Transfer Quality Improvement

Xinyuan Peng
Columbia University
xp2177@columbia.edu

Xinyi Zhang
Columbia University
xz2862@columbia.edu

Abstract

In this project, we'll implement the neural style transfer based on existing study by Gatys, etc. Briefly, Neural style transfer will take 3 images, a content image, a style image and an input image you want to style, then the style of style image will be transferred to the input image with content from content image. Our goal is to improve the transfer quality by adjusting intermediate layers used in content and style representations and discuss why certain layers are selected instead of others. An experimental model will be developed to compare with the baseline model that was designed by previous scholars. The performance of the models will be tested with respect of training iterations, random seeds and relative weight of content and style loss in the loss function.

1 Related Work

Gatys, etc proposed the artificial system based on deep neural networks in 2015[2], where he discussed the structure to build the model based on VGG-19. The method is further explained by the article published in the following year[1], where five layers of the content image and one layer of the style image are selected from the VGG-19 model. However, it failed to provide evidence to support the layer selection. Hotaj[3] tried to visualized the image output of the layers thus support the decision. It raised our interest to study features extracted by each layer in the VGG-19 model, and to build a more customized style transfer system.

Four combinations of content and style are in Figure1.

2 Methods

First, let's briefly go through the tutorial: Neural Style Transfer with Eager Execution. We reconstruct some intermediate layers to get both content and style representations, and we will transform the

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

base input image by minimizing the content and style distance(losses) with back-propagation, where content loss is sum of the euclidean distance between the two intermediate representations, one comes from input image and the other one comes from content image; style loss is sum of the mean squared distance between feature correlation map of the style image and the input image.

The following steps are efforts we made to improve the transfer quality on the basis of the tutorial with default setting and our own images. The tutorial has introduced how to implement all steps mentioned in the Gatys et al. paper but there are some differences between the tutorial and the paper and we will discuss them later.

2.1 Replace MaxPooling with AveragePooling

VGG19 in Keras application has MaxPooling2D at the end of each block, but the Gatys et al. paper mentions average pooling will result into slightly more appealing output than max pooling. After replacing max pooling with average pooling, we do see slightly differences in detail but not obvious unless you look at them by pixel. We choose to keep average pooling as faithful to the parper and the following steps are on the basis of the replacement. The code is available in the appendix.

2.2 Overlay Content/Style on Output

Once we have an output image already, we wonder if putting the output image back again(or several times) as a new content/style image would make the new output better or not. Therefore, we get three outputs, they are (1)output from average pooling; (2)use (1) as style + original content image; (3)use (1) as content + original style image. However, we do not find significant differences among 3 outputs. The reason might be (1) does not provides unseen features to (2) and (3).

2.3 Change Content Layers

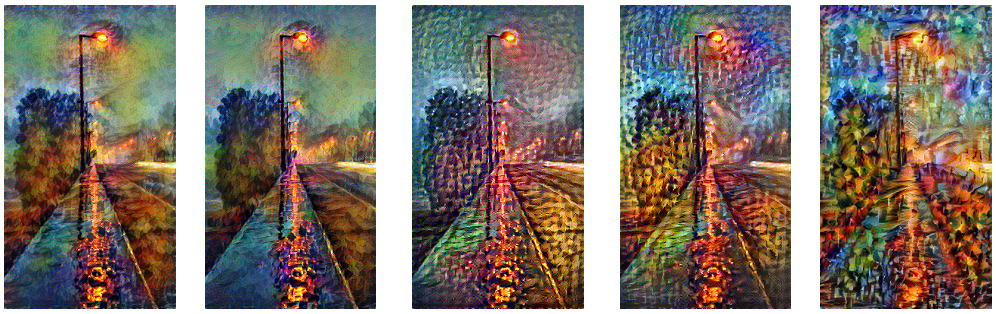
It is worth noting that the tutorial picks `block5_conv2` as `content_layers` but the Gatys et al. paper chooses to match the content representation on layer `conv4_2`(which is `block4_conv2` in tensorflow's version of VGG19), we understand that the choice of layers is subjective, but we are interested in finding out the differences between these two choices, furthermore, comparing with other applicable choices.

