Neural Style Transfer – Artist Aggregation Technique

Seth Singson-Robbins
Fordham Univesrity
ssingsonrobbins@fordham.edu

Abstract—This paper investigates an extension to current Neural Style Transfer models where the style loss is based on an array of images instead of a single image. This is done to help get a more holistic style from an artist on pretrained Convolutional Neural Networks. The results were a mix of qualitative artistry and quantitative style and content loss to determine the best models. The VGG models for simple and squared produced the best results with the VGG Squared model gravitating towards images with similar content for its style evolution.

I. Introduction

Tensorflow defines Neural Style Transfer as 'an optimization technique used to take two images—a content image and a style reference image (such as an artwork by a famous painter)— them together so the output image looks like the content image, but "painted" in the style of the style reference image."[(1)] Based on the original paper 'A Neural Algorithm of Artistic Style' by Leon Gatyz [(2)], Neural Style Transfer is a powerful tool as it uses the power of Convolutional Neural Networks to create completely new images just through the manipulation of its pixels helping create a framework to understand both the content of the image but also the way a completely different artist or producer of the image would have approached creating the content image.

One of the drawbacks of the current Neural Style Transfer models is that it is currently only pulls the style and style loss from one image per model. This paper will be looking at a method to pull from an array of images at a time as the style reference to get a more holistic style from the selected artist. This will be based on aggregating the style loss from each image of an artist that are aggregated together into a final style loss to utilize in backpropagation of the Neural Network weights within the models. We will be looking at different options such as different pretrained models, how we are weighting the loss of different images in the style dataset, and other additional hyperparameters.

One note of the results is the lack of valid metrics to compare results of different models and some of the hyperparameters due to the nature of the models being unsupervised. Each style and content image will need its own models and weights and cannot easily creating a very individualized final Neural Network to produce the final images. Many of the hyperparameters have to do with updating the loss function or utilize different pretrained models that have vastly different style and content scores that are not comparable. This is meant to be a tool that future users can utilize as ways to tune and try out different models where the user will then determine the best fit and quality of the final images for whatever use cases needed for those images.

II. Methodology

This section looks at the additional work done to have the models run effectively. This includes how images were collected, preprocessing of the images into datasets, how the models are being trained, how we are determining and updating the loss function, information on the pretrained models we are experimenting with, and the additional hyperparameters used to optimize the models.

A. Data Collection

For any type of neural style transfer model, images need to be used for both the content and the style references to combine the images into something new. This paper utilizes the datasets provided by user Icaro from Kaggle called the 'Best Artworks of All Time'. [(3)] It includes the top 50 most influential artists and includes thousands of images spanning across a variety of sizes, time periods, and art styles. One table has the list of artists in the images dataset along with information about the artists including the years they were alive, the genre of the art, the artists' nationalities, a biographical summary of the artist, a link to the artists' Wikipedia articles, and the number of paintings available in the main dataset. The main image dataset includes images of these artists in varying sizes, styles, and different time periods throughout the artists' lives.

For the test itself, 6 random artists were picked to be utilized as the style datasets. To speed up the models, only the first 30 images of each artist was used. Random images of other artists were selected as the training and test images, 10 images for the training set and 5 images for the test set.

B. Data and Text Preprocessing

Since the dataset is a list of images, some manipulation must be done so they can be read and manipulated by the Tensorflow models accurately. To do this, they need to be converted into standardized arrays. Note that this is based on the same methodology done by the Neural Style Transfer documentation produced by Tensorflow. [(1)]

The first step is turn them into RGB arrays. This means that for every pixel in the image there is an equivalent number mapping for the color presented by that pixel. This mapping is done with three numbers that represent the saturation of Red, Green, and Blue of that color and are in a range of 1 to 255 for each color meaning three numbers are used to represent that pixel.

Each image first needs to be resized to be 224 x 224 pixels. This is due to ensuring that all the images are of the same size since they will be pushed into models that also require this specific image size to run. This can be undone through the Tensorflow resizing function and will be done when doing the test analysis. The first two steps are done with the load_images_from_drive function which pulls the datasets from google drive and then resizes the images with the PIL Image module. [4] They are all then appended to an image dataset.

