ECE 590-07 Final Project Report

Hanling Su
Department of Economics
Duke University
hanling.su@duke.edu
https://github.com/HanlingSu/Duke-spring2021-ece590

Abstract

In this project, the Neural Style Transfer algorithm introduced in Gatys et al. paper was implemented to reproduce the style transferred image of Neckarfront photograph from the original paper. In addition, pictures with different levels of complexity were paired to study the effect of image complexity on style transferred outputs. Various content and style image pairs were selected to test the algorithm performance. Moreover, the trade-off between content and style matching was studied by adjusting the hyperparameter ratio. Finally, content representation performance was evaluated by choosing different layers of the convolutional neural network to extract content information.

1 Background

Convolutional Neural Networks have various applications in classification, regression, computer vision, and natural language processing areas. Its strength in object detection and classification is well proved by many researchers. Extended from this strength, Gatys et al. demonstrated in their paper[1] that deep neural networks can generate stylized images from a style painting and a content photo. The combination of art and technology is always a hot topic, so I chose this project topic to implement the algorithm and evaluate its performance.

2 Method

A publicly available pretrained VGG19 model[2] was used to implement this algorithm. As suggested by the authors, average pooling operations were used in place of maximum pooling layers. Content, style, and generated images were passed to the model, and features in specific layers were extracted as style information and content information. The difference between generated image features and content image and style image features are weighted to compute the total loss.

Gatys et al. claim content information of a given picture is well reconstructed from knowing a particular layer response of the neural network. Specifically, reconstruction from 'conv1_2', 'conv2_2', 'conv3_2', 'conv4_2', 'conv5_2' layer were shown in the original paper, suggesting detailed pixel information is lost in higher layers of the network. In this project, the 'conv4_2' layer was used in reproducing process, and the 'conv2_2' layer was selected as a comparison to evaluate this difference.

Style information or "texture information" was computed from a weighted summation of different features in different layers of the CNN. In the reproducing process, same weighting hyperparameters $w = 0.2$ were used for each layer among 'conv1_1', 'conv2_1', 'conv3_1', 'conv4_1', 'conv5_1'.

As introduced in the original paper, the style transfer process reduced to an optimization problem within a single neural network. Two types of loss functions were introduced that intended to

measure the loss of content and style between the generated image and the content input photo and the style input artwork. Two losses were weighted by hyperparameter α and β representing the trade-off between content and style matching. The total loss function was defined as $L_{total} = \alpha L_{content} + \beta L_{style}$.

More specificly, the content loss was formulated by sum of squared-error between the generated image features and the content input image features. Because only one layer was used for content representation, content loss only contain the squared-error of that layer. $L_{content} = \frac{1}{2} \sum_{i,j} \left(F_{ij}^l - P_{ij}^l\right)^2$.

Different from the origianl paper where devide each layer's squared-error by $4 \times N_l^2 \times M_l^2$, this project set number of channels(N_l) eauqals to 3 for all layers, which means the devisor is now $4 \times N_l \times M_l^2$. This change is accomendated to solve the imbalance between constantly large content loss and decreasing style loss due to increasing channel size for higher layers. [3]. The modified style loss function is: $L_{style} = \frac{1}{4 \times 3^2 \times M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2$.

The author claims no matter using white-noise, content, or style image as initialization of generated image should return similar output. However, empirical results during this project suggesting using the content image as initialization of generated result returns a more appealing output. Hence the experiment results are all initialized with content images. Detailed code implementation refers to following references[4][5]

3 Experiment results

3.1 Reproducing style transferred images in the original paper

In Figure 1, style transferred images were reproduced using the same content photo and style artworks from the original paper. The result here shows that this project successfully reproduced the original style transferred images with appropriately chosen hyperparameters.

