A Neural Algorithm of Artistic Style

E4040.2018Fall.SHYW.report
Yiwen Zhang yz3310, Wenting Su ws2512, Haotian Zeng hz2494
Columbia University

Abstract

We first followed the original paper to reconstruct images by transferring a style from an artwork to the content of a photography. After that, we extend the original algorithm with multiple style blending and color preservation to achieve better visual effects. Turning the hyperparameters can be tricky during training process. At the end, we got pretty good image reconstruction results.

1. Introduction

It has never been easier to have your own picture painted in the style of your favorite painter. Neural networks, once again, are at the heart of this capability. The first major step in this field was introduced in the paper: *A neural Algorithm of Artistic Style,* which shows that the task of transferring the style from one image to the content of another can be posed as an optimization problem that can be solved through training a convolutional neural network. Specifically, we're taking two images as inputs: content image determines how the generated image will look like while style image gives the style to the generated image. At the end, we will have the output image with content of the content image and the style of the style image.

In order to achieve such result, in this case, we are using a pretrained image classification network VGG19. The multiple-layer neural network learns to recognize more abstract parts of the content as it goes through the layers: it throws away much information about the raw pixel values and keep only semantic concepts. We care about these last layers because they contain the content we want to make artistic. The ability to merge style and content not only enables us to make "fake" artworks, but also allows us to study how art and style are created by artists.

In addition to reproduce the images provided by original paper, we also compare how different learning rate and different optimizer would impact the visual result in our project. What's more, we tried to preserve the color of the content and add more styles but not only a single style.

2. Summary of the Original Paper

2.1 Methodology of the Original Paper

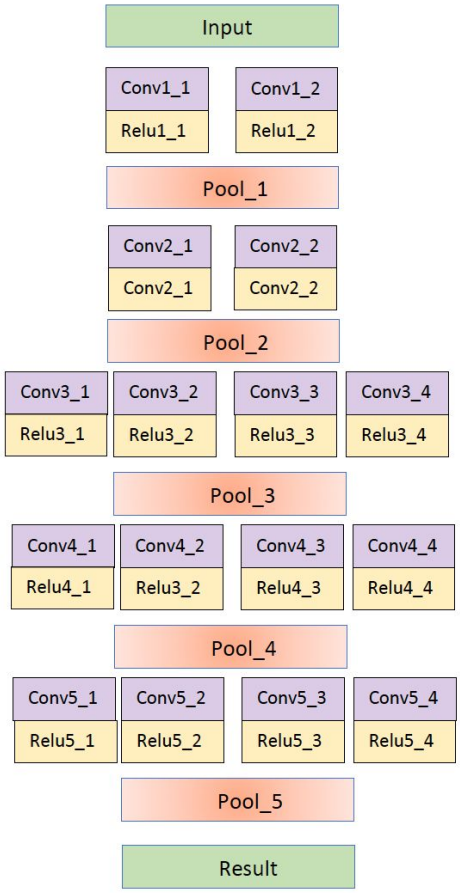

Figure 2.1. VGG19 network architecture

The author provides a method for combining the style and content from different input images, which is on the basis of a VGG-Network, a Convolutional Neural Network that rivals human performance on a common visual object recognition benchmark task.

In the paper, 16 convolutional and 5 pooling layers of the 19 layer VGG-Network, as shown in Figure 2.1, are used to extract the content and the style of different image separately. For style, the model extracts correlations from

VGG among features at multiple layers. For content, it matches the representation from a particular layer. It's worth mention that the author use Average Pooling instead of Max Pooling because information is so important and we don't want to throw away a large number of pixel values of the previous layer and keep only the highest values.

The key idea of transferring the style is to construct an minimization problem consisting of a content loss and a style loss, which shows us how far away our generated image is from the ideal style transfer image. We then iteratively improve the image using gradient descent via backpropagation.