Once the images have been updated, each image needs to be transformed into a Tensorflow array which requires first to turn the dataset in a 3d NumPy array of size (224,224,3).[5] Then the files are translated into an array image with TensorFlow's img_to_array function.[6] Lastly, the image pixel numbers are divided by 255 since the RGB values range from 1-255 and it's easier to model the numbers if they are in a normalized range between 0 and 1. This is all done in the custom image_to_dataset function and is undone for the final image in the tensor_to_image function.

The finalized datasets are a group of images for each style dataset and content dataset in an array of size (x, 1, 224, 224, 3) where x is the number of images in the array and can now be readable to the tensorflow models.

C. Training and Evaluation Methods

Training is not done in the typical supervised setting seen in typical supervised data science and machine learning models. One issue is that the loss can vary greatly based on not just what pretrained model is being utilized but what image is being used. This makes this analysis more unsupervised since the loss varies immensely between the content and style images making these models much more individualistic. However, averages between the training images will be utilized, and the goal is to pick hyperparameters that balance both the amount of time to run the model but also minimize the amount of total style and content lost. This is analyzed and evaluated between the different models particularly analyzing the style loss function over each epoch and which will be discussed further in the hyperparameter section.

D. Loss Function Calculations

As explained in the Tensorflow documentation for Neural Style Transfer, the main way that these models determine the performance of an image is through its loss function. This is a combination of the 'content loss' derived from the initial content image and the 'style loss' from the style image. [1]

For the content loss, the mean squared error is calculated between the feature maps of the generated image and the content image based on one of the later layers in the Convolutional Neural Network to ensure that the content is similar at the end of the model to what is in the input. Figure 1 shows the content loss function as shown in Blackburn's article in TowardsDataScience. [6]

Figure 1. Content Loss Function

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

P is the original image, F is the generated image, and l is the layer of each image.

The style loss utilizes a gram matrix, a mathematical representation that captures the correlations between different features in a set of vectors utilizing feature maps and vectorization of the datasets. This is done at the earlier or middle stages of the Convolutional Layers since we want these stylized changes to be affected earlier in the image creation while maintaining the main aspects of the content image. Figure 2 shows the style loss formula as shown in Blackburn's article. [7]

Figure 2. Style Loss Function

$$J(S,G) = \frac{1}{(2\,H^{l}\,W^{l}\,C^{l})^{2}} \sum_{k} \sum_{k'} \left(Gkk'^{[l][S]} - Gkk'^{[l][G]} \right)$$

G is the gram function, K is the original image, and K' is the generated image for each layer l.

For the models in the paper, the style loss will be determined from several images rather than just one. To do this, the style loss will need to be determined between the model and every image in the style dataset. They will then be aggregated with a variety of methods determined by an aggregation hyperparameter which will be discussed later in the model.

E. Initial Models

The original paper of the Neural Style Transfer utilized the Visual Geometry Group pretrained model, also known as VGG, as the main Convolutional Neural Network for these models. [8] This is a massive model utilizing many layers to get to the final new image. Since this is a variation of the typical Neural Style Transfer, other models will be utilized for testing possible variations including ResNet [9], Inception [10], and MobileNet [11]. ImageNet, which is a massive pretrained version of these models based on over 14 million images and is the standard for image based Neural Networks, will be used as a base for the pretraining weights. [12]

Unfortunately, each of these models will be using different layers and thus will have very different loss results that cannot

be compared to in any meaningful way. This means that the user of these models will need a qualitative metric, aka their artistic eye to pick which model works the best for achieving their goal, but this paper will try to analyze the results regardless.

F. Hyperparameters

Due to the unsupervised and individualized nature of the models, most of the hyperparameters for this paper need to be utilized to help the qualitative measurement of the images created instead of minimizing a loss function like typical supervised machine learning models. However, there are still a couple of quantitative hyperparameters available to pick the best model. The first is the optimizer, which will be decided by which causes the most efficient (fastest) way to get to the optimal minimization of the loss metric. The original documentation utilized Adam [13] so this will be tested but also tested will be Stochastic gradient descent [14] and RMSProp [15] to see if aggregated style loss requires a different optimizer than what is currently being used by Neural Style Transfer models.