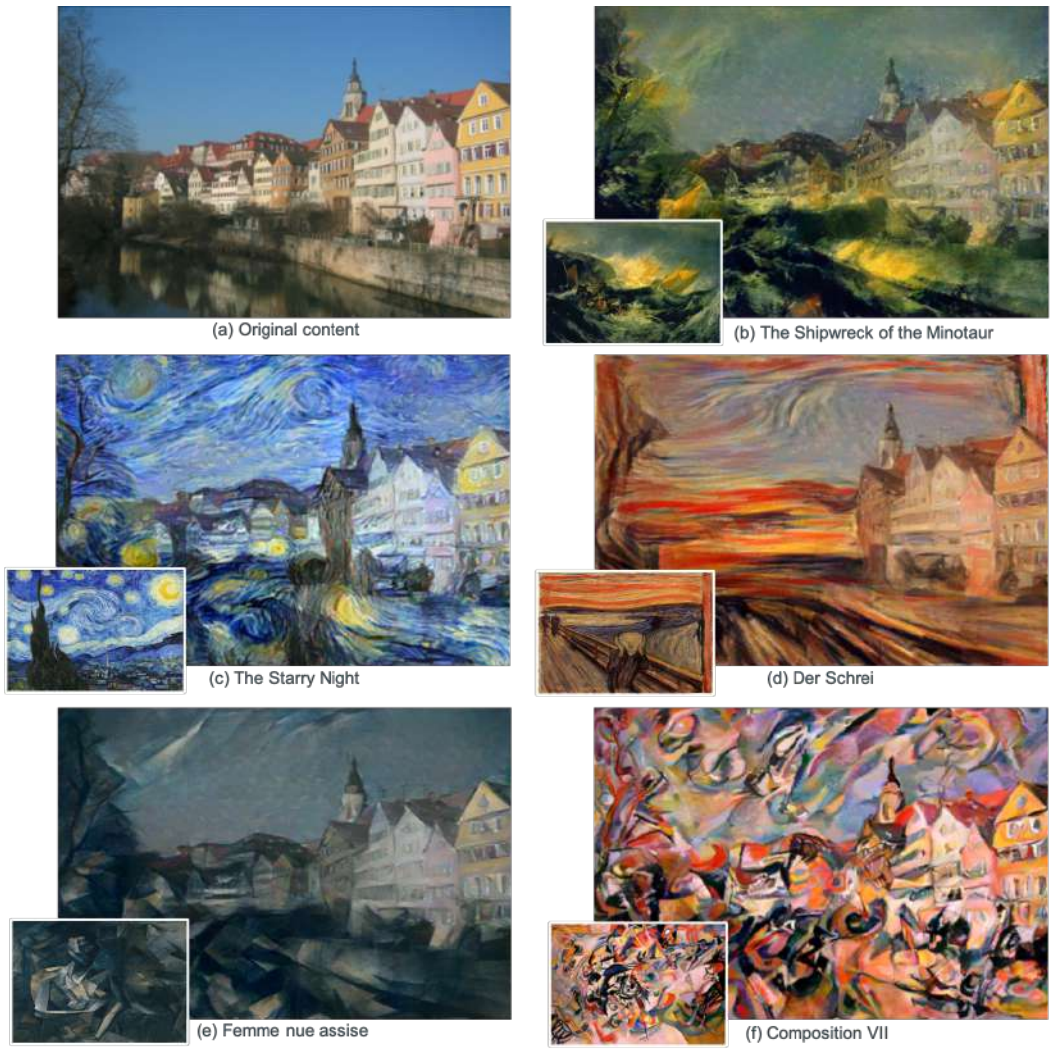


Figure 1: Images reproduced from original paper. All generated image were produced with α/β ratio $= 1 \times 10^{-5}$, for total 15000 training step using Adam optimizer with learning rate = 0.01, using 'conv4_2' layer to extract content features.

3.2 Visualization of Different Complexity Images

Multiple combinations of content and style images with different levels of complexity were used to study the effect of image complexity on results. A photo of duke chapel was selected as a complex content image, while a black and white cartoon figure of Snoopy was selected as a simple content image. *The Starry Night* by Vincent van Gogh was selected as a complex style image while a plain blue color block *PANTON 284C* was selected as a simple style image.

