NEURAL ARTIST: APPLICATION OF NEURAL STYLE TRANSFER ON A DIVERSE SET OF INPUTS

Nora Agah[1], Angela Chan[1], and Hannah Lee[1]

[1]Chandra Family Department of Electrical and Computer Engineering, The University of Texas at Austin

ABSTRACT

Neural style transfer effectively utilizes different techniques to transfer the style of one image to another image while retaining the content of the latter. We adapt a neural style transfer algorithm utilizing a deep neural network to work with pictures, images, doodles, gifs, videos, and portraits. Results varied depending on the number of training iterations, the similarity between the two input images, and varying parameter changes. We recognize several key limitations to our current neural style transfer algorithm and potential solutions to address them. Ultimately, we learned that style itself is an open-ended term, as it can encompass texture, color, structure, and more.

Index Terms— Neural Style Transfer, Convolutional Neural Networks, VGG19

1. INTRODUCTION

Each art movement and artist adopts distinctive and defining styles. The artist masterfully balances this particular style along with the content that they desire to convey. However, transferring the style of an artist or art movement to another image can be extremely difficult and time-consuming to accomplish by hand. The seminal work of Gatys, et al. proposed an algorithm that utilized a convolutional neural network to achieve this task, demonstrating the potential of deep-learning-based computational methods for this purpose [1]. This work would pioneer a new field known as neural style transfer (NST) [2].

In this paper, we seek to apply neural style transfer to a variety of inputs, including pictures and images, doodles, gifs, videos, and portraits. We first describe a general NST pipeline that we then adapt and modify to each of these inputs. We then discuss these specific changes and the results of the pipeline for each input as well. Finally, we conclude with an overall discussion of our takeaways from what we learned from the project.

2. GENERAL METHODS

We utilized MATLAB's Image Processing and Deep Learning Toolboxes in order to create our general NST pipeline [3]-[4]. Specifically, MATLAB provides a base framework for applying NST using deep learning, which we used as the initial pipeline that we later modified [5].

The initial pipeline provided by MATLAB is as follows. The pipeline takes in two input images: (1) the content image, defined to be the image that we desire to alter the style of and (2) the style image, defined to be the image that contains the style that we want to apply on to the content image. The pipeline uses a modified version of a pretrained VGG-19 deep neural network in order to extract the key features of the content and the style image [5]. The fully connected layers of the network are removed to achieve this, and the max pooling layers are replaced with average pooling layers in order to decrease any fading effect [5].

Before feeding the input images into the VGG-19 network, the images are preprocessed. The images are modified to be the same size, which we set to usually be smaller or equal to the content image size for quicker processing [5]. Then, since the VGG-19 network was trained on images subtracted by the channel-wise mean, we subtracted the channel-wise mean from the images, which is equal to the mean of the first layer of the VGG-19 network [5].

Following preprocessing, we initialized the transfer image. For this step, we initialized it using a weighted combination of white noise and the content image [5]. Using the content image for the transfer image initialization ensures that the transfer image produced an image similar to the content image, but it biases the result heavily toward the content image [5]. To reduce this bias, we initialized it along with some white noise [5].

We define our loss function as a weighted combination of both the content loss and the style loss, where α is the content loss weight (set to 1 in our code) and

β is the style loss weight (set to 1000 in our code) [5]:

$$L_{total} = \alpha \times L_{content} + \beta \times L_{style}$$

Content loss is defined as the mean squared difference between the features from the content image and that from the transfer image for each of the content feature layers, where Y-hat is the predicted transfer image feature map, Y is the predicted content image feature map, W^l_c is the weight of the content layer at the lth layer, H is the height of the feature map, W is the width of the feature map, and C is the number of channels in the feature map [5]:

$$L_{content} = \sum_l W^l_c \times \frac{1}{HWC} \sum_{i,j} (\hat{Y}^l_{i,j} - Y^l_{i,j})^2$$

Style loss is defined as the mean squared difference between the style image Gram matrix and the transfer image Gram matrix, where the Gram matrix represents the style of the image, Z-hat is the predicted transfer image feature map, Z is the predicted style image feature map, $G_{Z\text{-hat}}$ is the transfer feature Gram matrix, G_Z is the style feature Gram matrix, and W^l_s is the weight of the style layer at the lth layer [5]:

$$G_{\hat{Z}} = \sum_{i,j} \hat{Z}_{i,j} \times \hat{Z}_{j,i}$$

$$G_Z = \sum_{i,j} Z_{i,j} \times Z_{j,i}$$

$$L_{style} = \sum_l W^l_s \times \frac{1}{(2HWC)^2} \sum (G^l_{\hat{Z}} - G^l_Z)^2$$

