Neural Style Transfer

Haosheng Ai
Department of Statistics
Columbia University
New York, NY 10027
ha2583@columbia.edu

Zi Fang
Department of Statistics
Columbia University
New York, NY 10027
zf2258@columbia.edu

Changhao He
Department of Statistics
Columbia University
New York, NY 10027
ch3557@columbia.edu

Weiwei Song
Department of Statistics
Columbia University
New York, NY 10027
ws2621@columbia.edu

1 Introduction

1.1 Main problem description

The main problem that we are trying to solve is synthesizing the texture information from a content image and a style image. To construct a style transfer model, we build style representation and image representation by using a pre-trained model, which is a 19-layer VGG network. Then we create a new image by rendering the style presentation on the content representation. We apply a loss function on that, minimize the loss, and utilize gradient descent on the output to implement optimization. Taking four different combinations of test image sets to the defined style transfer model by choosing two content images and two style images is the first thing to do for testing. After that, we need to try different training iterations, random seeds, and weight coefficient combinations in the loss function to find out possible influences that the changes would bring.

1.2 Related references

[1] This paper is the guidance of the whole algorithm neural style transfer. It introduces what the neural style transfer is, how it works and what affects the performance. The author Gatys specifically explains how to use convolutional layers to extract the content representation and style representation and how to use that to create a new picture with both content and style extracted from two different pictures or artworks. The loss function for content features and style features and total loss as the weighted version of content loss and style loss. Then the paper uses examples to demonstrate how iteration number, random seed, and relative weight of content and style will influence the output.

[2] This is a coding example which introduces how to do a neural style transfer described in Gates's paper in practice in tensorflow using pre-trained 19 layer models. We refer to this to see how to do image preprocessing.

[3] This is an online resources that introduce the convolutional neural network and VGG Netowork.

[4] The post in the Medium is a tutorial that includes both detailed explanations and codes to introduce how to implement neural style transfer with eager execution. We refer to the loss function parts and optimization parts from the authors in our own notebook.

Preprint. Under review.

[5] This is an online resource we refer to as how to combine a content image and white noise image to create an input image for the training process. Specifically, we refer to the code block 51.

1.3 Data explanation

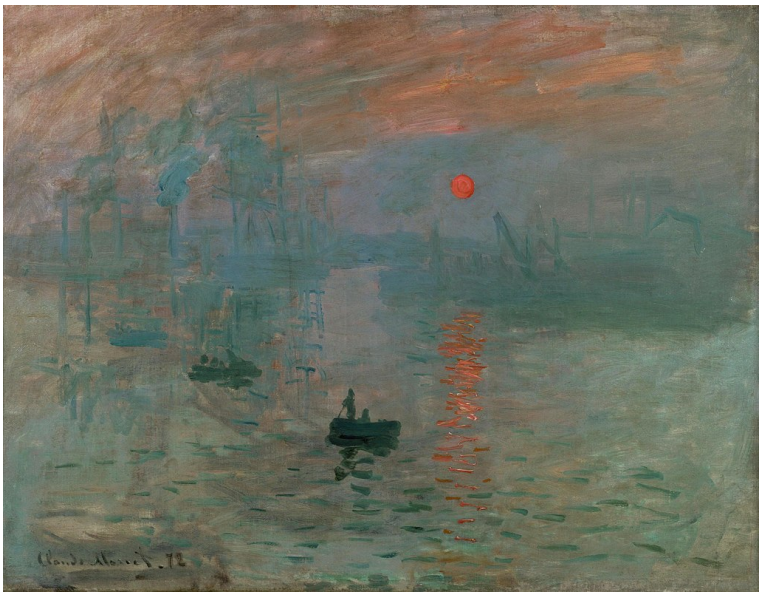
Georges Seurat is best known for devising the painting techniques known as chromoluminarism and pointillism. His large-scale work A Sunday Afternoon on the Island of La Grande Jatte (1884–1886) is a masterpiece that altered the direction of modern art to neo-impressionism.