Several interesting observations were noticed from the results. First of all, the color theme of the style image can always be passed to the generated image. Secondly, the algorithm will detect a similar texture area between the style image and generated image, and apply style texture to the generated one. For example, with simple style, the center part of generated duke chapel(Figure 2, (c)) is blurred into an irregular blue shape, because this area has fewer clear lines or structures like windows, doors, or trees, the algorithm applied style texture to it too. Thirdly, the NST algorithm tends to maintain the clear lines of content images. For instance, in Figure 2, (e), it is clear to tell the starry style successfully delivered to the generated image, however, the clear lines of the Snoopy have been preserved with minor color change. Another interesting finding is that the area around the Snoopy lines also preserves original content background color, with the style texture observable.

This phenomenon also shows up in Figure 2 (c) and (f). The area around the shape boundary of duke chapel and Snoopy has a blurry white-noise-like color.

In general, the NST algorithm works well with different levels of complexity of style and content images.

Figure 2: Images with different levels of complexity. All generated image were produced with α/β ratio $= 1 \times 10^{-5}$.

3.3 Hyperparameter Choice

In this section, different α/β ratio($1 \times 10^{-1}, 1 \times 10^{-2}, 1 \times 10^{-3}, 1 \times 10^{-5}, 1 \times 10^{-7}, 1 \times 10^{-9}$) were applied to the duke chapel content image with *The Starry Night* style image.

α is the weight of content loss and β is the weight of style loss. With a higher α/β ratio, the generated image pertains more information from a content image. As the ratio decreases, the generated image inclines more to the style image. When the ratio decrease to 1×10^{-9}, the shape boundary line of the duke chapel looks like the star shape in *The Starry Night*.

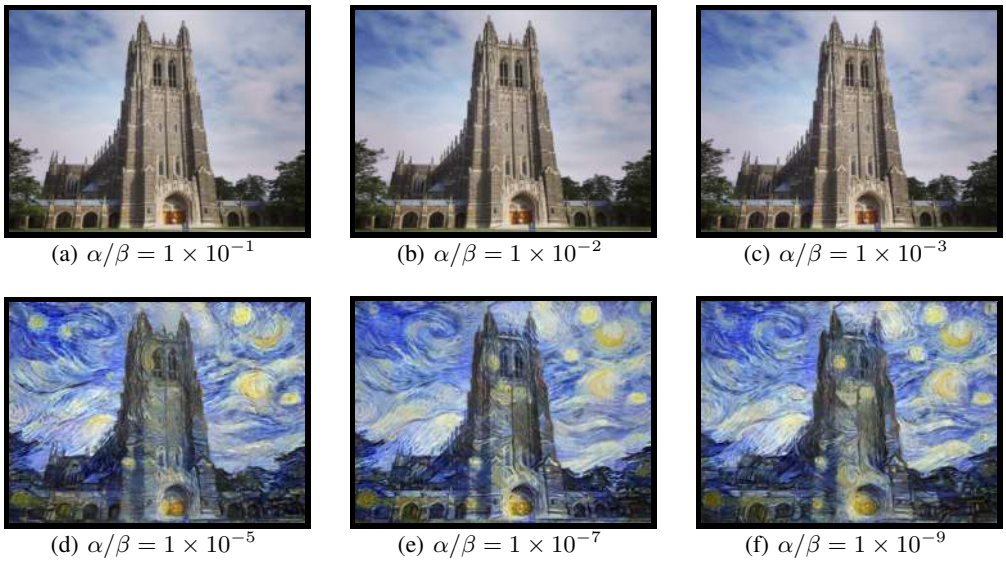
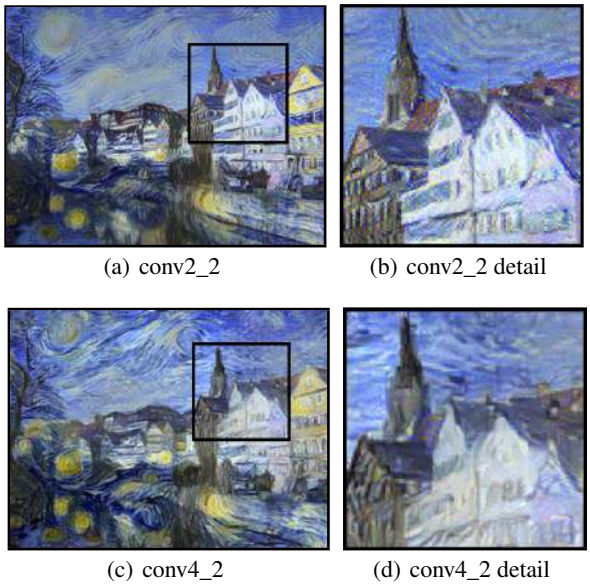