In detail, the authors first starts with a random generated image, which known as white noise. Then they pass the content image and style image through the VGG network to calculate the total style and content loss. After that, sending this loss back through the network and using backpropagation to determine the gradient of the loss function with respect to the input image. Applying gradient descent, a small update is made to the input image in the negative direction of the gradient which will cause the loss function to decrease in value. They repeat this process iteratively and get the final output.

2.2 Key Results of the Original Paper

For one thing, the paper proposes a groundbreaking idea that the task of transferring the style from one image to the content of another can be posed as an optimization problem, which can be solved through training a neural network. However, there is no promising that the generated image must be that pretty. The final performance of the output image actually relies a lot on the choice of the style image and content image. Not all style-content combinations can present a good visual performance and show harmony in the output. We need select the good combinations that wouldn't have too much conflict of the color or texture and then can get the harmonic outputs.

For another, it offers an algorithmic understanding of how humans create and perceive artistic imagery. The authors claim that the representations of content and style in the Convolutional Neural Network are separable, that is to say, neural representations can independently capture the content of an image and the style in which it is presented. Importantly, the mathematical form of the style representations, which simply compute the correlations between different types of neurons in the network generates a clear, testable hypothesis about the representation of image appearance down to the single neuron level. Such results suggest that performing a complex-cell like computation at different processing stages along the ventral stream would be a possible way to obtain a content-independent representation of the appearance of a visual input.

3. Methodology

In this section, we start with showing our objectives, follow by stating the technical challenges we met during achieving the goals. After that, we discuss how we formulate the task in the original paper step by step and where we can make some improvements.

3.1. Objectives and Technical Challenges

Our objectives are:
1) Reproduce the result in the original paper, which combines the style of artwork and the content of a photography to make artistic images.
2) Try different settings of parameters and explore how they can influence the outputs.
3) Try different optimizers and test how they could change the outputs.
4) Try to present some extensions to the neural artistic style transfer algorithm in the original paper.

And some of the technical challenges are:
1) The ratio of content loss coefficient alpha and style loss coefficient beta can be different time to time, we could not find a "procedure" to determine the ratio directly. For each input, we need a lot time to seek a proper balance between the content and the style.

2) We need to trade off the cost of time and the output. For example, more iterations always give us better outputs but taking longer time. Also, the original optimizer L-BFGS given in the paper can generally give better result but is much slower than some other optimizers, such as Adam.

3) There are also some unexpected problems during the implementation. For example, setting both alpha and beta

as power of 10 (e.g. alpha = 1, beta = 1000) cannot yield the expected result given by certain ratio. To solve the problem, we need to adjust at least one of the parameters (e.g. alpha = 5, beta = 10^n or alpha = 1, beta = $5 * 10^n$).

3.2. Problem Formulation and Design

In the project, we use VGG19 network architecture to match the corresponding style and content target representations as described in the original paper. After that, we extend the original algorithm with multiple style blending and color preservation to achieve better visual effects.

1) We first followed the original paper and reproduce the outputs. After taking two inputs, a content image and the style image that we want to match, we transform the input image by minimizing the content and style losses with backpropagation. As for content representation, we hope deeper convolutional layers are able to represent high-level content features of the original but lose the exact pixel information. So that content loss of layer *l*, as shown below, can be described as sum over the squared-error loss between the feature representation *P* of the original image *p* and the feature representation *F* of the generated image *x* of all filter *i* at position *j*.

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F_{ij}^l - P_{ij}^l\right)^2$$

Unlike content representation, we need the spatial correlation of the values within a feature map to represent styles. These feature correlations in layer *l* are given by the Gram matrix *G*, which is the inner product between the vectorised feature map *i* and *j* in the layer.