A Sunday on the La Grande Jatte By Georges Seurat

Source:

`https://commons.wikimedia.org/wiki/File:A_Sunday_on_La_Grande_Jatte,_Georges_Seurat,_1884.jpg#/media/File:A_Sunday_on_La_Grande_Jatte,_Georges_Seurat,_1884.jpg`

Impression, Sunrise is a painting by Claude Monet. This painting is special because it inspires the name of the impressionist movement.



Impression, Sunrise by Claude Monet

Source:

https://commons.wikimedia.org/wiki/File:Monet_-_Impression,_Sunrise.jpg#/media/File:Monet_-_Impression,_Sunrise.jpg

We wonder what it would be like if we could use the style of these two famous painters to create an artwork to depict our campus. We choose two well-known spots, one is a statutory Alma Mater and the other is a building Butler Library.

Alma Mater

Source: https://commons.wikimedia.org/wiki/File:Almamater.jpg#/media/File:Almamater.jpg

Butler Library

Source:https://commons.wikimedia.org/wiki/File:Butler_Library_-_1000px_-_AC.jpg#/media/File:Butler_Library_-_1000px_-_AC.jpg

Four Images including two style images and two content Images. Our task is to use a pre-trained 19-layer VGG model to test the four resulting combinations. Since the model is pre-trained, there are no training or validation samples in this project. We only have four testing samples. The inputs are a content image and a style inference image, the output will be a new image which is a blend of

content image but with the painted style from style reference image. The dimension of both input and output is 340*512*3 after preprocessung.

2 Methods

2.1 Starting Point

Our work started by applying the pre-trained model (19-layers VGG network) on images and extracting both the content features and styles features. This pre-trained model includes 16 convolutional and 4 pooling layers aiming on object recognition and localization from different depth of CNNs. Then we set loss functions for style and content by calculating "how far" between target image and the input images. And finally, we optimized on the target image by reducing over total loss to reach the ideal image through iteration.

2.2 Architectures setups

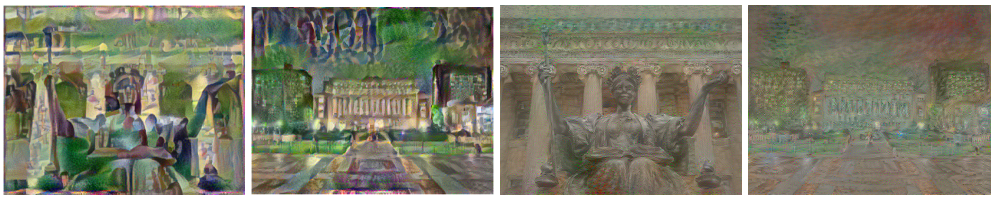
To reach the goal, there are multiple steps and details that needs to be implemented. We first need to preprocess input images, reshape the images to target size and convert to arrays. In the pre-trained 19-layer VGG network, we use top layers of each block (block1_conv1, block2_conv1, block3_conv1, block4_conv1, block5_conv1) to filter the style of input images and use deeper layers like block5_conv2 to filter content of input images.

We also try to implement the loss function by combining the style loss and content loss as the total loss. The content loss directly calculated the sum of square error of each pixel between the content features and the features of our generated image. Whereas, the style loss calculated the mean square error between the the style representations of the style image and the generated image. The style representations computes the correlation of different filters in the same layer, so the summation of style representations in each layer can provide the texture information of the style image[1]. Therefore, by applying different weights on content loss and style loss in the loss function, we can create an image which can show the contents from the content image clearly and also maintain a similar style and texture compared to the style image.

During the optimization process, we need an initialized target image which can be the mixing of the content image and a white noise image, a white noise image, the content image or the style image. And we also need the a content image and a style image. Then we choose Adam optimizer to run gradient descend on the initialized target image to minimize the total loss in the loss function defined above. With training in a thousand of iteration, we successfully combine the inputs picture to the target picture. We also used different random seeds to generate the initial image, different training iterations and different weights on the content loss and style loss. Some of these hyper parameters have significant effect on the target image.