The layer in which the content features are extracted is the fourth convolutional layer of the VGG-19 network, with the weights equal to 1 [5]. The layers in which the style features are extracted are the first through fifth convolutional layer, with the weights equal to 0.5, 1.0, 1.5, 3.0, and 4.0 respectively [5].

After extracting the content and style features, we then trained the model [5]. For a set number of iterations, we calculated the content loss and style loss using the above equations [5]. Then, we update the transfer image using adaptive moment estimation (Adam) [5]. From all the iterations, we select the best style transfer image [5].

Lastly, we postprocessed the transfer image by adding the channel-wise mean back to the transfer image and converting it into an 8-bit integer [5]. The transfer image is resized [5]. However, we noticed that the color was muted and did not transfer well. Style encompasses both texture and color, so to compensate for this, we matched the color histogram of the transfer image to the color histogram of the style image.

2.1 Division of Labor

We divided the labor as follows. First, all three of us got the general pipeline running on our individual devices. Upon doing this, we were tasked with modifying the pipeline to best tailor it to the specific input. Angela was responsible for the doodles, Hannah was responsible for the gifs and videos, and Nora was responsible for the portraits.

3. PICTURES AND IMAGES

This section describes the application of the general NST pipeline to pictures and images. No adaptations were made, as we sought to consider the most generic two-dimensional input. In the following section, we analyze the impact of changing specific parameters, input image similarity, and color histogram matching.

3.1 Effect of Number of Training Iterations

We varied the number of training iterations and visually inspected the quality of the style transfer. As expected, increasing the number of training iterations improved the quality of style transfer until a certain point where it plateaus. Figure 1 illustrates this result. However, there is an expected tradeoff in time. We kept all the other parameters constant, with a learning rate of 2 and noise to content ratio of 7:3.

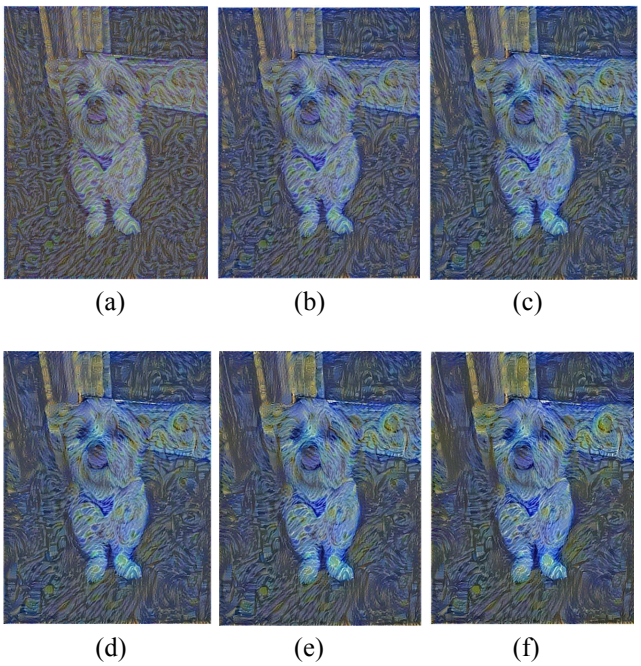
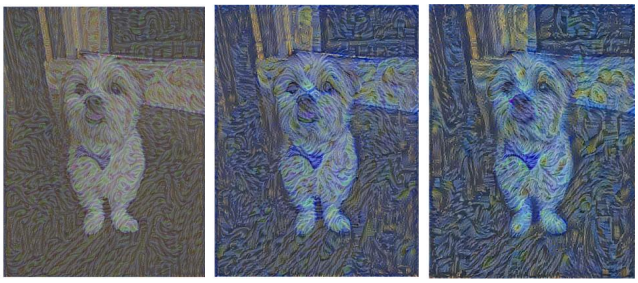


Figure 1. The Transfer Images of the Starry Night Art Style onto the Image of the Dog with a Bowtie with Different Training Iterations: (a) 50, (b) 100, (c) 200, (d)

300, (e) 500, and (f) 1000.