Figure 3: Generated images with different hyperparameters ratio

3.4 Results From Different Content Layers

Lower layers of convolutional neural network retain more detailed pixel information of the content image, so it helps the generated image to preserve the fine structures. Conv2_2 and Conv4_2 layers are used for comparison. Figure 4 shows, the result using Conv2_2 (Fig 4 a, b) pertains the shape of windows of original content image, while the result using Conv4_2 (Fig 4 c, d) distorts the shape of winsows into a starry texture in the style input image.



(a) conv2_2 (b) conv2_2 detail

(c) conv4_2 (d) conv4_2 detail

Figure 4: Detail of style transferred images ftom different content layer

3.5 More Visualization

Neural style transfer is a state-of-art procedure, which requires more intuitive design and explanation. During this project, I noticed that some appealing results occur when combining style and content images that depicting similar objects. For instance, when using a part of the *Dwelling in the FuchunMountains*(Fig 5, b) to transfer a photo of mountains in Zhangjiajie, Hunan China(Fig 5, a), the generated image(Fig 5, c) captures the style and pertains the content very well. Similarly, when applying *The Starry Night* to a photo of a starry night, an appealing image was generated. This result shows how well the neural style transfer algorithm works on images with similar objects.

However, when applying *The Great Wave of Kanagawa*(Fig 5, e) to two ocean photos, one with a similar waves shape(Fig 5, d), while the other one does not(Fig 5, g), the results shows style transfer does not recognize the objects by type but shape. Notice how well the style was transferred to the waves with similar shape in Figure 5, f while generating waves that does not exist in the original content image(Fig 5, i). However, it is noticeable that the waves in Figure 5, g are also transformed behind those generated curly waves.

This result suggests that the NST algorithm performs better when the style and content image contains similar objects.

