Using VGG19 for Neural Style Transfer Learning

Yuqing Chen
Department of Statistics
Columbia University
New York, NY 10027
yc3753@columbia.edu

Levi Lee
Department of Statistics
Columbia University
New York, NY 10027
ll3248@columbia.edu

Yue Liang
Department of Statistics
Columbia University
New York, NY 10027
yl4391@columbia.edu

Michael Petkun
Department of Statistics
Columbia University
New York, NY 10027
mjp2262@columbia.edu

Abstract

In this project, we use the VGG19 neural network for neural transfer learning, in order to combine two images by extracting the content of one and the style of another. We create our own model with adjustable hyperparamters and with our choice of content and style images, and we explore and discuss how changes in target image initialization, random seed, content loss weight (α), style loss weight (β), and training steps affect the target image. We also experiment with many different style images, finding that styles with distinctive color palettes and textures work best in our model. We end with a summary of key takeaways and a discussion of avenues for future research.

1 Introduction

One of the interesting applications of computer graphics is to transfer the style from one image to another. Image style transfer algorithms can be applied to photo and video editing software, enabling users to create artworks with various styles and contents without needing to know much about art. Yet style transfer is considered a difficult image processing task, due to the lack of explicit semantic information in images. Recent advances in studies of deep convolutional neural networks have provided some ideas on performing such a task. It is shown that convolutional neural networks trained for object recognition extract high-level semantic information from images in generic feature representations, which can be used for the style transfer task. [9]

In this project, we implement a neural style transfer algorithm based on Gatys, et al. (2016) [4] and various TensorFlow [1] [3] [7] and Kaggle [2] [10] tutorials.

Gatys, et al. (2016) [4] discussed how the VGG19 neural network can be used for neural transfer learning by using specific layers to represent the style and content of an image. Several TensorFlow [1] [3] [7] and Kaggle [2] [10] tutorials have outlined the steps to set this process up using Python 3–see TensorFlow's "Neural Style Transfer" and "Fast Style Transfer for Arbitrary Styles" [1] [3] [7] as well as Basu's "Style Transfer Deep Learning Algorithm" [2] and Surma's "Style Transfer" [10] on Kaggle for more details. We have taken elements and ideas from each of these guides to implement the process described by Gatys, et al. (2016) [4] in Google Colab, and we have created a tool that allows us to easily switch between different options and hyperparameters, including the choice of initialization, random seed, style loss weight, content loss weight, optimizer, and training steps.

Preprint. Under review.

Figure 1: Our choice of content images (top row) and style images (bottom row). (a) Top left: *Alma Mater* statue. Source: Image Reference [1]. (b) Top right: Columbia University's Morningside campus. Source: Image Reference [3]. (c) Bottom left: Portrait of Homer J. Simpson. Source: Image Reference [5]. (d) Bottom right: Jackson Pollock's *No. 5*. Source: Image Reference [6].

1.1 Transfer Learning with VGG19

To represent the features of various images, we applied transfer learning using weights from the VGG19 model, a 19-layer version of the network architecture developed by VGG (Visual Geometry Group) from the University of Oxford. [6]

The weights in the VGG19 model were trained on ImageNet, which is a database with millions of images organized according to the WordNet hierarchy. [5] Although VGG19 is typically used for classification purposes, the layers of VGG19 can grasp important features of an image. [8]

As suggested in Gatys, et al. (2016) [4], we used one high-level convolutional layer of the VGG19 model to represent an image's content and five other convolutional layers across various levels to represent an image's style. In our implementation of transfer learning, we froze the pretrained layer weights of these VGG19 layers, so that we can apply the VGG19 model's weights to our images, extracting the content and style information.

1.2 Style and Content Images

In the spirit of Columbia, we have decided to use photographs of the *Alma Mater* statue and Morningside campus as our content images. For our style images, we chose two contrasting images: one is a more traditional notion of art–Jackson Pollock's abstract work *No. 5, 1948*–and one a more modern-day work–a portrait of Homer J. Simpson, from the popular television show *The Simpsons*. Please see Figure 1. We feel that this contrast of color, linework, and style help to better understand the general performance of the VGG19 neural network and our algorithm. We also utilize other style images–such as Caravaggio's *Supper at Emmaus* and Dali's *The Persistence of Memory* as examples that highlight the limitations of our model. Thus, these images serve as the data for our project.