3.2 Effect of Learning Rate

We then tested the impact of changing the learning rate on the quality of the style transfer, while keeping the number of training iterations and the noise to content ratio constant at 100 and 7:3 respectively. We discovered that smaller learning rates lead to slower style transfers, while larger learning rates lead to faster style transfers. This is illustrated in Figure 2.



(a) (b) (c)

Figure 2. The Transfer Images of the Starry Night Art Style onto the Image of the Dog with a Bowtie with Different Learning Rates: (a) 0.5, (b) 2, and (c) 5.

3.3 Effect of the Noise to Content Ratio During Transfer Image Initialization

As described in Section 2, we initialize the transfer image with a weighted combination of white noise and the content image. We decided to vary the weights and analyze the impact that this has on the quality of the style transfer. For this, we kept the number of training iterations constant at 100 and the learning rate at 2. We varied the noise to content ratio from 0:10 to 10:0, and the results are illustrated in Figure 3. When the ratio is closer to 0:10, the initial transfer image is close to the content image, so the output image is also more similar to the content image in color and structure. On the other hand, when the ratio is closer to 10:0, the initial transfer image is more similar to a gray, white-noise image, so the resulting transfer image has lost most of the content definition.

(a) (b) (c)

Figure 3. The Transfer Images of the Starry Night Art Style onto the Image of the Dog with a Bowtie with Different Noise to Content Ratios: (a) 3:7, (b) 7:3, and (c) 9:1.

3.4 Effect of Input Image Content Similarity

Like style, image similarity is also an open-ended term, as it can potentially encompass structure, style, color, content, and more. In this specific case, we analyze the image of content similarity. This was selected because it is expected that the style will be transferred more seamlessly between images with similar contents. That is, the style transfer of an Impressionist-style painting of a lily pond to a landscape picture of nature will likely look more natural than a style transfer of a Jackson Pollock painting, even if the lily pond image and the nature picture have a completely different structure.

We tested this idea and visually inspected the result on various input content and style images. For these, the number of training iterations, the learning rate, and the noise to content ratio were 100, 2, and 7:3 respectively. The results aligned with our expectations, as shown in Figure 4. When the input and style images drastically vary, the quality of the style transfer is not as good.

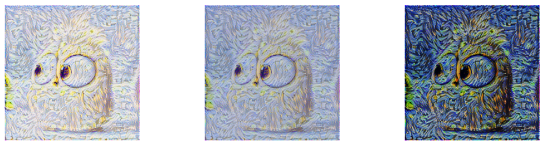
(a) (b)

Figure 4. The Transfer Images of a Claude Monet Style and Jackson Pollock Style onto a Picture of a Flower.

3.5 Effect of Color Histogram Matching

While we ultimately decided to match the color histogram of the transfer image to the style image since color is inherently a component of style, we analyzed the effect of color histogram matching to various weighted combinations of the content image and style image. We kept the number of iterations constant at 100, the noise to content ratio at 7:3, and the learning rate at 2. Figure 5 includes the results of this analysis.

We observed that when the color histogram was only matched to the content image, the only component of style transferred is textural in nature. When a combination of the content and style image is utilized instead, the color becomes more dull, especially if the colors of the content and style images drastically differ.



(a) (b) (c)

Figure 5. The Transfer Images of the Starry Night Art Style onto a Picture of a Bird [10]] with Color Histogram Matching to (a) the Original Content Image (b) an Average of the Content and Style Images and (c) the Style Image.

4. DOODLES

This section discusses the adaptation of the general NST pipeline to work for simple doodles. In comparison with images, doodles do not have much detail for each object in the picture. Because of the lack of details, the NST does a general overlay of the style onto the entire doodle, and the edges between objects in the resulting style transferred doodle are smeared together. In the following sections, we discuss the use of masking to solve the merging issue and issue of a general overlay [6]-[7].

4.1 Specific Methods

The preprocessing, initialization, and training method for style transfer of doodles still remain the same as the original NST pipeline.