The other quantitative hyperparameter is deciding the number of epochs to run for each model. Even though the loss function will most likely go down from backpropagation and a validation set is not available due to the individualized models, the loss will slow down as it reaches a plateau of how much more it can optimize. The epochs chosen for each model will be when the change in loss from the last epoch is less than 0.01% or when it reaches 100 epochs, whichever comes first.

Other than picking between the pretrained models, the other qualitative hyperparameter that will be utilized is how the style function will be aggregated. Since we have many images in a dataset, we need to get an average of these style losses.

The normal method would just be to sum up the loss functions and divide by the number of images in the dataset as shown in Figure 3. This will be called the Simple Aggregation model for style loss.

However, the model could also square each style loss, sum up the values, and then square root the final value as shown in Figure 4. This is similar to what is done when looking at mean square error by ensuring that the images that cause the highest style loss have the most influence in changing the model weights. This will be called the Squared Aggregation.

The opposite is done by doing a log of each style loss then doing the exponential of the summation to get the final loss shown in Figure 5. This will be called the Logged Aggregation.

Additional analysis will be done to ensure that the final loss values match the expectation of each aggregation function as well as a qualitative analysis of these final images.

Figure 3 – Simple Aggregation of Style Loss

$$\frac{\sum_{1}^{n} L_s^i}{n}$$

L is the style loss of each image versus the original image and n is the number of images.

Figure 4 – Squared Aggregation of Style Loss

$$\sqrt{\frac{\sum_{1}^{n} (L_s^i)^2}{n}}$$

L is the style loss of each image versus the original image and n is the number of images.

Figure 5 – Log Aggregation of Style Loss

$$exp(\frac{\sum_{1}^{n} log(L_s^i)}{n})$$

L is the style loss of each image versus the original image and n is the number of images.

III. Results

The results of the models have been evaluated and the quantitative hyperparameters were chosen based on which lead to the most efficient style and content loss from a minimization and speed perspective. Once done there are a total of 12 models, one for each of the four pretrained models (VGG, ResNet, Inception, and MobileNet) and three style aggregators (Simple, Squared, Logged) which were listed in the hyperparameters section. The additional information in Appendix A shows the performance of each optimizer for each model which, with a couple of exceptions, had clear winners and are delineated from the other optimizers for every epoch. Adam did best for 75% of the models while RMSProp performed better for two of the three VGG models (Simple and Squared) and the Inception Squared model. RMSProp might be performing better because of how it dampens gradients changes leading to better performance from higher loss functions.

After picking the optimizers, the epoch is chosen based on the average of the loss and picking an epoch that had less than 0.1% improvement between the epochs similar to how EarlyStopping CallBacks are implemented. This way it is

consistent when it stops the model but does it when the loss function typically plateaus in terms of loss improvement. The final epochs used varied from 26 for the ResNet Squared Modeled to the 100 limit for all three MobileNet models. VGG and ResNet had the lowest final epochs utilized which might mean that it does a better job hitting its optimal performance from its pre trained weights than the other models (which makes sense since VGG then ResNet are the top two models for Neural Networks on images). The loss over each model, epoch, and optimizer is in the additional documentation in Appendix B. Figure 6 below shows the final hyperparameters chosen for each model.

Figure 6 – Hyperparameter Winners

Model	Optimizer	Epochs
VGG Simple	RMSProp	36
VGG Squared	RMSProp	63
VGG Logged	Adam	100
ResNet Simple	Adam	100
ResNet Squared	Adam	26
ResNet Logged	Adam	62
Inception Simple	Adam	59
Inception Squared	RMSProp	100
Inception Logged	Adam	100
MobileNet Simple	Adam	100
MobileNet Squared	Adam	100
MobileNet Logged	Adam	100

Once these quantitative hyperparameters are picked, only qualitative parameters are left. This is highly subjective and, except for art experts, might be hard to differentiate and would be more for the user to determine. However, this paper will do some effort to decide on the models and any observations in the performance.