2 Methods

2.1 Data Preprocessing

For both the content image and the style image, they are by default RBG images. Using TensorFlow, we preprocessed both images to create a three-channel tensor, with each channel representing one layer of color. Then we resize them as 512×512-pixel images.

2.2 Content Representation

Generally, images are represented in convolutional neural networks by the filter responses of each layer of the network, where feature representations of lower levels focus more on precise appearance or the exact pixel values, while those of higher levels capture more information about the objects and their arrangement. Since convolutional neural networks extract more explicit information about the content of an image at higher layers, we use the block5-conv2 layer from VGG19 to represent the content.

The content loss function is defined as follows:

For a layer l with N_l distinct filters, suppose the N_l feature maps are each of size M_l, and the responses in this layer are stored in a matrix $F_l \in \mathcal{R}^{N_l \times M_l}$, where M_l is the height times the width of the feature map and F^l_{ij} is the activation of the i^{th} filter at position j.

Denote the original content image and the generated image as $\vec{p}$ and $\vec{x}$ respectively, and their respective feature representations in layer l as P^l and F^l. The content loss is then defined as

$$\mathcal{L}_{\text{content}}(\vec{p}, \vec{x}, l) = \frac{1}{N_l M_l} \sum_{i,j} (F^l_{ij} - P^l_{ij})^2$$

Since this implementation is different from Gatys, et al. (2016) [4] in terms of scaling, our choice of the weight α associated with the content loss will be different.

2.3 Style Representation

The style representation is a bit more complicated in that we need to extract information on the texture of the image from feature representations or filter responses that represent the objects. In this model, the Gram matrix of a given layer, consisting of inner products between different filter responses, is used to capture the texture information and to represent the style of an image. Such style representation can be constructed from multiple layers in the network, since both higher and lower level features contain information that can contribute to our modeling of the style of an image. We therefore use Gram matrices based on the block1-conv1, block2-conv1, block3-conv1, block4-conv1, and block5-conv1 layers from VGG19 to represent the style.

The style loss function is defined as follows:

The Gram matrix $G^l \in \mathcal{R}^{N_l \times M_l}$ is defined such that G^l_{ij} is the inner product between the vectorized feature maps i and j in layer l:

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}$$

Denote the original style image and the generated image as $\vec{a}$ and $\vec{x}$ respectively, and their respective style representations in layer l as A^l and G^l. The contribution of layer l to the style loss is defined as

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

The total style loss is then calculated as the average contribution from each layer. Denoting the number of layers as L,

$$\mathcal{L}_{\text{style}}(\vec{a}, \vec{x}) = \frac{1}{L} \sum_{l=1}^{L} E_l$$

Since this implementation is different from Gatys, et al. (2016) [4] in terms of scaling, our choice of the weight β associated with the content loss will also be different.

2.4 Total Loss Function

The general idea of the style transfer algorithm is to generate a target image that is close to the style image in their style representations and to the content image in their content representations. Therefore, the algorithm tries to minimize the weighted sum of content and style loss:

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

2.5 Optimizer Implementation

Gatys, et al, (2016) [4] used L-BFGS for optimization, but our implementation used Adam with learning rate of 0.1. From our testing, we found that this choice of optimizer gives us the best empirical results.

2.6 Training the Model

We implemented the algorithm described above by building a function that allows us to change important parameters such as the content and style images, method of initializing the target image, random seed, number of training steps, and weights of content and style loss. Using this function, we are able to train the model with different parameters and output the resulting image for visualization.

3 Results

We find that the algorithm described above does indeed allow us to combine the content of a given image with the style of another image. To demonstrate this, Figure 2 displays each of the four combinations of the content and style images mentioned previously. These images were all generated using the content image to initialize the target, running 100 training steps with $\alpha = 1$ and $\beta = 1 \times 10^{14}$. These parameters will be varied in later sections.