The method we used to improve the original NST lies in updating the definition of the loss function as discussed in the paper by Gatys, who was inspired by Champandard [6]-[7]. Champandard introduced a new concept, semantic style transfer, which involves using a mask for the style image and the content image to assist the style transfer. The loss function of the original neural style transfer only considers the difference between the feature maps of the content/style images and the transfer image at a global level; hence, the transfer image looks like the content image with an overlay of the style image. In order to style different objects in the content image with the desired patterns from the style image, the two masks need to match in the number of clusters and the colors. Let each mask have R colors clusters. The main idea is for the content and the style masks to match in the regions where the content image and style image are similar as shown in Figure 6.

Figure 6. The content image and mask are on the top layer, and the style image and mask are on the bottom layer.

With the masks, we define a new loss function where the style loss function will use a different gram matrix, G^r_l, where l still stands for the convolutional layer l and the r stands for the region r of the mask image. The guided Gram matrix, G^r_l, is defined in terms of the updated feature map, F^r_l, which is the original feature map multiplied by the mask, T^r_l [6]:

$$\mathbf{G}^r_\ell(\mathbf{x}) = \mathbf{F}^r_\ell(\mathbf{x})^T \mathbf{F}^r_\ell(\mathbf{x})$$

$$\mathbf{F}^r_\ell(\mathbf{x})_{[:,i]} = \mathbf{T}^r_\ell \circ \mathbf{F}_\ell(\mathbf{x})_{[:,i]}$$

Then, the style loss function is defined as the following [6]:

$$E_\ell = \frac{1}{4N_\ell^2} \sum_{r=1}^{R} \sum_{ij} \lambda_r \left(\mathbf{G}_\ell^r(\hat{\mathbf{x}}) - \mathbf{G}_\ell^r(\mathbf{x}_S) \right)_{ij}^2$$

To incorporate the regional information, we need to extract the R individual binary masks from the original masks, where 1 represents the corresponding pixel is in the region and 0 represents it is not. Then, we iterate through each region, and for each region, we iterate through the 5 convolutional layers (i.e. conv1_1, conv2_1, conv3_1, conv4_1, conv5_1 for style comparison). For each convolutional layer, we multiply the feature maps with the binary masks to create the guided feature maps, which we use to compute the guided Gram matrices.

4.2 Results

We applied the idea of semantic masking to doodles, where the input content image and the mask are both the colored doodle. Then, different objects in the style image transfer over to the corresponding colored regions of the doodle, and we create the transferred drawings shown in Figure 7.

Figure 7. The top layer is the style image and its corresponding mask, and the bottom layer is the doodle and the transferred drawing.

5. GIFS

GIFs (graphics interchange format) are commonly used in social media, text messaging, and presentations for short animated clips [8]. Due to its prevalence, we decided to use GIFs as a potential input to our NST pipeline. This section details the specific methods unique to the application of this pipeline to GIFs and our results.

5.1 Specific Methods

We applied the general NST pipeline to each frame of the GIF, with several key changes. First, GIFs compress colors using a color map, which means that the data must be read in a specific manner that differs from typical JPEG or PNG images [8]. Unfortunately, there are bugs in MATLAB's code to do this, so we used a specific package designed to read in and create GIFs [9]. Secondly, we made sure to initialize a separate transfer image for each frame in order to ensure adequate variance between each frame of the GIF.

5.2 Results

The results of a few frames from the GIF can be found in Figure 8. From this, we can see that NST was successfully applied to each frame, and due to the separate transfer image initialization, there is variance in the style transfer between the different frames.

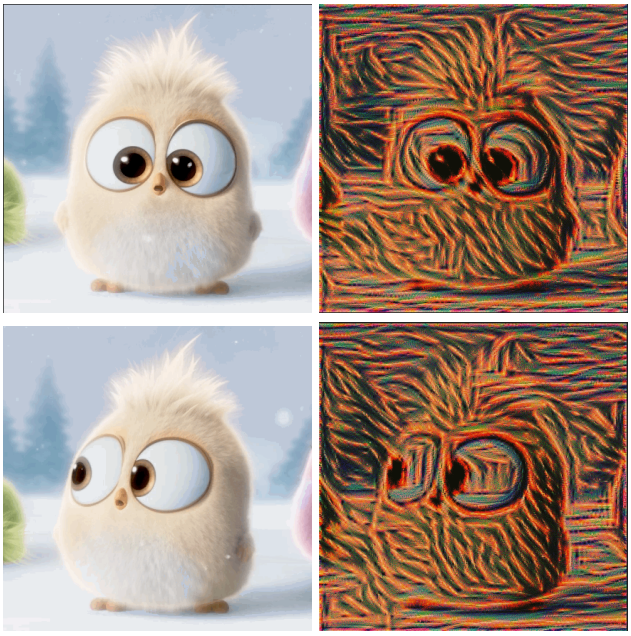


Figure 8. Frames of the Style of *The Scream* Applied to a Gif [10].