Of the four art styles, VGG, the original pretrained model used in the paper, looks the best. ResNet does not seem to make any significant changes while Inception and MobileNet makes very pixel-y images that do not seem to align with the styles of the style datasets. Additionally, the logged models looked to be very grain-y and not actually matching the style of the style datasets. The additional analysis will look at the results from the VGG Simple and Squared models for comparison. Appendix C contains all the final images for each of the models and 5 test images.

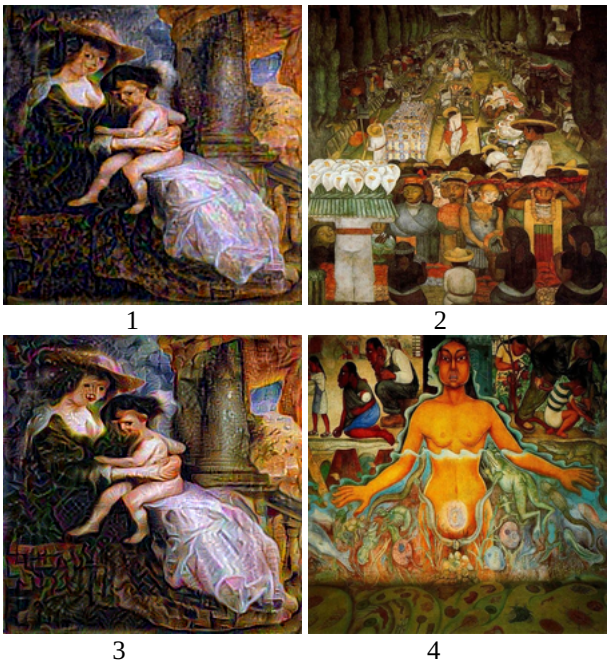
IV. Additional Analysis

One analysis that was done on top of picking the best model is, based on the list of style images used for the model, which image had the lowest style loss compared to the model image. This is done by the lowest_style_loss function which returns the index of the image in a list that has the lowest style loss compared to another image. This will help with the analysis between the different style weights.

Appendix C includes the images that had the lowest style loss for each of the final images in the size used to do the modeling (224,224,3). Between the three style weights, it should be assumed that the image that had the lowest style loss initially will have the most influence in the logged models and the least influence on the squared models (and in the middle for simple models). Based on novice analysis, I would say that the opposite occurred in the squared models while logged models did not look to have worked in an expected way as noted in the results section.

Figure 6 shows the two images for VGG Simple and VGG Squared shows the images created with the lowest loss function for those images in the Diego Rivera dataset. The VGG Squared model's lowest loss is an image focused around an individual similar to the flow to the model image. VGG Squared also has more straight lines and less curvature to the hallucinatory (the lines in more blank black and white space) aspects of the image.

Figure 6 – VGG Simple vs Squared Comparison



1 2

3 4

Image 1 – VGG Simple Model Creation
Image 2 – VGG Simple Lowest Loss Style Image
Image 3 – VGG Squared Model Creation
Image 4 – VGG Squared Lowest Loss Style Image

V. Related Work

There have been many variations and papers written on Neural Style Transfer and optimizing its capabilities. The paper 'Neural Style Transfer: A Critical Review' by Akhil Singh [16] looks at the capabilities of neural style transfer on videos and its efficiency compared to Generative Adversarial Networks. The paper 'AdaAttN: Revisit Attention Mechanism in Arbitrary

Neural Style Transfer' by Songhua Liu [17] looks at Attention optimizations to help with unwarranted distortions that occurs in the images from these changes. Lastly, on towardsdatascience there was an article called 'Mixed Neural Style Transfer with Two Style Images' by Paul Froehling [18] looks to combine two images for the style loss instead of the one in typical papers.

VI. Conclusion

Given the qualitative nature of images, it was a bit difficult to do major quantitative analysis of which models to use for this analysis. However, given the resources with the art industry and other experts in the art data science space, a metric could be developed to help analyze the style between paintings designing the quantitative metrics to help with hyperparameters that only have qualitative options for analysis. Some other analysis to further this study would be to keep going with additional training examples and running the epochs for longer, and trying other weights between the style and content losses.