Figure 2 (a) combines the content of the *Alma Mater* statue picture with the Homer Simpson style. With this combination, the image shows *Alma Mater* with the crisp black outlines and mostly white and yellow color palette of the Homer Simpson image. Figure 2 (b) combines the content of the *Alma Mater* statue picture with Jackson Pollock's style. Here, *Alma Mater* is rendered with the darker colors and disorderly (lack of) pattern for which Pollock is known. Figure 2 (c) combines the content of the Morningside campus picture with the Homer Simpson style. Again, we see the crisp outlines and bright white and yellow colors from this distinctive style. Figure 2 (d) combines the content of the Columbia Morningside campus picture with Jackson Pollock's style. With this style, the campus looks much darker and less orderly.

The next few sections explore the effects of key parameters on the generated images–namely, the number of training iterations, the initialization of the target image, and the relative weighting between content loss and style loss. For illustrative purposes, we examine these effects on two of the four combinations: *Alma Mater* with Homer Simpson and Columbia University's Morningside campus with Jackson Pollock.

Figure 2: Main output from our neural style transfer learning. Function settings: initialization from content image, seed $= 0$, steps $= 100$, $\alpha = 1$, $\beta = 1 \times 10^{-14}$. (a) Top left: *Alma Mater* and Homer Simpson. (b) Top right: *Alma Mater* and Pollock's *No. 5*. (c) Bottom left: Morningside campus and Homer Simpson. (d) Bottom right: Morningside campus and Pollock's *No. 5*

3.1 Effect of Training Iterations

In these examples, since the target image is initialized using the content image, each step in the training process serves to infuse more of the style into the picture. Figures 3 and 4 show the evolution of the learning process after different numbers of training iterations (10, 25, 50, 75, and 100 steps).

For the *Alma Mater* and Homer Simpson combination (Figure 3), there is a clear progression of the style seeping into the content. After 10 steps (Figure 3 (a)), the color palette has barely begun to transform. After 25 steps (Figure 3 (b)), the black outlines are forming and some of the white spaces are getting whiter. This process continues, adding the distinctive yellow coloring as well, as we go through more training steps. Between the 75-step and 100-step images (Figure 3 (d) and (e)), the returns are diminishing, with the latter image having only marginally brighter yellows and clearer white spaces.

For the Morningside campus and Jackson Pollock combination (Figure 4), there is also a progression as the steps go by, though the target image seems to stabilize sooner. To the naked eye, there is not much difference between the 50-step, 75-step, and 100-step images (Figure 4 (c), (d), and (e)).

Figure 3: Results from varying the number of training iterations. *Alma Mater* and Homer Simpson. Function settings: initialization from content image, seed $= 0$, $\alpha = 1$, $\beta = 1 \times 10^{-14}$. (a) Top left: steps $= 10$. (b) Top right: steps $= 25$. (c) Middle left: steps $= 50$. (d) Middle right: steps $= 75$. (e) Bottom: steps $= 100$.