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l.$$

That is to say, if we had two images whose feature maps at a given layer produced the same Gram matrix we would expect both images to have the same style. For each layer with *N* distinct filters each of size *M*, we can develop a style loss function that measures mean-squared distance between the style representation (gram matrix) of the original style image *A* and the style representation of the output image in that layer *G*, which can be expressed by:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2$$

The total loss between original style image *a* and generated image *x* can be reached by summing over the loss of all layers proportional to weighting factors (contribution) of each layer:

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

Combining all things together, the total loss can then be written as a weighted sum of the both the style and content losses, where the weights can be adjusted to preserve more of the style or more of the content.

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

By executing the training process, we could achieve similar output as the original paper.

2) Different learning rates is examined to explore how they could affect our outputs. At the end we need to determine the optimal parameters that make the best "artwork".

3) We also tried different optimizers such as Adam, Adagrad, Rmsprop other than L-BFGS, the optimizer given in paper. During the process, we want to understand why L-BFGS is chosen and is there any better optimizers in the art style transfer application.

4) Although the original paper can give good enough results by transferring styles to an input, it also has some potential shortcomings. For example, the algorithm only allows one style to be transferred at a time. Also, transferring the colors of the original painting may alter the appearance of the scene in undesirable ways. We presents some extensions to the original artistic style transfer algorithm to address those problems.

4. Implementation

This section describes how we implement our algorithms to achieve the objectives. First, we give an overview of the whole algorithm structure of our project. Then we discuss our network design in detail and our software design step by step.

4.1. Deep Learning Network

Followed the original paper, we used the feature space provided by the 16 convolutional and 5 pooling layers of the 19 layer VGG Network. The whole architecture

consists of 5 sequential groups of layers, each composes of a series of convolution layers and pooling layers (as shown in Figure 2.1). For each convolution layer, we use 1-by-1 kernels and pad outputs spatially to remain the same size. For the pooling layer, as discussed above, we use average pooling with the same 2-by-2 filter size. As for our output images, we matched the content representation on layer 'relu4_2' and the style representations on layers 'relu1_1', 'relu2_1', 'relu3_1', 'relu4_1' and 'relu5_1' (weight = 1/5 in those layers, weight = 0 in all other layers) . The ratio alpha/beta was either $5 * 10^{-3}$ (Fig 5.2) or $5 * 10^{-4}$ (Fig 5.6).

Besides the structure used by the original paper, we extend our algorithm to satisfy the needs of preserving the color of the content and add more styles on a single content image. Also, our algorithm allows the selection of different optimizers to reach the wanted outputs (as shown in Figure 4.1).

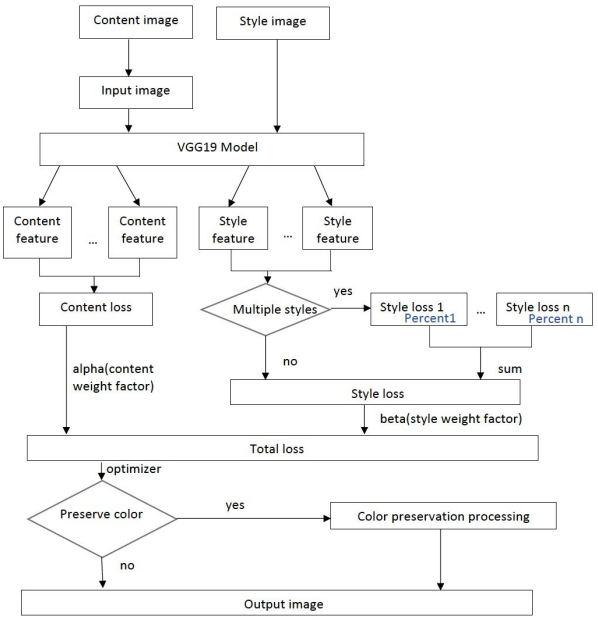

Figure 4.1. Flowchart of the network in project

The dataset we used are listed below.
Artwork for style usage:
1) *The Starry Night* by Vincent van Gogh, 1889.
2) *Guernica* by Pablo Picasso, 1937.
3) *Femme nue assise* by Pablo Picasso, 1910.
4) *Sunrise* by monet, 1872.
5) *Nymphéas* by monet, 1915.

