Neural Artistry: A Journey through Style Transfer

Kabir Thakur
Syracuse University
kathakur@syr.edu

Pankaj Yadav
Syracuse University
pyadav05@syr.edu

Indraneel Somayajula
Syracuse University
imsomayaj@syr.edu

Yash Rajeev Karkhanis
Syracuse University
ykarkhan@syr.edu

Abstract

This paper explores the intersection of artificial intelligence and art by implementing and experimenting with Neural Style Transfer (NST) algorithms. The objective is to blend the content of one image with the artistic style of another, creating unique art pieces. It involves an in-depth investigation of various optimizers like Adam and L-BFGS and pre-trained neural network models, such as VGG19 and AlexNet, to evaluate their impact on NST outcomes. The aim of this paper is to explore the underlying mathematics of NST by manipulating the loss function and experimenting with hyperparameters to create a spectrum of stylized images. Through this project, the authors seek to understand how different neural network architectures and NST parameters contribute to the art generation process, with the ultimate goal of achieving a nuanced understanding of NST in artistic creation.

Keywords: Neural Style Transfer (NST), PyTorch, Adam Optimizer, L-BFGS Optimizer, VGG19, AlexNet

1. Introduction

The intersection of art and artificial intelligence has led to the emergence of Neural Style Transfer (NST), an exciting field that harnesses deep learning to create visually stunning artworks. This paper explores NST's capabilities to blend distinct images' content and style, transforming conventional images into unique art pieces by drawing the styles of some contemporary artistic styles. We investigate the use of various pre-trained models like VGG19 and AlexNet, dissecting underlying feature maps from the models and experimenting with techniques such as layer selection and style weight adjustment. Our methodology involves preprocessing images for compatibility with image classification models, utilizing pre-trained networks for feature extraction, computing content, and style loss, and employing gradient descent for optimization. This choice of models is not arbitrary. To perform NST we need models that have robust filter weights which can capture the style of an image in their feature maps. The research provides insights into optimal NST strategies, balancing artistic style with content integrity, and understanding the algorithmic nuances that drive this fascinating blend of art and technology.

1.1. NST Process

Neural Style Transfer is a technique that utilizes deep neural networks to analyze and combine features from a content image and a style image, producing a target image that exhibits the features of both the content and artistic style image.

Examining the technical aspect, this technique involves several step-by-step processes, as discussed below and represented in figure 1. First, the content image and style image are chosen as the inputs, which are then fed into a robust pre-trained Convolutional Neural Network (CNN) model like VGG19. This well-known pre-trained model captures complex features of both content and style images. Next, different layers in the CNN architecture are selected to capture different levels of abstraction. Typically, style is extracted from shallow layers (for our experimentation, we chose 5 layers), and content is extracted from the deeper layers (for our experimentation, we chose the last layer). The selection of layers is a balance between capturing intricate style patterns and preserving content details. We aim to study this balance.

After the selection of different layers, the next step is Feature Map Extraction. The chosen layers are used

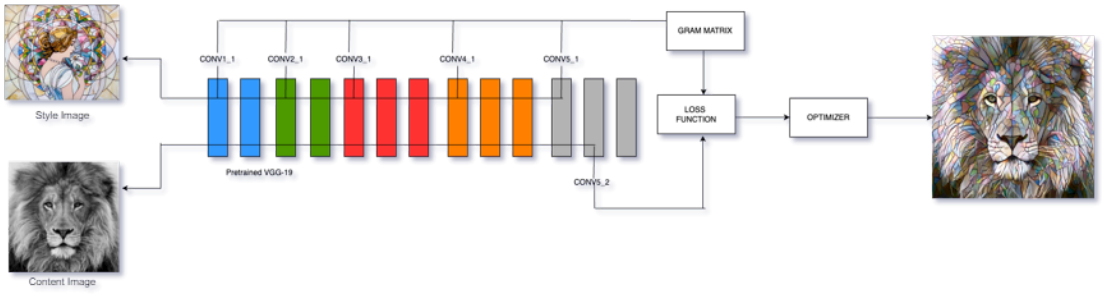


Figure 1. Flow of NST process

to extract feature maps for both the content and style images once passed through the model, and these feature maps (representing the activation of neurons in the network) are stored in the form of a Gram Matrix G(x). This Gram matrix measures the correlations between different features, obtained by taking the dot product of the feature matrix with its transpose, emphasizing the patterns and textures in the style.

