5242 Final Project-Neural Style Transfer Applications

Caspar Chen(Section2, jc5067) **Jiahao Luo(Section2, jl5182)**

Chenhao Feng(Section2, cf2743) **Yanyan Liu(Section1, yl3976)**

Abstract

This paper reviews neural style transfer applications from several aspects. This paper asks and answers questions such as how does change in parameters of representations based on a pre-trained convolutional neural network affect the visual results in synthesized images. Three content images and three different style images selected to generate nine blended images for comparison. The 19-layer VGG network model serves as the primary framework for this analysis. Modifications to the original implementation such as adjusting learning rates, exploring different initializing image, changing training iterations, experimenting different random seeds, and attempting relative weights of content and style loss in the loss function help observe the blended image difference, etc. The results from trial and error will be discussed and explained to reach conclusion and further research suggestions.

1 Introduction

In this project, we implemented the deep learning neural style transfer and Adam optimization algorithm, an extension to stochastic gradient descent, which generates artistic images by blending content image and style image together so the output image contains the objects in content image while mixed effects in the style of the style image. Ideally, the research aims to achieve a style transfer algorithm that is able to extract high-level semantic image content from the content image and comprise the semantic style from the style image to form a target image. The research is based on a normalized version of 16 convolutional neural networks and 5 pooling layers of the 19-layer VGG network. The network is a mature pre-trained model, which is developed for extracting content and style features. We used three content images and three style images and different iterations times, weight ratio, and seeds to test different combined effects in color and texture.

2 Research Purpose

We mainly want to study and answer the following questions:

- How much is the learning rate in Adam algorithm appropriate for efficient runtime and good performance in the content-style mixture?
- How do different initializing images affect the blended images after controlling variables?
- How many iterations are suitable for balancing the mixing effects in the algorithm?
- Does seed play an important role in the algorithm?
- What is the influence of art types of style images on the performance of mixed images?
- How do relative weights between content and style in total loss affect generic feature representations learned by convolutional neural network?

3 Data and Model

3.1 Images Selection

All images are shown in Figure 1.

3.1.1 Content Images

Three different content images are used in our project, which contains type of animal, objects, and architecture. Specifically, they are:

- A panda climbing on a brunch (green color)
- Car and woman in yellow background and clear sky (yellow)
- Alma mater of Columbia University (perfect for black and white contents)

3.1.2 Style Images

- One of five Van Gosh's Sunflowers are selected to be the style image. It represents impressionism in art in the 19th century. An oil painting demonstrates numerous variations of a single color. (similar color to Content-car)
- A bamboo photo is taken in photography. (similar color to Content-panda)
- A coca-cola image is chosen to represent Photorealism (red color).

Figure 1: Image Introduction

3.2 Model

According to the original paper, 5 layers are used to extract style features, containing 'conv1_1', 'conv2_1', 'conv3_1', 'conv4_1', 'conv5_1'; 'conv4_2' layer is used to extract content features.

3.3 Cost computation

The loss function was built in three steps:

- Build the content cost function

$$J_{content}(C,G) = \frac{1}{4 \times n_H \times n_W \times n_C} \sum_{allentries} (a^{(C)} - a^{(G)})^2$$

Here, n_H, n_W and n_C are the height, width and number of channels of the hidden layer you have chosen, and appear in a normalization term in the cost. For clarity, note that $a^{(C)}$ and $a^{(G)}$ are the volumes corresponding to a hidden layer's activations.

- Build the style cost function

$$J_{style}(S, G) = \sum_l \lambda^{[l]} J^{[l]}_{style}(S, G)$$

$$J^{[l]}_{style}(S, G) = \frac{1}{4 \times {n_C}^2 \times (n_H \times n_W)^2} \sum_{i=1}^{n_C} \sum_{j=1}^{n_C} (G^{(S)}_{ij} - G^{(G)}_{ij})^2 2$$

where $G^{(S)}$ and $G^{(G)}$ are respectively the Gram matrices of the "style" image and the "generated" image, computed using the hidden layer activations for a particular hidden layer in the network.