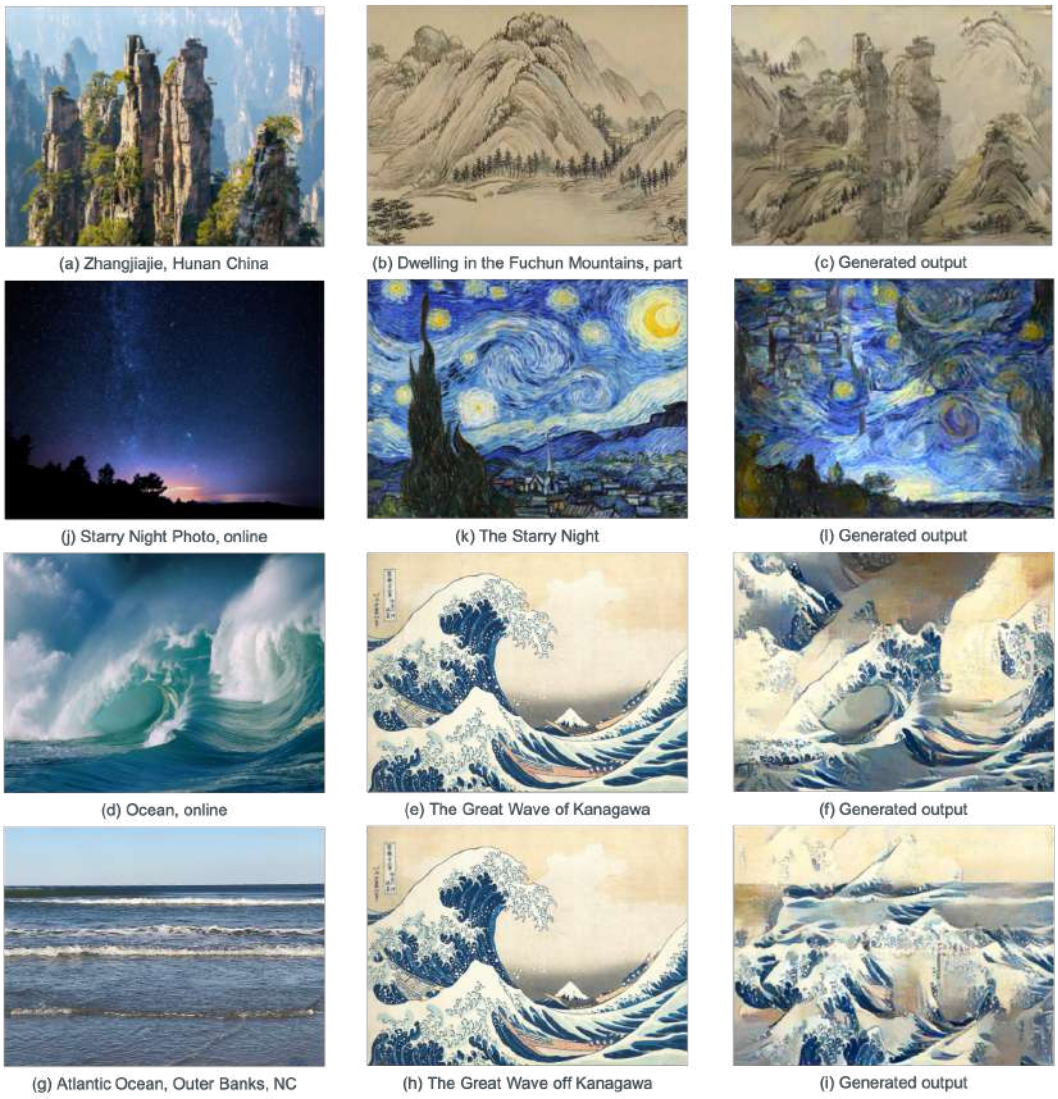


Figure 5: Style transfer process using style and content images contains similar object. All generated image were produced with α/β ratio $= 1 \times 10^{-5}$.

4 Conclusions

In this project, I implemented the Neural Style Transfer algorithm, successfully reproduced the results from the original paper, tested the performance under a variety of content and style pairs with different complexities, and studied the effect of hyperparameter choice and content layers on generated images. Furthermore, I discussed the appealing output from style and content images that contain similar objects. Convolutional Neural Networks are fundamentals of deep learning, while using them to classify pictures and process natural language, it is appreciatable that researchers put efforts into finding the potential art value of technology. Certainly, machines should not take the place of humans in artworks including painting, literature, and music, but their abilities are worth studying.

References

[1] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge, 2016, *Image Style Transfer Using Convolutional Neural Networks*.

[2] Pretrained VGG19 Model, *Pytorch*. `https://pytorch.org/vision/stable/models.html`

[3] mamrehn, 2018, *Neural style transfer with Keras, github*. https://github.com/keras-team/keras/blob/fcf2ed7831185a282895dda193217c2a97e1e41d/examples/neural_style_transfer.py

[4] Gaurav Singhal, 2020, *Artistic Neural Style Transfer with PyTorch,* https://www.pluralsight.com/guides/artistic-neural-style-transfer-with-pytorch

[5] Ritul, 2018, *Style Transfer using Deep Neural Network and PyTorch*. https://medium.com/udacity-pytorch-challengers/style-transfer-using-deep-nural-network-and-pytorch-3fae1c2dd73e

A Timeline and task allocation

I spent the first-week reading and understand the algorithm, then it took me around 3 days to implement the algorithm with help from open source code. After that, I use around 2 days to reproduce the results from the original paper, designing the loss function, deciding the initialization of generated image, and changing the hyperparameters ratio to generate appealing results. After I settled with the hyperparameters ratio, I started to analyze the choice of hyperparameters ratio, results using different pairs of content and style images, and the effect of different layers of content layers. During the project, I noticed the appealing results of combining style and content images with similar objects.