As the Gatys et al. paper mentions, we will used the `content_layers` to reconstruct and transform it into content representation, "lower layers simply reproduce the exact pixel values of original image", and "higher layers in the network capture the high-level content in terms of objects and their arrangement in the input image but do not constrain the exact pixel values of reconstruction." The paper proves higher layers are more appropriate to be `content_layers` but it does not specify how high these layers should be. Therefore, we compared 5 applicable choices in Figure2, which shows the outputs of selecting `block3_conv2`, `block3_conv3`, `block4_conv2`, `block4_conv3`, `block5_conv2` as `content_layers` respectively while holding other parameters all the same. It is obvious that `block3_conv2` and `block3_conv3` keep too many original pixel value like texture and color, the trace of style image is that clear. `block5_conv2` shows a strong style but it is hardly to recognize the object. Therefore, `block4_conv2` and `block4_conv3` may be good choices for `content_layers` since they balance the trace of styles and the sharpness of objects well.

2.4 Add Weights to Style Layers

Previous work of Gatys.ect compute the style loss with equally through five selected layers. To study the information extracted from the style layers, we adjusted the style weights of the five style layers. To highlight the effects, we set the weights for 5 selected `style_layers` to 10 respectively, and keep the weights for other `style_layers` small. For example, in figure3(a), we set the weight for `block1_conv1` to 10 and set the weight for other layers to 0.2. Since the style representation computes correlations between different features in different layers, adding weight to a specific layer will amplify features extract by this layer. The output can be found in figure3.

Figure3 shows that `block2_conv1` has the most saturated and warmest color. `block3_conv1` preserve the silhouette of the bulb and also the street lamps in the back. `block4_conv1` has the most detailed texture of the style image. `block5_conv1` has the coldest color and highlights the brighter



(a) block3_2 (b) block3_3 (c) block4_2 (d) block4_3 (e) block5_2

Figure 2: Outputs while Selecting Different Content Layers
(with α/β set to be 1e5, num_iteration = 1000, and lr = 5)

(a) block1_1 weight=10 (b) block2_1 weight=10 (c) block3_1 weight=10 (d) block4_1 weight=10 (e) block5_1 weight=10

Figure 3: Adjusting weights on style layers
(with α/β set to be 1e6, num_iteration = 1000, and lr = 5)

color. Compare to the images created by equal loss computation, the adjusted weights pictures tend to have subtle differences between colors and textures. Again, there's no best output for this model. The standards are subjective. The code is available in the appendix.

2.5 Adjust Weights on Content Loss and Style Loss

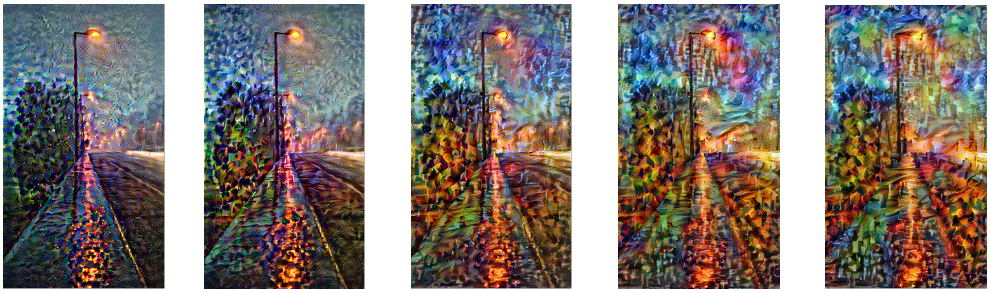
Adjusting the ratio of content loss and style loss during the training process can be informative. Process focusing on reducing style loss will tend to have output images with significant features and process focusing on reducing content loss with preserve more flavor of the original content image. Explained by equation1.

$$L_{total} = \alpha L_{content} + \beta L_{style} \tag{1}$$

It's interesting to find out that the style losses we got are around tens of millions while the content losses are around thousands. To balance the quantity, we start with $\alpha/\beta = 1e9$ and gradually update it to 1e5. Outputs are as shown in figure4.

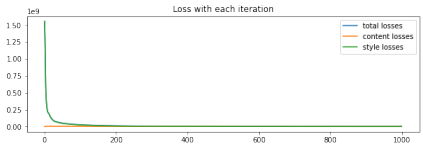
2.6 Iterations

The number of iterations determines the quality of output images together with the learning rate of the optimizer. For a constant learning rate of 5 in the Adam algorithm, we have the ideal image at the 200 iterations and there's no further improvement achieved(see Figure5a). However, we would expect a larger iteration number for a smaller learning rate as the style will transfer slower. We trained the model with 1000 iterations to maximize the power of parameters. For efficient transformation, numbers around 200 should be expected.