- Put it together to get total cost function

$$J(G) = \alpha J_{content}(C, G) + \beta J_{style}(S, G)$$

3.4 Procedure

- Run the content image through the VGG19 model and compute the content cost
- Run the style through the VGG19 model and compute the style cost
- Compute the total cost
- Define the optimizer and the learning rate, and then run gradient descent algorithm

3.5 Modifications to original implementation

- Tested added or deleted layers and rectifiers in content layers. (conv5_1, conv4_2, conv4_1)
- Parameters tuning, i.e. for learning algorithms, optimized the hyperparameters using gradient descent (lr-alpha, ranging from 0.003 to 2.0).
- Explored different initializing image (content image, white noise image, and style image).
- Ran different training iterations (3000, 2000, 1000, every 100 for step).
- Content and Style weight scaling (β ranging from 1e-4, 1e-3, 1e-2, 1, 10, 1e2, ... , 1e6).
- Set different seeds (0, 1, 123).

4 Result

- After training different convolutions layers and rectifiers, we find that even though the network is not deep and wide, we can still get a relatively high similarity in outputs in different training time with different number of content layers. This result shows the tradeoff between runtime and depth in convolutional neural networks. Thus, we choose to stay with the original network with conv4_2 content layer (Figure 2).

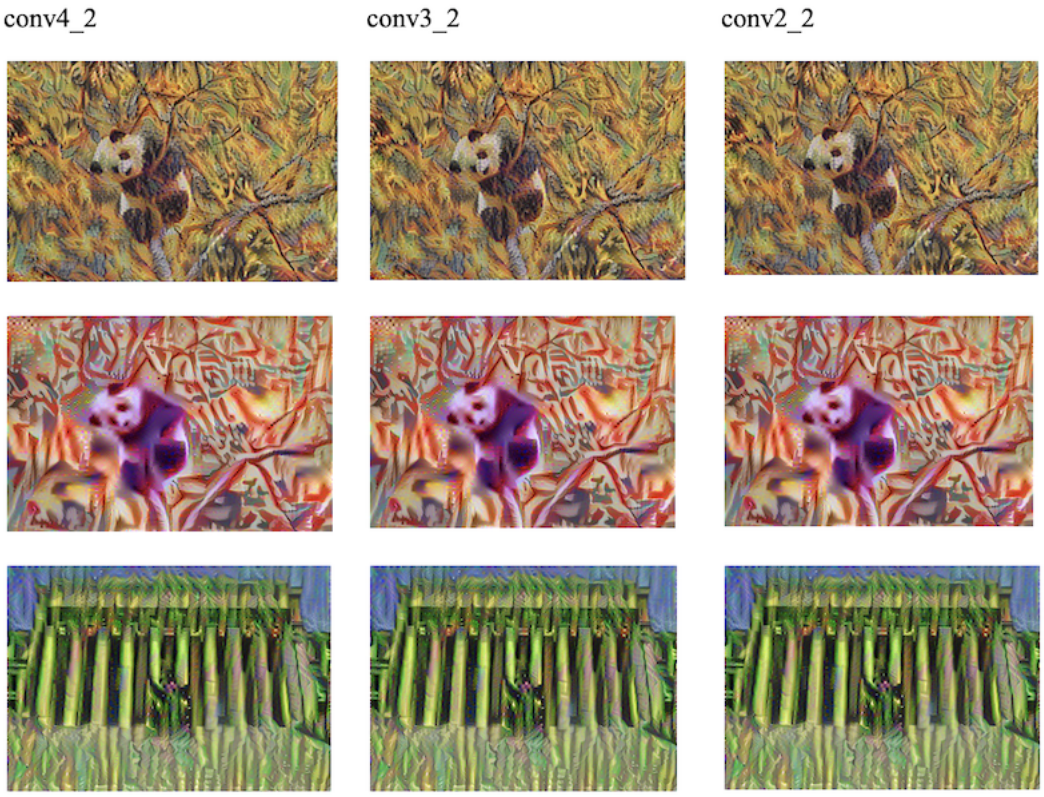

Figure 2: Content Feature Layer Comparison
P.S.Source: content, Parameters: iterations = 1000, alpha = 1, beta = 1e3

- When controlling the learning rate equals 0.003 and α/β ratio equals 1e-6, the mixed image texture becomes more blurry for 2000 or more iterations. Style becomes dominated in the background in all three contents as the iteration goes by. When controlling the learning rate equals 0.5 and α/β equals 1e-6, the image texture becomes blurry starting from 100 iterations. Thus, larger learning rate leads to quick convergence, but it may result in learning a sub-optimal set of weights too fast or getting an unstable training process. Therefore, after balancing the runtime and output performance, we decided to use 0.5 as our learning rate (Figure 3).