3 Results

3.1 Four combinations

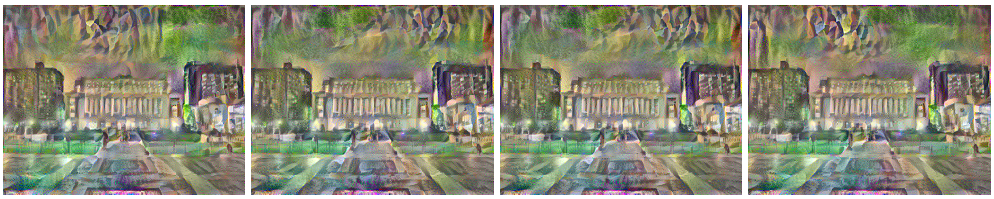


The first image is A Sunday on the La Grande Jatte combined with Alma Mater. The second image is A Sunday on the La Grande Jatte combined with Butler library. The third image is Sunrise combined with Alma Mater. The last image is Sunrise combined with Butler library.

Since there are no labels for the output image, we cannot decide if the output image is successful or not numerically. Our intended result should present recognizable contents from the content image, and also integrate styles such as color map, painting style and scenes from the style image. The ideal

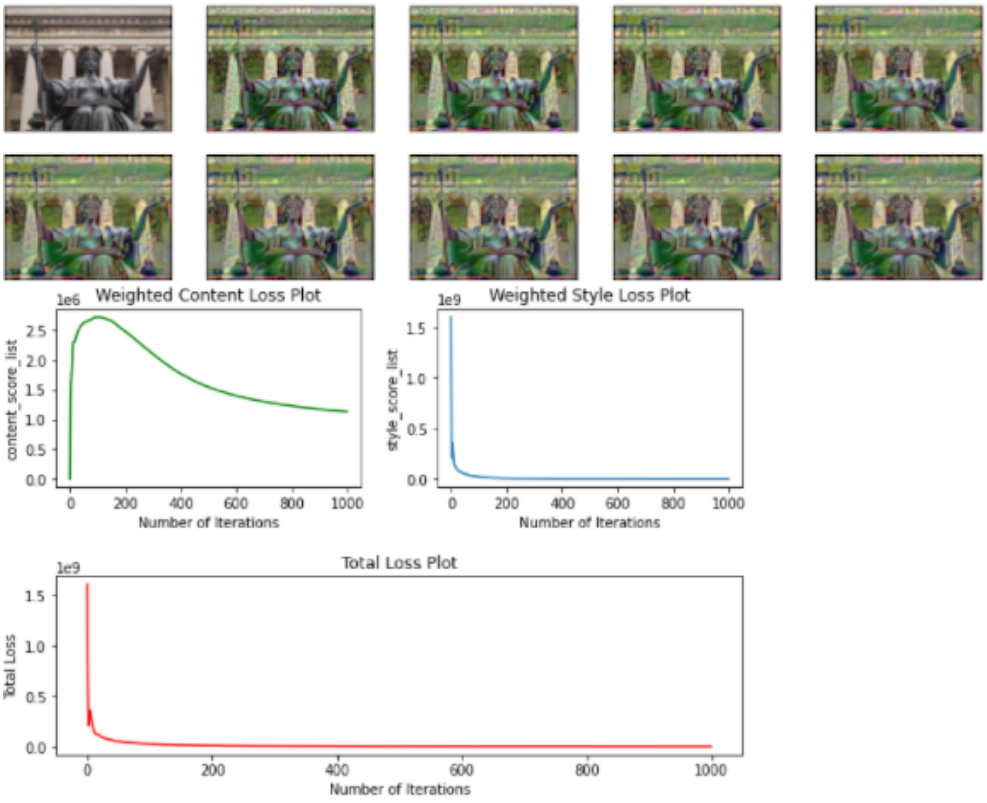
result image should be the most aesthetic combination. For example, the first and second images above are style image and content image respectively. And the third image is an ideal style transfer image.