Photography for content usage:
1) *Fortress of Rumeli*, Istanbul, Turkey.
2) The view of *Bosphorus* from Asia side, Istanbul, Turkey.
3) *İstiklal Avenue*, Istanbul, Turkey.
4) View of *La Tour Eiffel* from *Arc de Triomphe*, Paris, France.
5) *Manhattan bridge* and *Manhattan*, New York City, United States.
6) Panda in *Research Base of Giant Panda Breeding*, Chengdu, China.

4.2. Software Design

Our software design can be roughly divided into three parts, including:
1) VGG model construction
2) Auxiliary functions, like load image and save image
3) Training function, providing three style transfer modes and six different optimizers

To use our package to implement a style transfer task, there are just two simple steps to follow:
a) Instantiate an artist object from the class Artistic Style
b) Call the training function and adjust parameters as needed, like style transfer mode, learning rate, alpha/beta ratio, optimizer, etc

Our entire process design follows the user-friendness rule, users just need to provide the names of the content, style, output and choose the parameter and then our predefined function would perform other tasks automatically and return the expected result.

5. Results

5.1. Project Results

The key finding of original paper is that the representation of content and style is seperate, thus we can manipulate the two representation separately and produce new images. To verify this finding, we generated some images combining different artistic works and photography pictures. In these combination, we remained the same optimizer as the paper, the L-BFGS-B optimizer, adopted a $5 * 10^{-3}$ alpha/beta ratio, and trained for 500 iterations.

As shown in figure 5.1, the results are delightful. We reproduced the approach described in the paper and could generate new images using arbitrary pictures and artworks. From figure 5.1, we can see that the pictures all inherited the style from the artistic pieces successfully and presented a new harmonious effect.

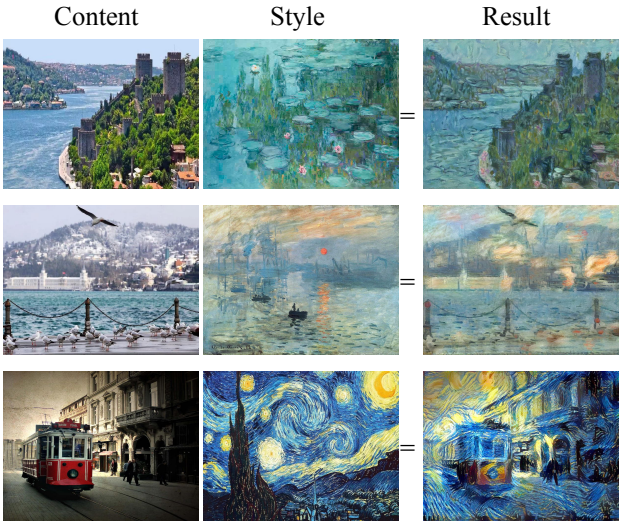

Figure 5.1 Results using original method in the paper

5.2. Comparison of Results

5.2.1 Different transfer methods

Besides the original style transfer method described in the paper, we implemented another two extensions, multiple style blending and color preservation. For all three methods, we adopted "Adam" optimizer and an alpha/beta ratio at a 10^{-3} magnitude. The results were gained after training for 1000 iterations and shown in Figure 5.2.

In Figure 5.2, the upper left image is our first content image, while the upper right one is the output of original style transfer with the style image Guernica. We obtained the bottom left image through multiple style blending approach using two style images, Guernica and Starry night, contributing 0.6 and 0.4 respectively. The bottom right one is the production of the original method after adding the color preservation effect.

As we can see, although the vanilla version could transfer the image successfully, it would affect too much on the color of the content image. Often if the style image has a dark or dim color composition, the vivid content image may suffer from that. The multiple color blending could alleviate the problem by adding another more colorful style, however, it is often hard to choose an appropriate style combination that wouldn't make the image too strange and have a harmonious blending effect. Therefore, to achieve better visual performance, a way that could not only apply the style to the image but also maintain the image's initial color is very necessary. And that's what color preservation does. As shown in the figure, the result looks pretty good.