6. VIDEOS

For videos, we desired to adjust the objective of the pipeline. We decided to test gradually transferring between multiple styles. This section describes the specific methods we used to implement this, as well as our results.

6.1 Specific Methods

Similar to the specific methods for GIFs, we applied the general NST pipeline to each frame of the video. However,

since we desired to implement gradual style transfer between multiple styles, we had to modify this general pipeline to achieve this.

Our overall objective with gradual style transfer was to first start with the original image and then gradually transfer to the first style. Once this first style is fully applied to the frame, from this first style, we wanted to gradually transfer to the second style and so on. This meant that we needed to address two questions: (1) What does it mean to gradually transfer styles? (2) What does it mean to merge two different styles?

We achieved this through two primary changes. The first was that at the beginning when we were gradually transferring the first style onto the original image frame, we gradually changed the noise to content ratio during the transfer image initialization. As demonstrated in Section 3.3, increasing this ratio results in an increase in the presence of the transferred style since it is less biased toward the original image. Thus, we iteratively increased this noise to content ratio until it became equal to 7:3. We still wanted to retain the content image, so we set an upper limit of the noise to content ratio to 7:3. Additionally, when transferring from one style to another style, this noise to content ratio was kept constant at 7:3.

The second addition we made to the pipeline was to split the number of training iterations between the initial style and the style we want to gradually apply. In the beginning, when we gradually transfer the first style onto the original image frame, we increasingly increase the number of training iterations until it hits the total we set. We kept this equal to 100 for the sake of time, though this is adjustable. When transferring between two different styles, the total number of training iterations was split between the two styles. To achieve the gradual effect, the number of iterations for the initial style gradually decreased, while the number of iterations for the next style gradually increased, though the sum of both remained equal to the total number we set.

6.2 Results

The results for the gradual style transfer process on videos are illustrated in Figure 9. In these, we see the gradual application of the first style onto the original image and of the gradual transfer from one style to another style. The results of these look promising, and we expect that increasing the total number of iterations will improve the results and quality of the style transfer.

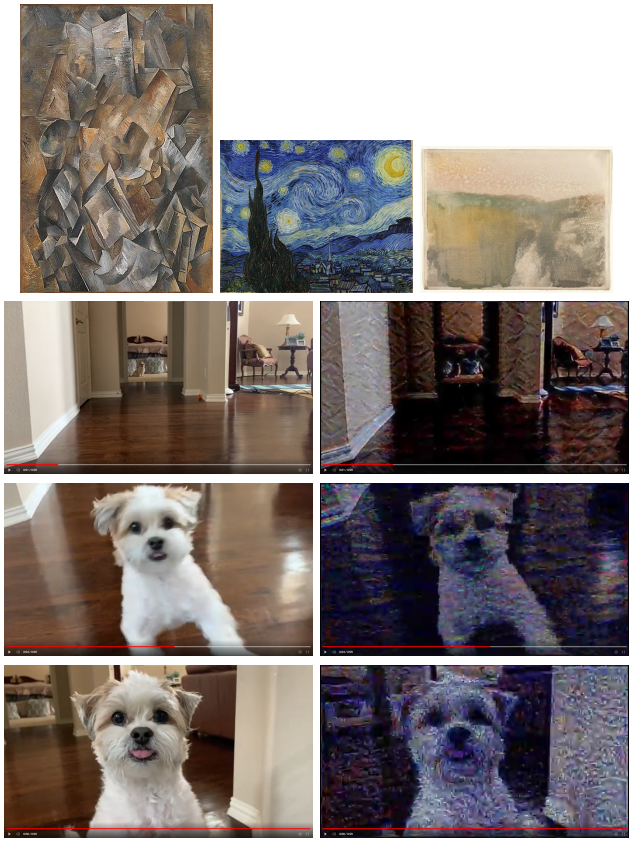


Figure 9. Neural Style Transfer Applied to Frames of a Video. (1st Row) Styles that are Transferred: Georges Braque, *Starry Night*, and Edgar Degas Respectively. (2nd Row) Application of Gradual NST of Georges Braque on to the Original Image. (3rd Row) Transfer from Georges Braque to *Starry Night.* (4th Row) Transfer from *Starry Night* to Edgar Degas.