Additional studies could be done to look at a more narrow or broad perspective of the style datasets. Narrow studies could look at the model's capabilities blocks of years of an artists as their artistic style transitioned. Broader studies could look at the model's capabilities of multiple artists that are in the same art periods and styles. This can help understand how generalized these models can become or if they mainly only work at a certain level of granularity.

This models have many use cases outside of just creating new works of art. It helps get the general sense of a style of the artist by looking at what the computer sees as important in the style of a group of images. It also helps identify similarities in images based on the style loss between two images to find new types of stylization or transition between art styles.

VII. References

[1] Neural style transfer, "Neural style transfer | TensorFlow Core," TensorFlow, 2019. https://www.tensorflow.org/tutorials/generative/style_transfer

[2] L. Gatys, A. Ecker, and M. Bethge, "A Neural Algorithm of Artistic Style," Journal of Vision, vol. 16, no. 12, p. 326, Sep. 2016, doi: https://doi.org/10.1167/16.12.326.

[3] "Best Artworks of All Time," www.kaggle.com. https://www.kaggle.com/datasets/ikarus777/best-artworks-of-all-time/

[4] Image Module," Pillow (PIL Fork). https://pillow.readthedocs.io/en/stable/reference/Image.html#PIL.Image.open (accessed Dec. 14, 2023).

[5] "numpy.ndarray — NumPy v1.22 Manual," numpy.org. https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html

[6] "tf.keras.utils.img_to_array | TensorFlow Core v2.7.0," TensorFlow. https://www.tensorflow.org/api_docs/python/tf/keras/utils/img_to_array

[7] Blackburn, "How Do Neural Style Transfers Work?," Medium, Aug. 04, 2021. https://towardsdatascience.com/how-do-neural-style-transfers-work-b76de101eb3

[8] "tf.keras.applications.vgg16.VGG16 | TensorFlow v2.13.0," TensorFlow. https://www.tensorflow.org/api_docs/python/tf/keras/applications/vgg16/VGG16

[9] "tf.keras.applications.resnet.ResNet101 | TensorFlow v2.13.0," TensorFlow. https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet/ResNet101

[10] "tf.keras.applications.inception_v3.InceptionV3 | TensorFlow v2.13.0," TensorFlow. https://www.tensorflow.org/api_docs/python/tf/keras/applications/inception_v3/InceptionV3

[11] "tf.keras.applications.MobileNetV3Small | TensorFlow v2.13.0," TensorFlow. https://www.tensorflow.org/api_docs/python/tf/keras/applications/MobileNetV3Small

[12] "Papers with Code - ImageNet Dataset," paperswithcode.com. https://paperswithcode.com/dataset/imagenet

[13] "tf.keras.optimizers.Adam | TensorFlow v2.13.0," TensorFlow. https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet/ResNet101

[14] "tf.keras.optimizers.experimental.SGD | TensorFlow v2.13.0," TensorFlow. https://www.tensorflow.org/api_docs/python/tf/keras/applications/inception_v3/InceptionV3

[15] "tf.keras.optimizers.experimental.RMSprop | TensorFlow v2.13.0," TensorFlow. https://www.tensorflow.org/api_docs/python/tf/keras/applications/MobileNetV3Small

[16] A. Singh, V. Jaiswal, G. Joshi, A. Sanjeeve, S. Gite and K. Kotecha, "Neural Style Transfer: A Critical Review," in *IEEE Access*, vol. 9, pp. 131583-131613, 2021, doi: 10.1109/ACCESS.2021.3112996.

[17] S. Liu et al., "AdaAttN: Revisit Attention Mechanism in Arbitrary Neural Style Transfer," openaccess.thecvf.com, 2021. https://openaccess.thecvf.com/content/ICCV2021/html/Liu_AdaAttN_Revisit_Attention_Mechanism_in_Arbitrary_Neural_Style_Transfer_ICCV_2021_paper.html

[18] P. Froehling, "Mixed Neural Style Transfer With Two Style Images," Medium, Dec. 09, 2021. https://towardsdatascience.com/mixed-neural-style-transfer-with-two-style-images-9469b2681b54