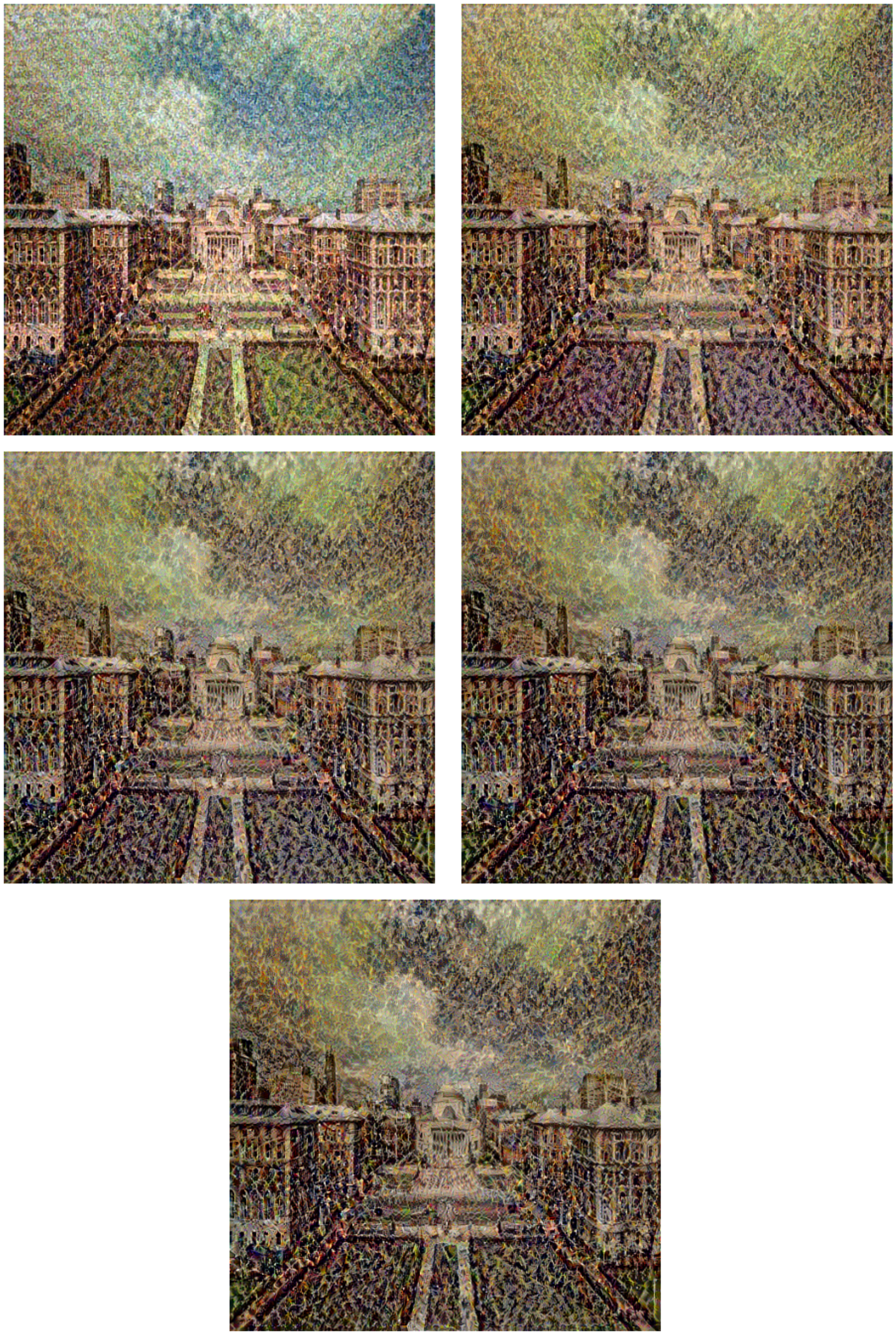

Figure 4: Results from varying the number of training iterations. Morningside campus and Pollock's *No. 5*. Function settings: initialization from content image, seed $= 0$, $\alpha = 1$, $\beta = 1 \times 10^{-14}$. (a) Top left: steps $= 10$. (b) Top right: steps $= 25$. (c) Middle left: steps $= 50$. (d) Middle right: steps $= 75$. (e) Bottom: steps $= 100$.

Figure 5: Results from varying the target image initialization. *Alma Mater* and Homer Simpson. Function settings: steps $= 100$, $\alpha = 1$, $\beta = 1 \times 10^{-14}$. (a) Left: initialization from content image, seed $= 0$. (b) Middle: initialization from random white noise image, seed $= 0$. (c) Right: initialization from random white noise image, seed $= 1$.

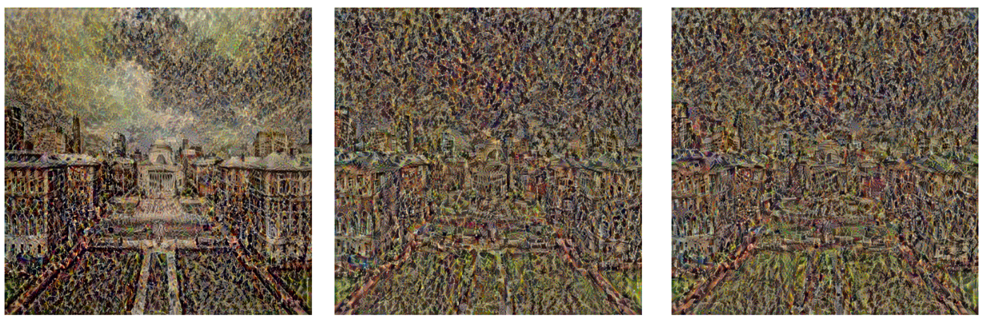

Figure 6: Results from varying the target image initialization. Morningside campus and Pollock's *No. 5*. Function settings: steps $= 100$, $\alpha = 1$, $\beta = 1 \times 10^{-14}$. (a) Left: initialization from content image, seed $= 0$. (b) Middle: initialization from random white noise image, seed $= 0$. (c) Right: initialization from random white noise image, seed $= 1$.