Following Feature Map Extraction, a style loss, and a content loss is calculated. The final loss is the sum of the two individual losses. [2]

$$L = \alpha L_{\text{content}}(x) + \beta \sum L_{\text{style}}(G(x))$$

The final step is Optimization. Optimization algorithms such as Adam or L-BFGS are employed to iteratively adjust the pixel values of the generated image to minimize the overall loss. The objective is to converge to an image that matches the content of one image and the style of another. Throughout this process we can generate the image using the pixel value at each iteration, visualizing the change in the loss in our target image.

There are two prevalent methods for selecting the initial target image, which is then refined through backpropagation by adjusting its pixel values based on the calculated loss. One approach involves starting with an image of random noise and then shaping it into the desired output by minimizing the loss through optimization. However, this method can burden the optimizer due to the initially high loss. In our experiments, we have opted to use the content image as the starting point to simplify the loss minimization process. This approach results in an initial content loss of zero, which gradually increases as the style loss decreases.

2. Related Work

In recent times, style transfer has attracted a lot of attention due to its use in Snapchat filters and apps like Prizma, which transfers the style of an image onto a content target. This section will weave in how some papers on NST (Gatys et al. [2], Nguyen et al., 2021 [5])approach style transfer differently and what unique contributions they make to the field, contrasting with the findings from our experiments.

Nguyen et al., 2021 [5] presented a novel approach to neural style transfer (NST), utilizing Deep Feature Rotation (DFR). In this paper the authors discuss a new approach of creating multiple stylized images by using the same content and style image by rotating the feature maps before forming the gram matrix. This allows for the creation of multiple stylized outputs from a single pair of content and style images. This approach enhances the diversity of the outputs giving us more choices for the final image.

In the medium article [4], Slav Ivanov talks about what optimizers to use while running the style transfer loss optimization. The author compared various optimizers like L-BFGS, Adam, Adagrad, RMSprop, and gradient descent. The best results were for L-BFGS and Adam Optimizer. The L-BFGS method uses an approximation for the second-order derivatives also known as the hessian and Adam uses adaptive momentum. The two methods converge to the same final loss and thus the same final image starting from a completely random noise image. For the author, however, the loss for L-BFGS initially shoots up and then minimizes exponentially while the loss for Adam decreases from the initial epoch. In our experiments we aim to explore L-BFGS and Adam but with the initial image as the content image.

3. Data Description

Since we are dealing with art creation by blending two or more images, the choice of dataset is arbitrary with some constraints for style images. For the style images, we chose some contemporary art paintings like Van Gogh's Starry Night and La Cafe and some other images which displayed obvious and transferable styles. For content images, we chose a few images with scenic backgrounds and a few where there was a subject on to which the style would be transferred.

4. Methodology

The methodology for this Neural Style Transfer (NST) paper involves designing a modular class for conducting experiments with adjustable NST parameters for real-time parameter tuning and detailing a systematic experimentation process. It also includes observational techniques like loss analysis and visual transformations, with a final update to the class incorporating optimized NST parameters. The NST process described spans from image preprocessing to iterative optimization for style transfer, as shown in figure 2.

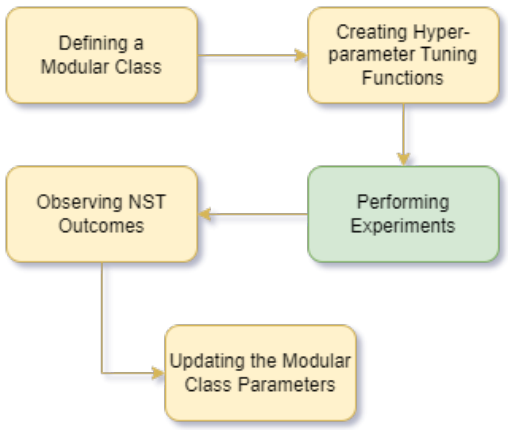


Figure 2. Methodology Flowchart

We started with the construction of a modular class specifically tailored for NST experimentation. This class was architected to encapsulate all necessary components of the NST algorithm, including image preprocessing, feature extraction, loss computations, gradient descent optimization, etc. Initial default parameters were set based on conventional practices in NST to establish a baseline for subsequent experiments.

To facilitate the exploration of NST parameter space, we implemented setter and getter functions within the class. These functions allowed us to dynamically adjust key variables such as the content-style weight ratio, layers, optimizer settings, rotation, etc. This system enabled a systematic investigation into how each parameter influenced the style transfer results.

With the class in place, we proceeded to conduct a series of experiments. By varying each parameter, we were able to observe the impact on the NST process. Each experiment was designed to isolate the effect of a single parameter while holding others constant, ensuring clarity in the resultant analysis.