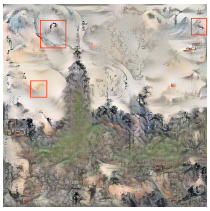


(a) 1e9 (b) 1e8 (c) 1e7 (d) 1e6 (e) 1e5

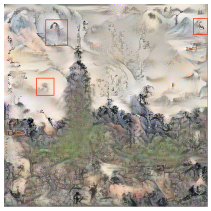
Figure 4: Adjusting weights on content loss and style loss
(with style layer equally weighted, num_iteration = 1000, and lr = 5)



(a) Total loss



(b) seed = 2020



(c) seed = 2021

Figure 5: Outputs with α/β set to 1e5, iteration = 1000 and lr = 5

2.7 Random Seeds

Randomness is introduced in the transformation with optimization. Different seeds set before optimization would lead to output images with subtle differences among colors. To amplify the effect, we set the ratio of weights on content loss and style loss to be 1e-5. The textures and overall structures will remain the same. Examples are shown in Figure5b and Figure5c. Different seeds are set to the two images on the right and marked with red boxes.

3 Results

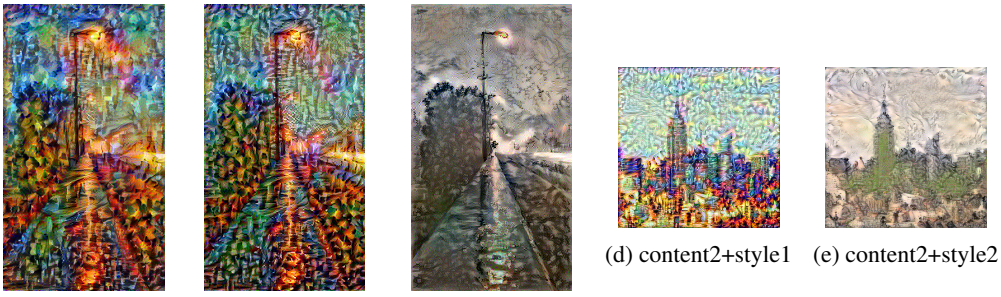
The final model chooses `block4_conv2` as the content layer, sets weights on different style layers as (0.05, 0.2, 1, 0.2, 0.2) to keep the structure of the content images with more saturated color. The ratio of content loss and style loss is set to be 1e3, learning rate equals 5 with 200 iterations. Figure6a shows the output image of the baseline model while Figure6b shows the final model. Compare to the baseline model, it is clear to see that we keep the silhouette of the objects in the background with a clear pattern from the style image. Specifically, the performance of the two models with content1 and style1 are in Table1.

The output images are highly subjective. To have the ideal transformation, following the recommendations below.

1. To keep most of the composition of the content image, relatively lower layer (higher than `block3_conv3`) are preferred.
2. Applying the style to the content image, the ratio between content loss and style loss plays a big role. Larger ratios tend to keep more painting style of the original content image.

	Running Time	Total Loss	Features
Baseline	111.73	1780934.40	smoother
Final model	23.38	1643762800.00	with clearer structure and more saturated color

Table 1: Model Compare



(a) baseline model (b) content1+style1 (c) content1+style2 (d) content2+style1 (e) content2+style2

Figure 6: Outputs

3. To add more detailed textures to the output images, give larger weights to the `block4_conv1`; to add more saturated color, give larger weights to the `block2_conv1`.
4. When efficiency is the priority, larger learning rate with fewer iterations are preferred; when image quality is more important, train the model with smaller learning rate with more iterations for fine tuning.

4 Discussion

Neural Style Transfer has a lot of applications that need to be explored. For example, some researchers are working on use Neural Style Transfer as a data augmentation method, to improve the image classification accuracy. Our research on improve the transfer quality may provide more specific features while doing augmentation.

References

[1] L. A. Gatys, A. S. Ecker and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 2414-2423, doi: 10.1109/CVPR.2016.265.

[2] Gatys , Leon A., Alexander S. Ecker, and Matthias Bethge. "A Neural Algorithm of Artistic Style." arXiv, 2015.

[3] Hotaj, Eugen. "How to Get Beautiful Results with Neural Style Transfer," March 31, 2020. https://towardsdatascience.com/how-to-get-beautiful-results-with-neural-style-transfer-75d0c05d6489.

[4] Gatys , Leon A. "Google Colaboratory." Google. Google, 2016. https://colab.research.google.com/github/tensorflow/models/blob/master/research/nst_blogpost/4_Neural_Style_Transfer_with_Eager_Execution.ipynb.

[5] Paul, Sayak. "Implementing Neural Style Transfer Using TensorFlow 2.0," June 21, 2019. https://www.datacamp.com/community/tutorials/implementing-neural-style-transfer-using-tensorflow.