3.2 Effect of Target Image Initialization

As with any high-dimensional optimization problem, the initialization of the target has a notable impact on the final output. Figures 5 and 6 show the differences in the generated images using varying initializations of the target image. Specifically, for each of these content/style combinations, we initialize with the content image and with two different random white noise images.

For the *Alma Mater* and Homer Simpson combination, initializing with the content image (Figure 5 (a)) renders the clearest picture of the *Alma Mater* statue. While initializing with random white noise (Figure 5 (b) and (c)) still gives us a recognizable pattern, the outline is not nearly as crisp. Using different random seeds generates different-looking images, though unsurprisingly neither is observably "better" than the other.

For the Morningside campus and Jackson Pollock combination, initializing with the content image (Figure 6 (a)) also gives the clearest picture of the campus. The difference is perhaps more apparent here, as the random initializations (Figure 6 (b) and (c)) only have very vague building and walkway shapes. This is likely because of the style itself; Pollock's method of creating the artwork via paint drip is itself a somewhat random process, so the random initialization is more similar to the style image than to the content image. Therefore, the optimization process approaches a local minimum for the loss function that minimizes the style loss more than the content loss.

The takeaway from this experimentation is that initializing the target image with the content image renders an output that more closely adheres to the shapes in the content image. Initializing with random white noise, on the other hand, gives the output a more amorphous shape, which can vary with the use of different random seeds.

Figure 7: Results from varying the content vs. style loss weights. *Alma Mater* and Homer Simpson. Function settings: initialization from content image, seed $= 0$, steps $= 100$, $\alpha = 1$. (a) Left: $\beta = 1 \times 10^{-13}$. (b) Middle: $\beta = 1 \times 10^{-14}$. (c) Right: $\beta = 1 \times 10^{-15}$.

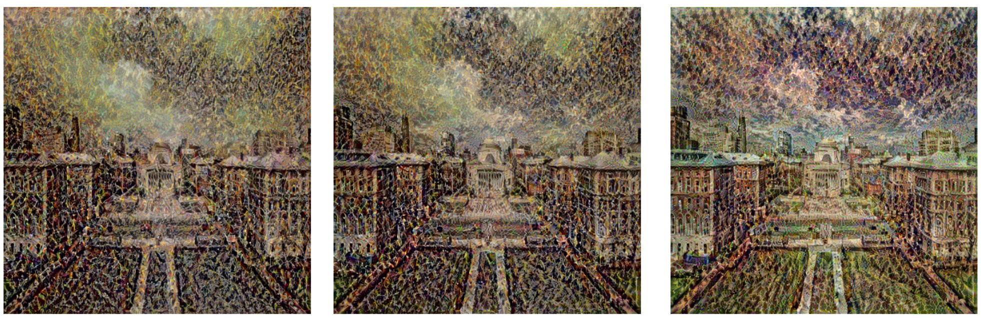

Figure 8: Results from varying the content vs. style loss weights. Morningside campus and Pollock's *No. 5*. Function settings: initialization from content image, seed $= 0$, steps $= 100$, $\alpha = 1$. (a) Left: $\beta = 1 \times 10^{-13}$. (b) Middle: $\beta = 1 \times 10^{-14}$. (c) Right: $\beta = 1 \times 10^{-15}$.

3.3 Effect of Content vs. Style Loss Tradeoff

The algorithm's loss function is, in actuality, a linear combination of two distinct loss functions: the content loss and the style loss. We must manage the tradeoff between content and style by adjusting the parameters that define the relative weights between content loss (α) and style loss (β). As mentioned previously, our content loss and style loss functions are scaled differently than those in Gatys, et al. (2016) [4], so the ratios of α to β may be different. While we experimented with many different combinations, for illustrative purposes we show $\alpha = 1$ with $\beta = 1 \times 10^{-13}$, 1×10^{-14}, and 1×10^{-15}.