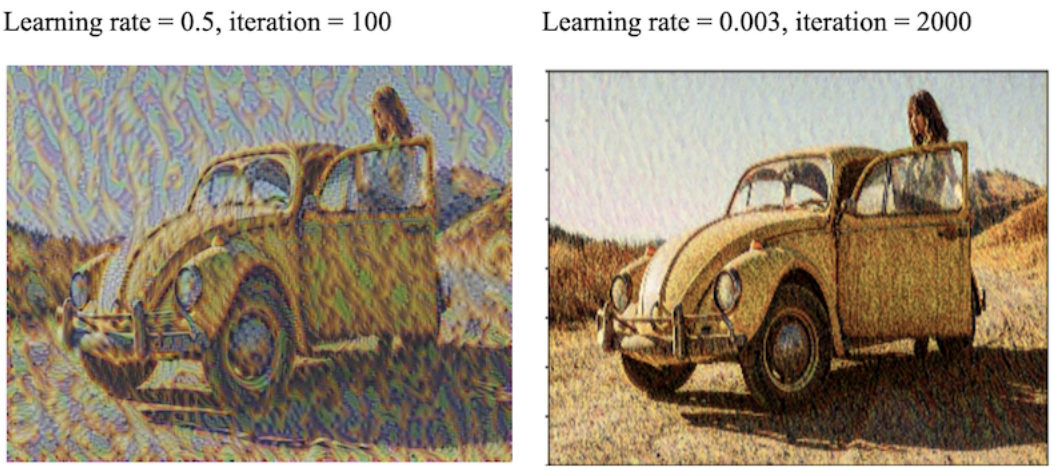

Figure 3: Learning Rate Comparison

- For different number of iterations, and fix α/β ratio to 1e2, the visual appearances for panda, car, and alma mater with bamboo, coca-cola, and sunflower style are well mixed for 500, 1000, and 2000 iterations but a slight difference in terms of brightness in color. Comparatively speaking, 2000 iterations performs with good brightness. And the running time for 2000 iterations is about 170s with GPU. Thus, we suggest 2000 iterations after controlling alpha/beta ratio to 1e2 (Figure 4).