The evolution of the style transfer during each experiment was meticulously recorded. We utilized loss minimization plots and generated GIFs to visualize and analyze the progression of loss minimization over time. This not only provided insight into the optimization dynamics but also served as a qualitative measure of the style transfer's success.

Post-experimentation, the class was updated with the newly optimized parameters, reflecting the most effective settings for NST as determined by our results. These parameters were then established as the default settings within the class, providing a refined tool for future NST applications.

5. Experiments

In our comprehensive exploration of Neural Style Transfer (NST), we initiated a series of experiments commencing with an evaluation of Adam and L-BFGS optimizers to determine their efficacy in our NST framework. This was followed by an analytical adjustment of the content and style loss weights, assessing their impact on the stylization process. Investigations into the selection of different network layers provided insight into their contribution to NST's performance. Additionally, we experimented with the manipulation of the Gram matrix and its implications on the visual output. The culmination of our experiments involved a novel adaptation of the loss function to facilitate the integration of two distinct styles into a single content image. The detailed methodologies, outcomes, and analytical insights garnered from these explorations will be elaborated upon in the following subsections.

5.1. Experiment 0: Comparing Optimizers

This experiment's core aim was to evaluate the effectiveness of Adam and L-BFGS optimizers during the Neural Style Transfer process, focusing on their impact on loss minimization and image stylization quality.

The qualitative assessment as shown in figure 3 revealed that images processed with the Adam

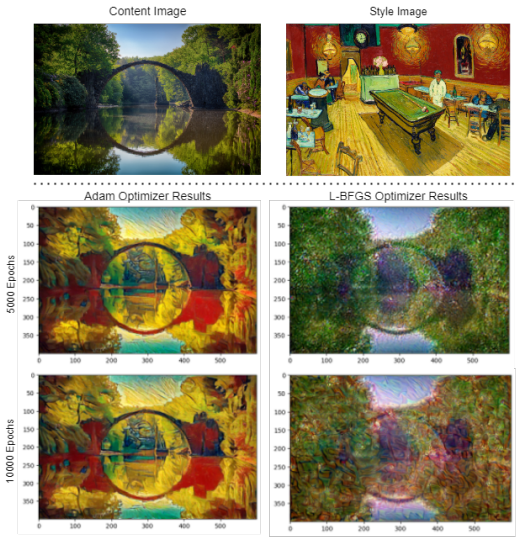


Figure 3. Initial Adam vs L-BFGS Comparison

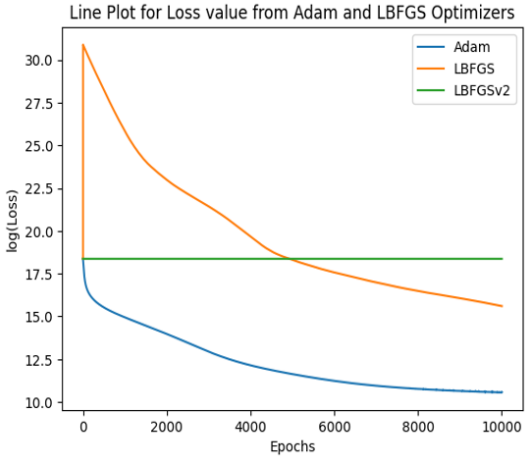


Figure 4. Loss Comparison for Adam and L-BFGS

optimizer exhibited more pronounced stylization attributes, whereas the L-BFGS optimizer tended to produce images with less style capture, resulting in a comparatively blurred effect. The significant difference between the two final images was an unexpected result. To study what caused it we dive deep into the actual loss values.

In figure 4, we see both optimizers starting at the same value for loss. The Adam optimizer minimizes the loss rapidly at the beginning and shows down after 4000 iterations. On the other hand, for L-BFGS we noticed a sharp increase in the loss after the first iteration. This was an expected result as shown by [4]. The L-BFGS optimizer minimized the loss exponentially following the sharp increase. We suspect that given enough iterations it would reach the same loss value as Adam. The purpose of this experiment however was to choose the better optimizer and clearly, Adam outperformed L-BFGS.

It's quite common for the L-BFGS optimizer to show an increase in loss during the first few iterations, especially in applications like Neural Style Transfer (NST). This behavior is due to several factors inherent to the L-BFGS optimization process:

Line Search: L-BFGS often uses a line search algorithm to find an appropriate step size. Early in the optimization, the line search might struggle to find the optimal step size, potentially leading to an initial overshoot in the loss.

Adapting to Loss Landscape: The optimizer initially needs to 'feel out' the loss landscape. Since NST involves a complex loss surface (combining content and style losses), the first few iterations might involve adjustments that temporarily increase the loss before the optimizer settles into a more effective trajectory.