7. PORTRAITS

This section discusses the exploration of applying style transfer to portraits. Images of faces were discussed as a point of difficulty in [11], inspiring further research through this section. The NST method was initially tried and adapted, with us eventually switching to GANs, as described below. All figures in Section 7 are ordered, from left to right, the content image, the style image, and the transferred image.

7.1 Original Pipeline

When we ran the original NST pipeline with the content image as a human face and the style image as a cartoon face, we generated odd-looking images, as seen in Figure 10.

Figure 10. NST Between Portraits.

We also generated an image using the original NST with the content image as a human portrait and the style image as an image without a face, as seen in Figure 11.

Figure 11. NST Portrait and Non Portrait.

In Figure 10, the NST code ends up extracting content features from the hair and treating the hair texture as the style to transfer to the content portrait. The combined image, as shown in Figure 11, demonstrates that the structure is also impacted, leaving certain areas like the lips blurry. We expect that with even more training iterations, the structure will be impacted further. Thus, we decided that the pipeline required significant differences specific to faces. We realized that these modifications include: (1) we need to recognize when there is a face in the content image before transferring style, (2) we need to perform regional NST instead of global, and (3) we need to consider alternative methods of style transfer that better transfer structural features.

7.2 Facial Recognition

In order to detect faces, we used MATLAB's Facial Recognition code [12] so that it would return if there was a face in the image and the bounding box of where the face was. We used the face detection code so that whenever the code detected a face in the content image, it would switch to the pipeline specific for portraits. Once the code switched to the specific pipeline for face, the pipeline further divides depending on whether the style image also had a face in it or not. The bounding box given by the facial recognition code was first used to crop the content image so that it only contained the face, not the neck or extra background. This was necessary to see if this would help improve the methods and to simplify computation.

We then ran the code with the original NST pipeline on cropped portrait content images to assess quality. The combined images were not really improved in quality versus the full images, but they improved computation time and made it easier to visually assess quality. Therefore, we decided to maintain using cropped images during the course of the project.

7.3 Regional NST

To address the issue in which the structure of the transfer image was significantly impacted during textural style transfer, we considered implementing regional NSTs. With these, the style image was constrained to images that did not contain faces. While there are regional NST algorithms that use semantic segmentation [13], we decided to segment the image using k-means clustering. The motivation for this was that semantic segmentation provides a lot more information than we need, as it predicts what each object in the image is. Additionally, not all images will have clear, predictable objects, and it would make sense to section the image based on color regions since a single object may have different elements. Using k-means clustering to define the regions accomplishes this.

We changed the original pipeline in the following manner. We used k-means clustering to define the different regions in the content image. The number of regions is predetermined in the code. Then, instead of applying the transfer to the whole image at the same time, we apply the transfer to each individual region independently. The results of this are illustrated in Figure 12. As shown in this figure, the resulting image does appear to retain the structure of the original content image, though the color has been a bit altered. Additional work will likely be required to further refine this method.

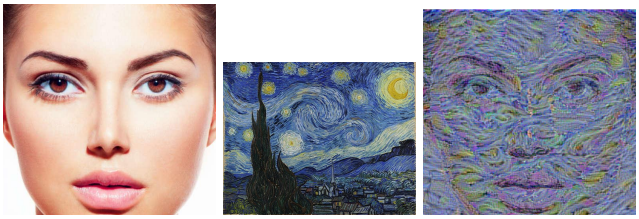


Figure 12. Regional NST Portrait.