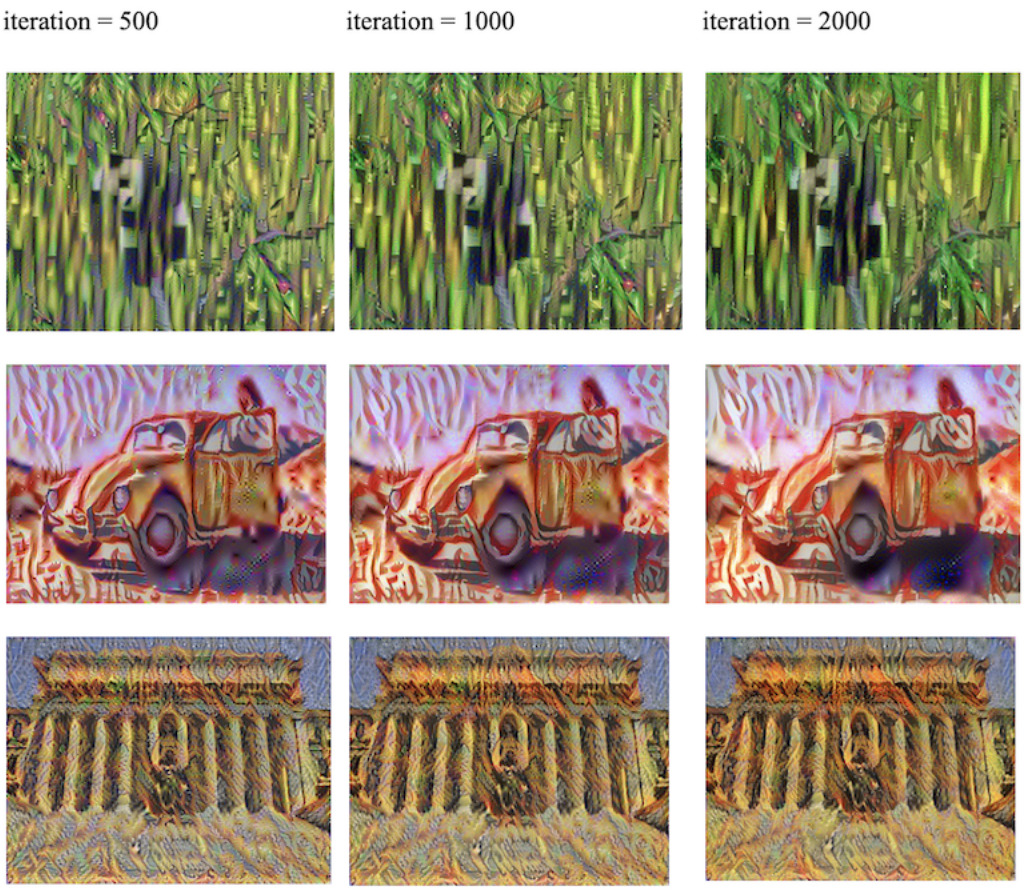

Figure 4: Number of Iterations Comparison

- After controlling iteration times to 2000 and the ratio between alpha and beta is fixed to 1e2, different initialized pictures leads to different performance. When iterated 2000 times, all three initialization: content, style, white noise successfully blended content with style. Our preferred output should contain content features with moderate style features. So we decided to start with content picture (Figure 5).

Figure 5: Initializing with different types of starting image

- When starting with content image, in this way it can preserve more content features, starting with all three initialize pictures (content, style, white noise) can produce a well performed image by manipulating alpha/beta ratio. When initializing with content picture, those ratios lower than 1 can generate a well-matched picture, when initializing with style picture or white noise, a higher weight is needed to preserve and present more content features (Figure 6).