3.2 Different random seeds



We tried different random seeds 10,50,100 and 1000 to generate four white noise images with uniform distributed value from -20 to 20 on each pixel. Then generated four input images by using (0.5 * RGB of the white noise images) + (0.5 * RGB of the content image). Treating these four input images as the initial images of the learning process, we finally got four combined images. Comparing these four images, the overall structure of the contents and textures are quite similar except some slight differences on the color maps. Therefore, randomly initializing the input images will somehow influence the output, but will not have too much effect on it.

3.3 Different training iterations

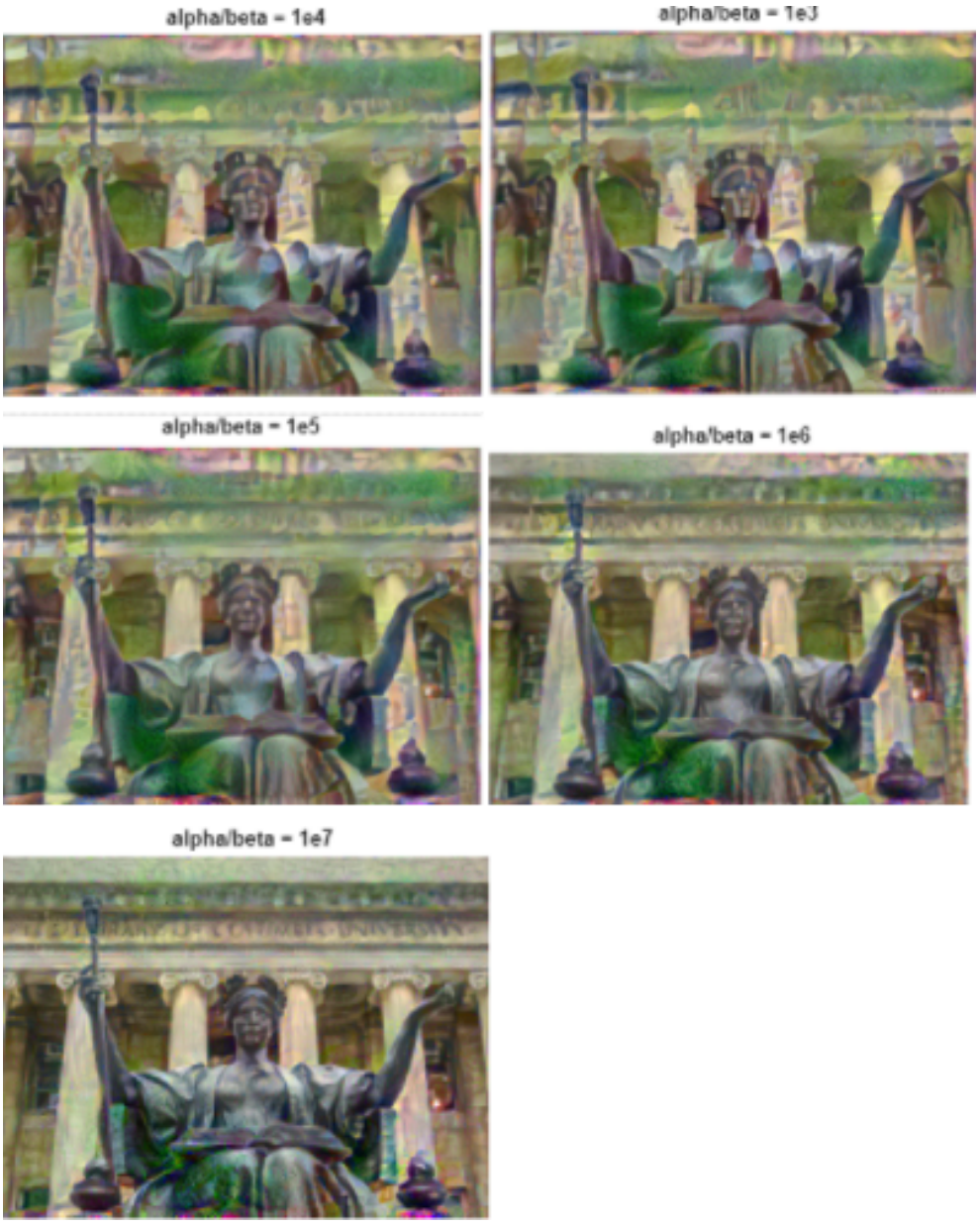


We will use content image: A Sunday on the La Grande Jatte and style image: Alma Mater as an example to make illustration, and results of running the other three combinations are similar to the example.