7.4 Transferring Structural Features

As was seen in Figure 10, when attempting to use NST to transfer styles between two portraits, the NST pipeline looks for the textures in the style image to transfer them to the content image. However, this leads to visually displeasing results. What would make a more desirable style-transferred image of a face would be if the structure, rather than the texture of the style image, was transferred. We needed the eyes of the content image to change structure to match that of the style image and the same for the nose and mouth and so on. Research into this led to the discovery that GANs, or Generative Adversarial Networks, have been shown to be successful in this sort of portrait transfer [14]. We found a website that had a GAN trained on similar images to the ones we were using. Utilizing this, we generated the image in Figure 13, which illustrates the result of transferring the structure of the cartoon style to the content portrait. As can be seen in Figure 13, the eye structure was transferred with high retention of the style structure from the style to the content portrait.

Figure 13. GAN Style Transfer for Portraits [15].

8. DISCUSSION

From this project, we learned the basic process of style transfer. We also learned that style encompasses many different components. Style can consist of texture, color, medium, and many other possible aspects. However, with the single method of using a CNN to extract the style features of images, only one aspect of style, the texture, primarily dominates the transfer process. Additionally, different methods and algorithms may be better at transferring particular aspects of styles than others. As such, a single pipeline may not be sufficient to transfer all aspects of style. In other words, different algorithms may be required depending on what aspect of style we wish to transfer.

In addition to this, we learned that the input images (the content and style images) impact the quality of the style transfer and the specific method we use to implement style transfer. This is significant, as learning more details about the content and style images may be helpful for improving NST as well. These details can help predict the desired aspect of style that the user would like to transfer and better inform any preprocessing steps that may vary depending on certain features of the input. This would be an interesting potential problem to explore in the future.

9. CONCLUSION

Using a pretrained VGG-19 network to extract features from images, we found that NST was fairly successful at performing textural and color style transfer for pictures, images, doodles, gifs, and videos. However, we found that this particular algorithm performs poorly on structural style transfer. Based on these results, we proposed the exploration of additional models such as GANs, and we discussed the open-ended definition of style, which may pose a challenge in creating a consistent NST pipeline.

10. CODE

The code that we implemented and some of our outputs can be found at: https://github.com/noraagah/EE-371Q-Final-Project.

11. REFERENCES

[1] L. Gatys, A. Ekcer and M. Bethge, “A Neural Algorithm of Artistic Style,” in *arXiv preprint*, 26 Aug. 2015, arXiv:1508.06576.

[2] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu and M. Song, "Neural Style Transfer: A Review," in *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 11, pp. 3365-3385, 1 Nov. 2020, doi: 10.1109/TVCG.2019.2921336.

[3] *Image Processing Toolbox***.** (2022b), Mathworks. [Online]. Available: https://www.mathworks.com/help/images/index.html.

[4] *Deep Learning Toolbox***.** (2022b), Mathworks. [Online]. Available: https://www.mathworks.com/help/deeplearning/index.html. .

[5] *Neural Style Transfer Using Deep Learning***.** (2022b), Mathworks. [Online]. Available: https://www.mathworks.com/help/images/neural-style-transfer-using-deep-learning.html.

[6] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann and E. Shechtman, "Controlling Perceptual Factors in Neural Style Transfer," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3730-3738, doi: 10.1109/CVPR.2017.397.

[7] A. J. Champandard, “Semantic Style Transfer and Turning Two-Bit Doodles into Fine Artworks,” in *arXiv preprint*, 5 Mar. 2016, arXiv: 1603.01768.

[8] S. Neuendorffer, “CompuServe GIF Image Format.” EECS20N: Signals and Systems. https://ptolemy.berkeley.edu/eecs20/sidebars/images/gif.html

[9] *Tools to Read and Write Animated Gif Files*. (2022), DGM. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/52514-tools-to-read-and-write-animated-gif-files

[10] Gifer. https://gifer.com/en/1ABW.

[11] E. Wang and N. Tan, “Artistic Style Transfer,” in *EE 368: Digital Image Processing, Stanford University*, 2016.

[12] *Deteting Faces in Images*. (2014), Mathworks. [Online]. Available: https://blogs.mathworks.com/pick/2014/03/14/detecting-faces-in-images/.

[13] L. Kurzman, D. Vazquez and I. Laradji, "Class-Based Styling: Real-Time Localized Style Transfer with Semantic Segmentation," *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019, pp. 3189-3192, doi: 10.1109/ICCVW.2019.00396.

[14] *BlendGAN*. (2021), GitHub. [Online]. Available: https://github.com/onion-liu/BlendGAN.

[15] J. Pinkney. “Toonify!” Toonify. https://toonify.photos/