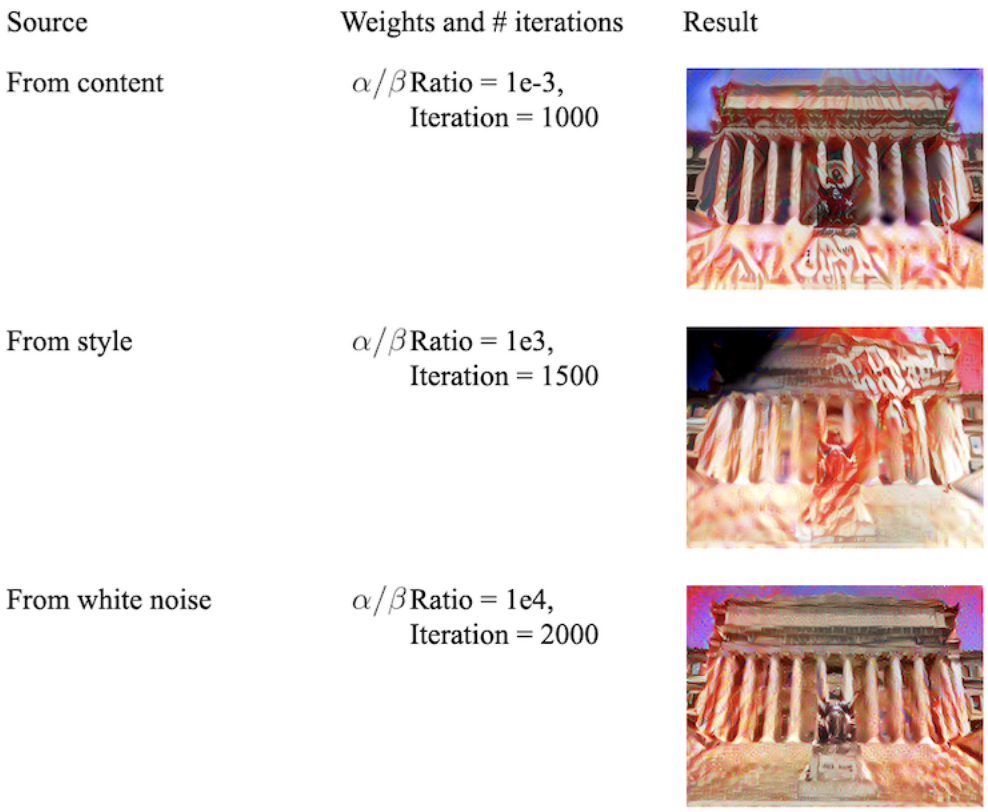

Figure 6: Different types of starting image Comparison

- Three seeds: 0, 1, 123 are attempted in this procedure, and we conclude that there is not a significant difference between mixed pictures while changing seeds, which means seeds do not play an important role in affecting outputs (Figure 7).

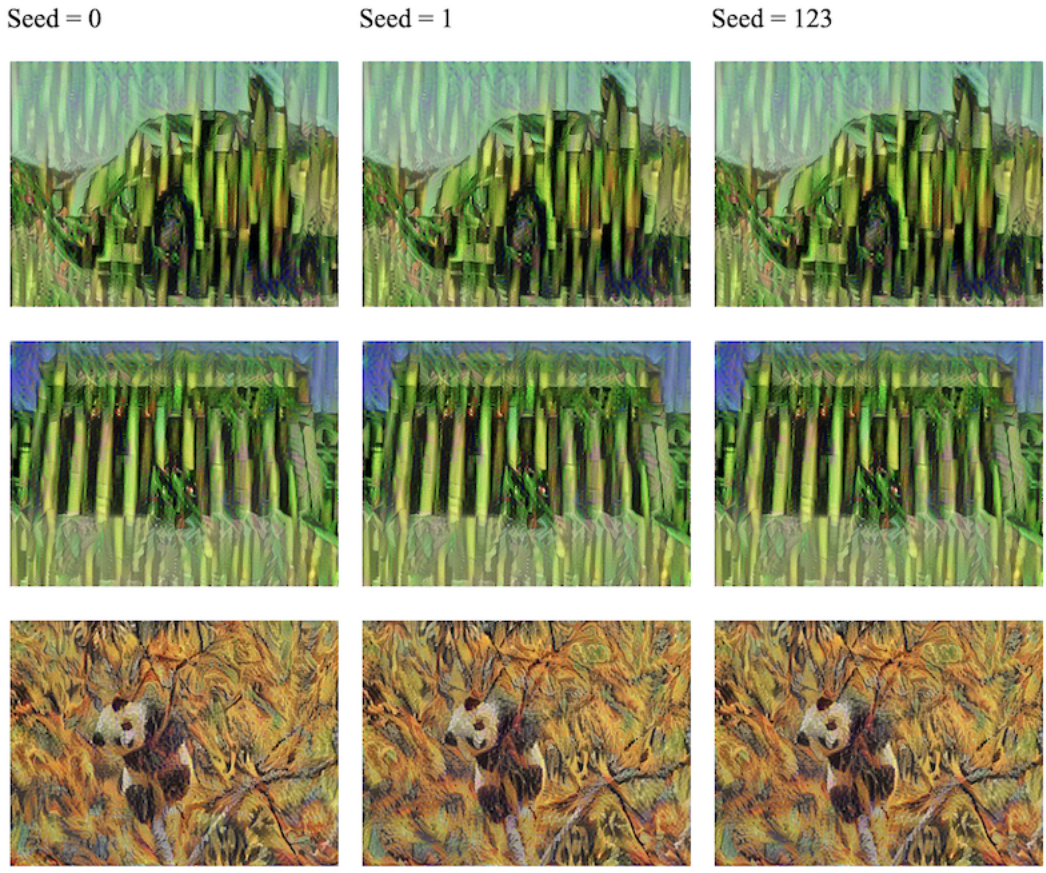

Figure 7: Random Seeds Comparison

- Different art types in style images lead to different outputs in art performance. For example, when the style image has type of photography, the mixed picture doesn't perform well in texture, for example, the combination of Content-panda and Style-bamboo reveals messiness. However, while the style image is impressionism, the art performance of the mixed picture is much better (Figure 8 middle). General Speaking, as the style picture goes more abstract, the performance of the mixed image transforms more obvious in texture (Figure 8).