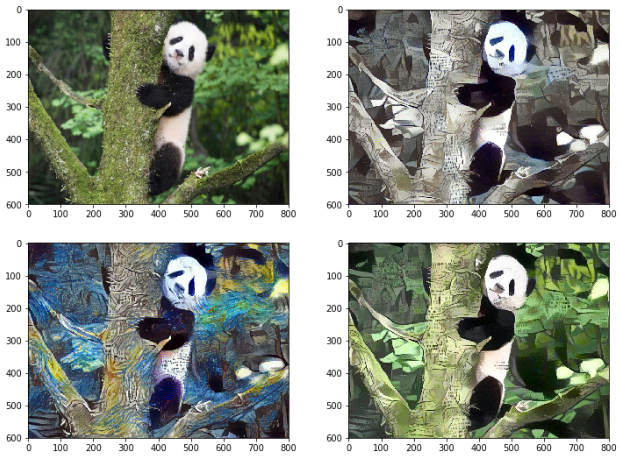

Figure 5.2 Comparison between different style transfer

5.2.2 Different optimizers

In the original paper, L-BFGS-B optimizer was suggested. To explore if there would be a better optimizer, we compared five commonly used optimizers and recorded their results. This time, we chose a picture of New York as our content and the Sunrise as our style, the alpha/beta ratio as $5 * 10^{-3}$. The learning rate for all optimizers is the same, 0.2. For each optimizer, we trained for 1000 iterations.

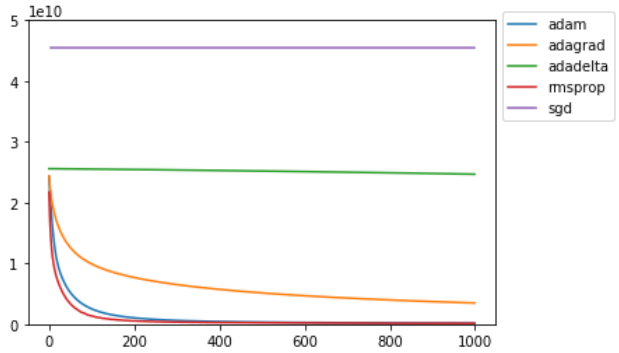

Figure 5.3 Loss curve under different optimizers

As shown in Figure 5.3, Adam and RMSProp performed best among the five optimizers. Gradient Descent optimizer is the one with the worst performance, which barely learned from the style.

To have a more direct sense of the performances, we presented the output images in Figure 5.4.

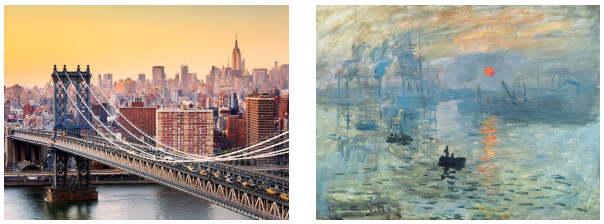

Figure 5.4 a) Content(left) and Style(right) adopted

Figure 5.4 b) Output under different optimizers

In figure 5.4 b), from left to right and then from up to down, the optimizer is Adam, Adagrad, Adadelta, RMSProp, Gradient Descent and L-BFGS-B respectively. Clearly, L-BFGS-B is the one with the best performance among all optimizers, which successfully finished the reconstruction of both content and style. While Gradient Descent is the worst, which completely failed at the reconstruction. Although Adam and RMSProp have better performance than others, but when compared with L-BFGS-B, they seemed not so satisfying.