5.2. Experiment 1: Alpha Beta Variation on L-BFGS

Due to the sudden spike in loss of the L-BFGS, this experiment focuses on adjusting Content and Style weights (α and β respectively) to see if they affect the performance of the L-BFGS optimizer and subsequently affect the outcome.

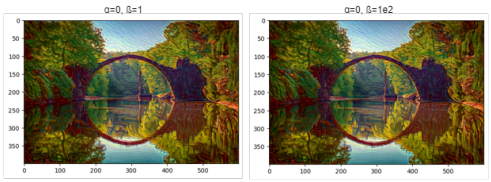
With different α and β parameters, as shown in figure 5, the L-BFGS optimizer performs much better and shows a loss convergence trend similar to that of Adam without any sudden spikes in the loss. However, the resulting images still seem to be less stylized than Adam's.

In the L-BFGS optimizer, we earlier noticed a sharp increase in loss for high α and β values, however, low α doesn't seem to have the loss spike issue. The fine-tuned values for the L-BFGS optimizer were $0 \leq \alpha \leq 1$ and $0 \leq \beta \leq 1e2$

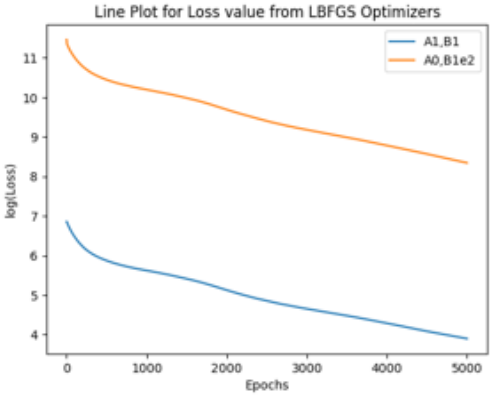
5.3. Experiment 2: Alpha Beta Variation on Adam

This experiment also aims to study the effects of varying content (α) and style (β) weights in NST using the Adam Optimizer. Since we started from the content image as our initial image, we can keep the content weight (α) as low as 0. This means that the only loss we consider would be the style loss and the image would transition more towards style.

From the above figure 6, we can see that there is not much difference between the first two images, however,



(a) Results with different α and β



(b) Losses with different α and β

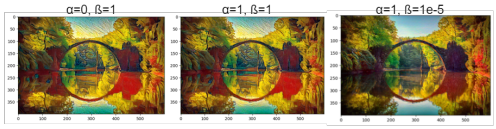
Figure 5. L-BFGS results with updated α and β

when content weight is greater than style weight more of the content image is preserved in some places and there is not much of stylization. We found that the Adam optimizer tends to converge on most values of α and β but it is generally advisable to have $\alpha \leq \beta$ so the image gets stylized.

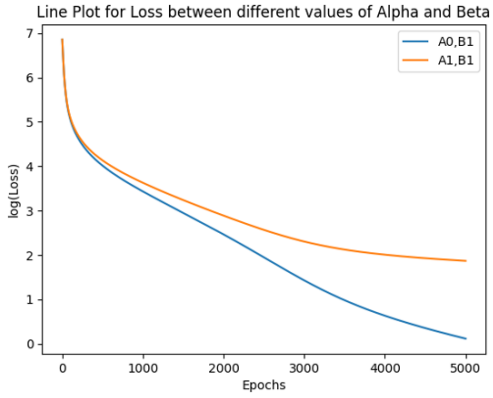
From figure 6, we can also see that the loss of $\alpha = 0, \beta = 1$ starts converging much faster than $\alpha = 1, \beta = 1$. This is because the optimizer only has to focus on minimizing the Style loss in the former case. In the case of $\alpha = 1$ and $\beta = 1$ when style loss decreases, content loss increases, and thus we reach a convergence point between the two losses.

Comparing the loss variations of Adam and L-BFGS with the same α and β parameters as shown in figure 7. We can see that now with low values of α and β, the loss for L-BFGS is not spiking and showing a similar decreasing trend as Adam. However, we can also see that the rate of decrease of loss for Adam is still much better than L-BFGS.

Based on the above comparisons and results, we can say that low values of α and β are preferred, because excessively large values of α and β don't give us significantly better results and only increase the processing time and increases the loss range. For low α and β, L-BFGS and Adam both show similar loss convergence trends, however, due to slower convergence and unpredictability of L-BFGS for high loss α and β, it



(a) Results with different α and β



(b) Losses with different α and β

Figure 6. Adam results with updated α and β

is preferable to use Adam Optimizer.