In the example, we set alpha/beta = 1e4, and run for 1000 iterations to see how the output images change per 100 iterations. From the ten generated images, when the number of iterations increases, the style representation has a greater influence on the generated image.

Then, let's turn to the three loss plots. The weighted style loss and total loss have similar shapes. Both loss values drop fast as the number of iterations increases from 0 to 1000, have a small peak at the beginning of the iteration increasing processes, and then the graph tends to be flat with values close but greater than zero. For the content loss plot, the loss value starts at zero, which is reasonable since we use the content image as the initial generated plot for the trial. As the number of iterations increase, the weighted content loss values first increase dramatically, and then start to decrease at around iteration 130, but the decreasing pace is moderate.

3.4 Different relative weight of content and style loss



Remember that the neural style transfer algorithm helps separate style representation and content representation from the chosen style image and content image. In order to quantitatively compare the differences between the generated image and the test image sets, content loss and style loss are introduced, and by combining them with the formula alpha * content loss + beta * style loss, total loss value can be calculated.

We keep using the content-style image combination stated previously to illustrate the influence of changing relative weight, and similar results can be found if running the other three combinations.

In the total loss formula, alpha and beta represent the weights for content and style reconstruction loss. When we increase the alpha/beta ratio, the shape of the three loss plots are similar to the previous example, but we can easily find that the decreasing rate of weighted content loss plot increases. Also, from the generated plots, as the alpha/beta ratio increases, the final output looks more like the content image, and it contains more content representation information than the style information extracted from the style image.

4 Discussion

4.1 Takeaways and discussion for future work

The important takeaways are that the work helps us clarify useful references, understand the processes, choose needed test image sets, and organize thoughts. We could treat the style image as a filter and be applied to the content image. Also, when applying the model, the phenomenon of preserving color and texture information of the style image can be normally seen. The problems/opportunities for further work are performing the neural style transfer step by step takes some time, a faster way of performing the task will be prefered. The pre-trained network, such as 19-layer VGG network might not be the optimal network, so trying different networks, editing the existing network, or designing a new network could be a possible next step.

References

[1] Gatys, L. A., Ecker, A. S., and Bethge, M. (2015). A Neural Algorithm of Artistic Style. arXiv preprint arXiv:1508.06576.

[2] Neural style transfer | TensorFlow Core. (n.d.). TensorFlow. Retrieved December 12, 2021, from https://www.tensorflow.org/tutorials/generative/style_transfer

[3] Singh, M. (2017, September 4). Artistic Style Transfer with Convolutional Neural Network | by Manjeet Singh | Data Science Group, IITR | Medium. Medium; Data Science Group, IITR.https://medium.com/data-science-group-iitr/artistic-style-transfer-with-convolutional-neural-network-7ce2476039fd

[4] Neural Style Transfer: Creating Art with Deep Learning using tf.keras and eager execution | by TensorFlow | TensorFlow | Medium. (2018, August 3). Medium; TensorFlow. https://medium.com/tensorflow/neural-style-transfer-creating-art-with-deep-learning-using-tf-keras-and-eager-execution-7d541ac31398

[5] SreeHarshaNelaturu. (n.d.). GitHub - SreeHarshaNelaturu/Neural-Style-Transfer: Tensorflow Implementation of Neural Artistic Style Transfer using VGG-19. GitHub. Retrieved December 19, 2021, from https://github.com/SreeHarshaNelaturu/Neural-Style-Transfer