Optimizer	Adam	Adagrad	Adadelta	RMS -Prop	SGD	L-BFGS-B
Running time (min)	13	13	13	13	12	33

Table 5.1 Running time comparison between optimizers

We ran all our examples on an instance at the Google Cloud Platform, with one NVIDIA Tesla K80 GPU. From table 5.1, we can see that L-BFGS-B would take the longest time to finish training, 33 mins for 1000 iterations, while the rest optimizers just need around 13 mins. So when there is a strict time limit, L-BFGS-B might not be the best choice.

5.2.3 Different learning rates

Although the alpha/beta ratio is an important factor that affects the content and style reconstruction in the output image. There are other parameters that could also impact a lot on the content-style tradeoff. The learning rate is one of them. We ran all our examples on an instance at the Google Cloud Platform, with one NVIDIA Tesla K80 GPU. From table 5.1, we can see that L-BFGS-B would take the longest time to finish training, 33 mins for 1000 iterations, while the rest optimizers just need around 13 mins. So when there is a strict time limit, L-BFGS-B might not be the best choice.

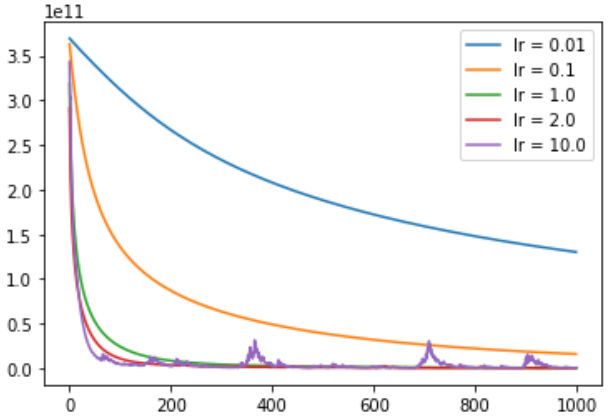

Figure 5.5 Loss curve under different learning rate setting

To compare how different learning rate would influence the content and style composition in the final results, we

chose a photography of Paris as content and Femme nue assise as style, the content weighting factor alpha = 5, the style weighting factor beta = 10000. For each learning rate, we ran for 1000 iterations and recorded the loss changes during the process.

From figure 5.5, we can see that a learning rate of 0.01 is too small to get enough learning during 1000 iterations. Setting the learning rate as 0.1 is better but there's still space to reduce the loss to a smaller one. A learning rate 1.0 or 2.0 is a pretty good choice under our situation. Having a learning rate as large as 10.0 would be too much and cause the big fluctuation in the curve.

After checking how learning rate impacts the loss curve, we need to see how the difference would reflect on the output images.

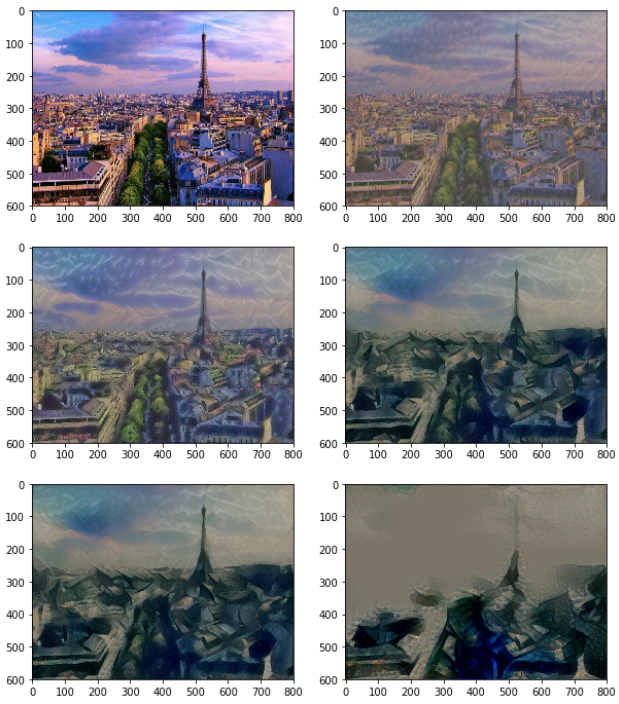

Figure 5.6 Output under different learning rate setting

In the figure 5.6, the image on the upper left is the initial content picture. For the rest, from left to right and then from up to down, the learning rate increases from 0.01 to 10.0. As we can see, the larger the learning rate is, the more the style would influence.