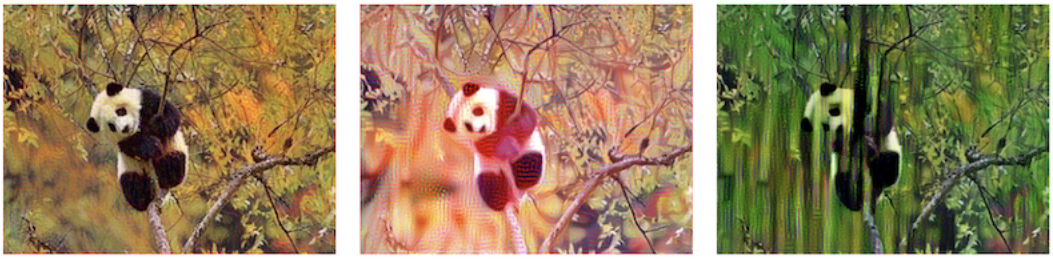

Figure 8: Different Art Types in Style Images Comparison

- While starting with the same source of image but changing different content and style weights, the results are shown as follows:
  - When initializing with content image, changing the content and style weight α/β scaling did not significantly affect the mixture performance (Figure 9).

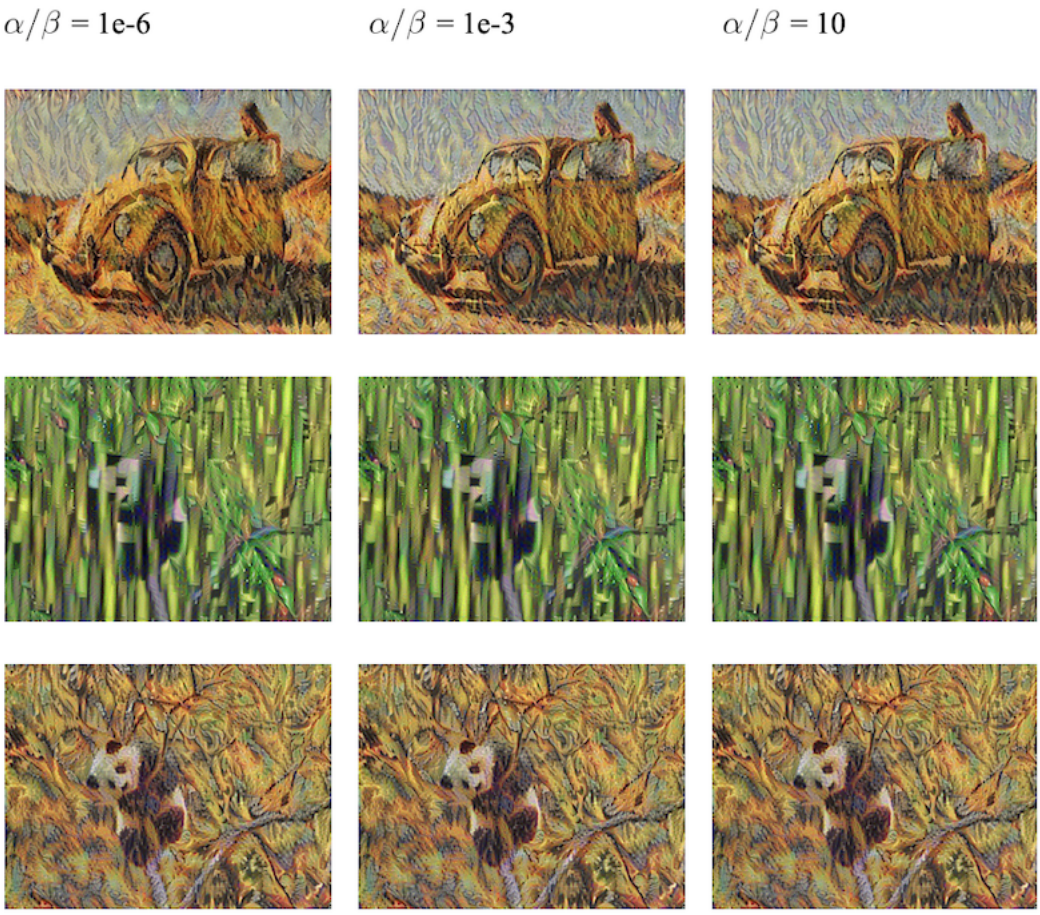

Figure 9: Initializing with content image

– When initializing starting image with white noise, the difference due to changing weights becomes clearer. When the ratio between alpha and beta is extremely small (10), the output image is closer to the style image. When the ratio between alpha and beta becomes relative large (1e4), the target image tends to be more similar to content image (Figure 10).