For the *Alma Mater* and Homer Simpson combination (Figure 7), there are clear differences between these three parameter choices.

With $\beta = 1 \times 10^{-13}$ (Figure 7 (a)), many of the statue's features have been washed away, with only the basic outlines remaining. Since this is the example where the style loss receives the highest weight, it is unsurprising that this image looks the most "cartoony".

With $\beta = 1 \times 10^{-14}$ (Figure 7 (b)), more of the statue's finer features appear, and the yellow and white color palette is not as sharp. With $\beta = 1 \times 10^{-15}$ (Figure 7 (c)), there is very little of the Homer Simpson style. This weighting seems to tilt too heavily toward the content loss. For the Morningside campus and Jackson Pollock combination (Figure 8), the parameter choices still have an impact, but it is not quite as obvious as it was in the previous example. All three images are reminiscent of Pollock's chaotic splashes of paint, but the color palette is darker when the style loss has greater weight ($\beta = 1 \times 10^{-13}$, Figure 8 (a)) and more colorful (like the content image) when the style loss has less weight ($\beta = 1 \times 10^{-15}$, Figure 8 (c)).

Based on this analysis, it is clear that giving more relative weight to the style loss injects more of the style (and less of the content) into the final image, and vice versa. From our experimentation, we empirically observed that a β-to-α ratio on the order of 1×10^{-14} seems to strike the best balance, while 1×10^{-13} may be appropriate if the goal is to recreate the style more closely.

4 Discussion

4.1 Main Takeaways

In summary, we have utilized the VGG19 neural network to combine Columbia-themed content images–the *Alma Mater* statue and Morningside campus–with traditional and non-traditional style images–Jackson Pollock's *No. 5* and a portrait of Homer J. Simpson from *The Simpsons*, respectively–to create different renderings of the university in those artistic styles. Using the methods outlined in Gatys et al (2016) [4] and various TensorFlow tutorials [1] [3] [7], we developed our own tool that allow us to adjust many different hyperparameters create these renderings. We find that using the content image to initialize the target, running 100 training steps with $\alpha = 1$ and $\beta = 1 \times 10^{14}$, gives us output images that appropriately blend the content and style together.

This project is a visual example of the power and limitations that neural transfer learning can provide. Being able to successfully train and select the right hyperparameters for a convolutional neural network gives rise to a unique approach to art and design.

Graphic designers could potentially find a new rival or ally in neural networks. Social media giants such as Facebook and Snapchat can utilize this technology to expand their filter options. The world of digital art–or art in general–could find its next major movement.

4.2 Commentary on Other Style Images

Beyond the examples shown above, we explored many other artistic styles, some of which were more successful than others in blending with the content images. Based on this experimentation, we noticed an interesting pattern: The style images that worked best for these combinations tended to be those with distinctive color palettes (e.g. Homer Simpson's bright white and yellow or Jackson Pollock's dreary browns) and recognizable characteristics (e.g. Homer Simpson's sharp borders, or Jackson Pollock's random splatter). On the other hand, the style images that did not work as well tended to be those with a greater variety of shades and more diversity of shapes within the composition.

One example of a style image that did not work as well is Salvador Dali's *The Persistence of Memory* (Figure 9 (a)). This painting is known for its surreal depiction of melting clocks against a rocky background. However, these qualities do not translate well when run through the algorithm (Figure 9 (b)), perhaps because the shapes in the painting all have somewhat different characteristics (e.g. sharp edges in the landscape but flaccidity and roundness in the clocks).

Another example of a style image that did not work as well is Michelangelo Merisi da Caravaggio's *Supper at Emmaus* (Figure 9 (c)). Caravaggio is known for using shading and shadow to elicit an almost photorealistic appearance. However, this technique is difficult for the algorithm to learn (Figure 9 (d)), as it requires a very specific way of blending of many shades.

4.3 Further Research

Although our work builds on the the work of Gatys et al. (2016) [4] and various TensorFlow [1] [3] [7] and Kaggle tutorials [2] [10], we feel that there are many more changes worth considering in future work. One challenge we have seen–as discussed above–is how certain style images do not work well with our current implementation. There are many possible reasons for this: (1) we need to train for a greater number of steps that goes beyond the memory and computing power we currently have access to (2) we have yet to discover the right combination of current hyperparameters within the range of training steps we are currently capable of (3) the choice of layers of in the VGG19 neutral network should be different from our current choice, and (4) we need an entirely different neural network different from VGG19 altogether.