5.3. Discussion of Insights Gained

VGG 19 network would take a lot of time on parameter tuning and the algorithm itself is not so efficient, which could take us 15 minutes or so to get image reconstruction results. For our project, this network would be fine, but may not be suitable for style transfer task that has real time transfer needs.

We also found that different choice of style and content would perform quite differently at various optimizers, learning rates, and weighting parameters. Therefore, there is no universal rule that suits for all combination of style and content.

When we tried different optimizers, we found for most artistic work, Adam could perform well. However, for a few artistic works that are hard to learn, for instance, the Composition VII by Wassily Kandinsky, using L-BFGS-B would produce the best result.

6. Conclusion

Based on the paper *A Neural Algorithm of Artistic Style,* we applied a deep CNN that further improves the style transfer quality by incorporating depth information in the training process of image transformation network. Our result not only shows that this model is implementable, but also demonstrated some high quality style transfer results which produced promising improvements on the original method. This success may inspire future research in improving quality of styles transfers by introducing even more parameters into styles transfers like segmentation information. We also tried to incorporate two sets of styles into one. If we had more time, we would try all of those implementations upon our depth perception model, such as multiple content, partial content preservation and so on. On the more personal level, in this project, not only did we gain a much deeper understanding of computer vision, convolutional neural network, we also became significantly more familiar with the tools to implement neural networks, like python numpy, and Tensorflow.

7. Acknowledgement

We sincerely appreciate everything we learned from both inside and outside the class. It is all the lectures and the previous assignments that paved the way for this project. And we want to express our gratitude to the TA team who helped us in the process, they are so selfless and patient for answering all our questions and doubts when we got stuck, and guided us to obtain clear direction of our project. Without any of this, we couldn't finish such an interesting and meaningful project.

8. References

[1]. Link to Bitbucket: https://bitbucket.org/ecbm4040/2018_assignment2_ws2512/src/master/SHYW.project.yz3310.ws2512.hz2494/
[2]. Ashikhmin, N. Fast texture transfer. IEEE Computer Graphics and Applications 23, 38–43 (2003).
[3]. Gatys LA, Ecker AS, Bethge M. A Neural Algorithm of Artistic Style [Internet]. arXiv [cs.CV]. 2015.
[4]. Gatys LA, Ecker AS, Bethge M. Image Style Transfer Using Convolutional Neural Networks. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016. pp. 2414–2423.
[5]. Gatys LA, Ecker AS, Bethge M, Hertzmann A, Shechtman E. Controlling Perceptual Factors in Neural Style Transfer. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017. pp. 3730–3738.
[6]. Johnson J, Alahi A, Fei-Fei L. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. Computer Vision – ECCV 2016. Springer, Cham; 2016. pp. 694–711.
[7]. Gatys LA, Ecker AS, Bethge M, Hertzmann A, Shechtman E. Controlling Perceptual Factors in Neural Style Transfer. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017. pp. 3730–3738.
[8]. Zhu J Y, Park T, Isola P, et al. Unpaired image-to-image translation using cycle-consistent adversarial networks[J]. arXiv preprint, 2017.

9. Appendix

9.1 Individual student contributions in fractions - table

	yz3310	ws2512	hz2494
Last Name	Zhang	Su	Zeng
Contribution percentage	1/3	1/3	1/3
What I did 1	Developed the multiple style blending extension	Developed the original algorithm and wrapped up code	Developed the color preservation extension
What I did 2	Wrapped up report writing	Tried different optimizers and tuned learning rate	Finalized the report