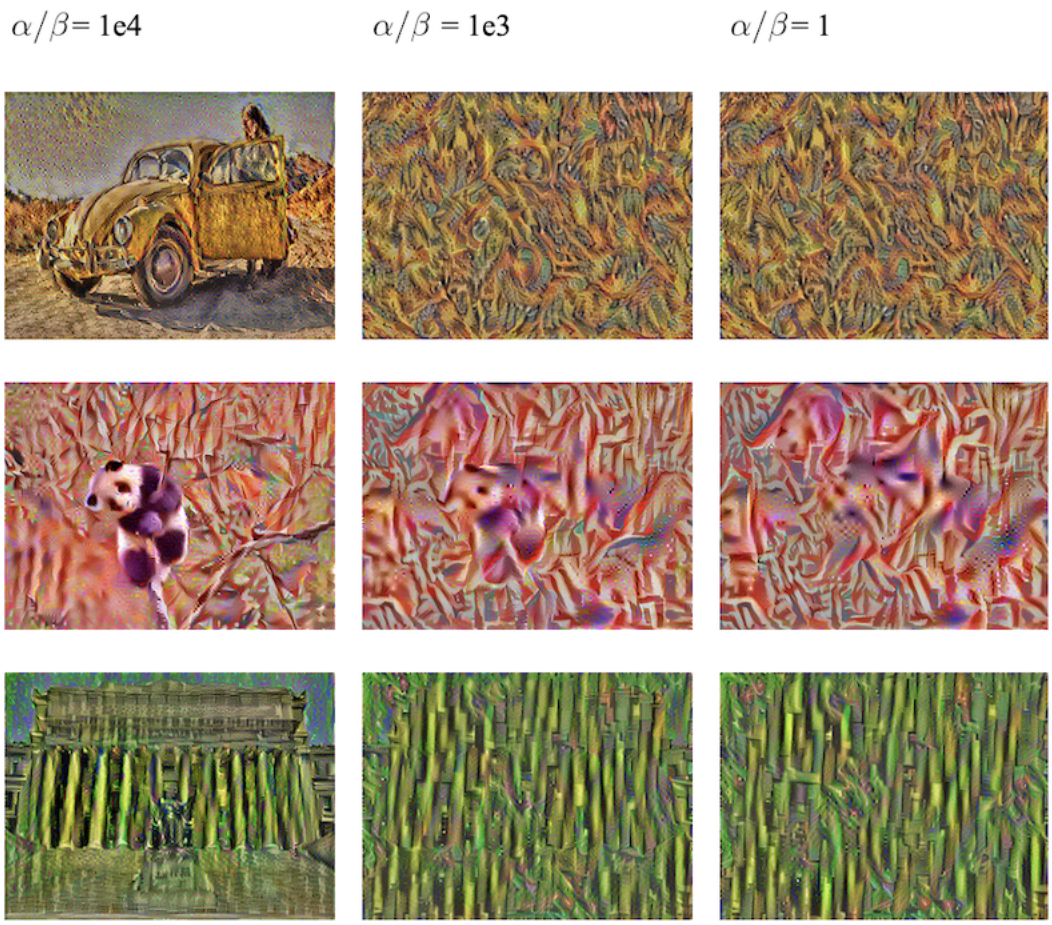

Figure 10: Initializing with white noise image

– When initializing with style image, decreasing the content and style weight α/β will make target images assemble the style image (10), and vice versa (Figure 11).