Additionally, we also find difficulty in quantifying exactly which rendered image is arguably better based on changing the hyperparameters alone. However, this may not even be necessary as the issue is more a matter of opinion and artistic preference. If this were to be used in application, the "best"

Figure 9: Other style images and results with other style images. (a) Top left: Salvador Dali's *The Persistence of Memory*. Source: Image Reference [4]. (b) Top right: *Alma Mater* and Dali's *The Persistence of Memory*. Function settings: initialization from content image, seed $= 0$, steps $= 100$, $\alpha = 1$, $\beta = 1 \times 10^{-14}$. (c) Bottom left: Michelangelo Merisi Caravaggio's *Supper at Emmaus*. Source: Image Reference [2]. (d) Bottom right: *Alma Mater* and Caravaggio's *Supper at Emmaus*. Function settings: initialization from content image, seed $= 0$, steps $= 100$, $\alpha = 1$, $\beta = 1 \times 10^{-14}$.

result would often be dictated by one's intentions, the context in which the output is used, and the audience that is viewing the output.

References

[1] Abadi, et al., "TensorFlow: Large-scale machine learning on heterogeneous distributed systems." *arXiv preprint arXiv:1603.04467* 2016.

[2] Basu, Victor. “Style Transfer Deep Learning Algorithm.” *Kaggle*, Kaggle, 26 Apr. 2019, www.kaggle.com/basu369victor/style-transfer-deep-learning-algorithm.

[3] “Fast Style Transfer for Arbitrary Styles.” *TensorFlow*, www.tensorflow.org/hub/tutorials/tf2_arbitrary_image_stylization.

[4] Gatys, et al., "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.

[5] *ImageNet*, Stanford Vision Lab, Stanford University, Princeton University, 2016, image-net.org/index.

[6] “Large Scale Visual Recognition Challenge 2014 (ILSVRC2014).” *ImageNet Large Scale Visual Recognition Competition 2014 (ILSVRC2014)*, Stanford Vision Lab, Facebook, Google, Stanford University, University of North Carolina at Chapel Hill, 2014, image-net.org/challenges/LSVRC/2014/index.

[7] “Neural Style Transfer.” *TensorFlow*, www.tensorflow.org/tutorials/generative/style_transfer.

[8] Olah, et al., "Feature Visualization", *Distill*, 2017.

[9] Simonyan, et al., "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

[10] Surma, Greg. “Style Transfer.” *Kaggle*, Kaggle, 13 Jan. 2019, www.kaggle.com/greg115/style-transfer.

Image References

[1] French, Daniel Chester, and Beyond My Ken. “2014 Columbia University *Alma Mater* Closeup.” *Wikimedia Commons*, Wikimedia Commons, 1 Jan. 2015, commons.wikimedia.org/wiki/File:2014_Columbia_University_Alma_Mater_closeup.jpg.

[2] Caravaggio, Michelangelo Merisi da, and National Gallery, London. “1602-3 Caravaggio, *Supper at Emmaus* National Gallery, London.” *Wikimedia Commons*, Wikimedia Commons, 28 Mar. 2018, commons.wikimedia.org/wiki/File:1602-3_Caravaggio,Supper_at_Emmaus_National_Gallery,_London.jpg.

[3] “Columbia University Morningside Campus.” *Columbia University*, Columbia University, images.app.goo.gl/i5ZkryXru6Sjav1t6.

[4] Dali, Salvador, and Xennex. “La Persistencia De La Memoria.” *WikiArt*, WikiArt, 13 Aug. 2020, www.wikiart.org/en/salvador-dali/the-persistence-of-memory-1931.

[5] “Homer Simpson Portrait.” *NY Daily News*, NY Daily News, interactive.nydailynews.com/2016/05/simpsons-quiz/.

[6] Pollock, Jackson. “Number 5, 1948.” *Jackson Pollock Biography, Paintings, and Quotes*, Www.Jackson-Pollock.org, www.jackson-pollock.org/number-5.jsp.