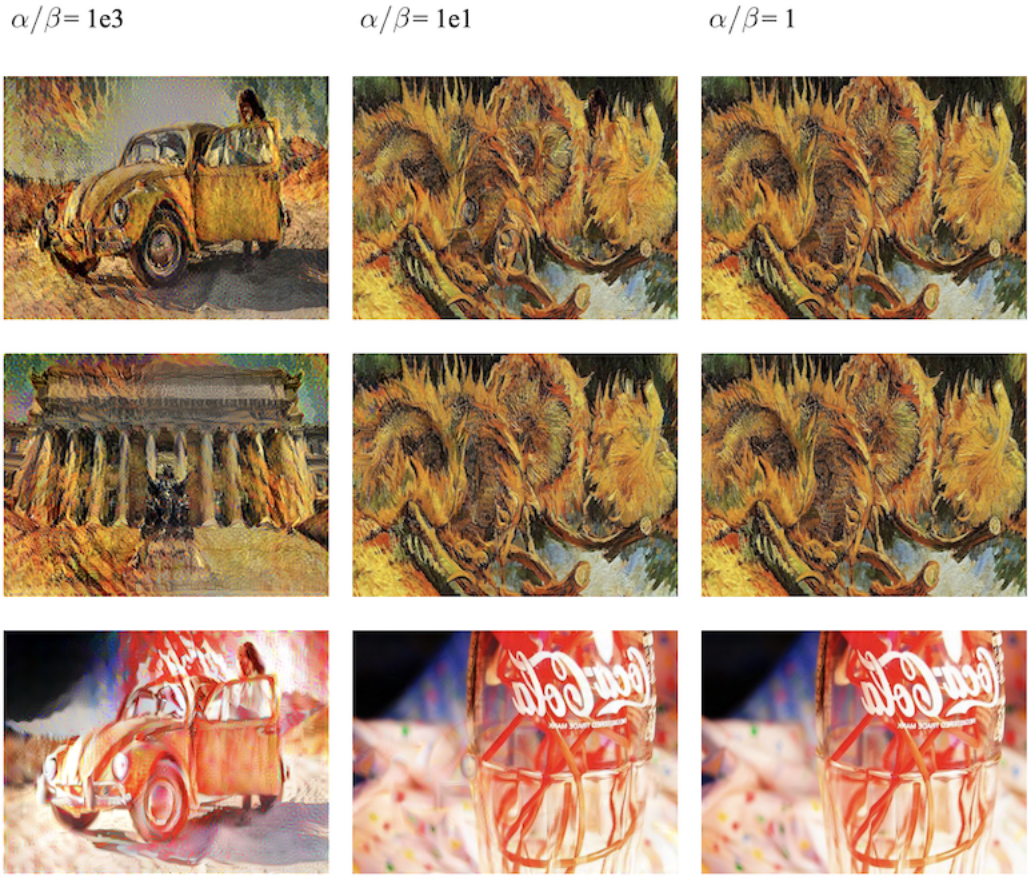

Figure 11: Initializing with style image

5 Conclusion

In this research, different layers, learning rates, iterations times, weight ratios, initializing images, art type of styles, and seeds are studied and tested. The results indicate while changing seed and layers the performance of the mixed image does not change significantly. However, changing the learning rate and iteration times can dramatically affect final performance. According to our results, 0.5 for learning rate and 2000 iteration times lead to the best performance. Additionally, art types of style image can also affect the performance of the output image. As the style picture goes more abstract, the performance of the mixed image transforms more obvious in texture.

Finally, the weight ratio between content features and style features and initializing images played an important role in the output performance. While initializing with content image, low weight ratios emphasize on the style in texture effects of the blended image, in contrast, high weight ratio emphasizes the content with only little stylization in blended image. Therefore, a lower ratio leads to a better performance since it balances the content and style. On the other hand, while starting with style image or white noise, lower weight ratios ignore all content features, which produce bad mixture result.

6 Future Work

- Explore more pre-trained image classification models like VGG-16 (Simonyan and Zisserman 2014), InceptionV3 (Szegedy et al. 2015), and ResNet5 (He et al. 2015).
- Apply color preservation for different content and style images using a mask.
- Add Multiple Style Transfer for one content image but several style images.

- Transfer videos instead of images

References

[1] Gatys, L. A., Ecker, A. S., and Bethge, M. (2016). Image style transfer using convolutional neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition(pp. 2414-2423).