5.4. Experiment 3: Layer Selection

In this experiment, we aimed to observe the effect of choosing different style layers and understanding which layers are able to best represent the style of an image. To do an in-depth comparison we worked with two models here, namely VGG19 and AlexNet. We chose these models because their features have similar architecture.

As shown in figure 8, the VGG model which came out in 2014 has 19 convolution layers while AlexNet which came out in 2012 has only 5 convolution layers.

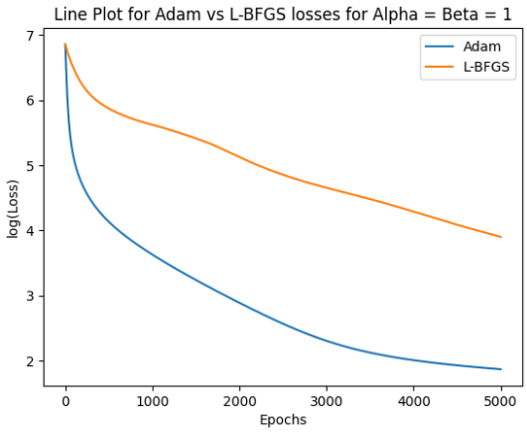
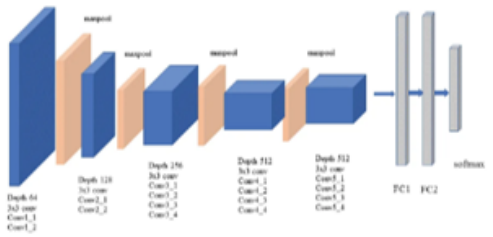
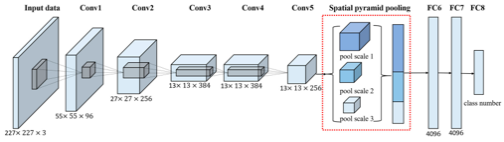


Figure 7. Loss Comparison for Adam and L-BFGS



(a) VGG19 [1]



(b) AlexNet [3]

Figure 8. Architectures of VGG19 and AlexNet

We ran experiments with various combinations of style layers and layer weights.

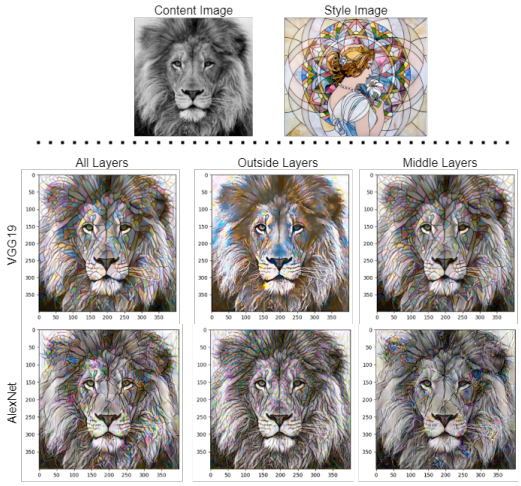


Figure 9. Results of Layer Selection

The results shown in figure 9 tell us that for VGG19 choosing the middle layers or all layers captures the style equally well. Choosing the outside layer, however, does not work. For the AlexNet model, we can see that since it has a much smaller architecture even the outside layer is able to capture style elements. The AlexNet results show us that the outside layers are able to capture the colors well while the inside layers do a better job of capturing the structure of the style image. Choosing all the layers of AlexNet captures a decent representation of the style (both color and structure) however VGG19 outperforms AlexNet as it is able to capture the nuances in style which AlexNet is not. These observations were qualitative and had no quantifiable measures.

5.5. Experiment 4: Gram Matrix Manipulation

In this experiment, we attempt to generate different styles from the same image by rotating the Gram Matrix. To capture style, we use a Gram matrix which is calculated using the feature map from the intermediate layers. First, the feature matrix is reshaped into a 2D matrix by multiplying the width and height into one dimension. The reshaped feature map is then multiplied with its transpose to get the gram matrix using the below formula.

$$G(x) = \text{FeatureMap}(d, w * h) \cdot \text{FeatureMap}(d, w * h)^T$$

To generate a different style, we rotate the above feature map while computing the gram matrix for either the style image or the target image. The style loss is then minimized as the mean squared error between the $G(x)$ of the style image and G(R(x)) of the target image where R is the rotation function used to rotate the feature map. The angle between the feature maps leads to variation in the transfer of style. Since we are computing MSE between two square matrices we stick to rotations in increments of 90 degrees.

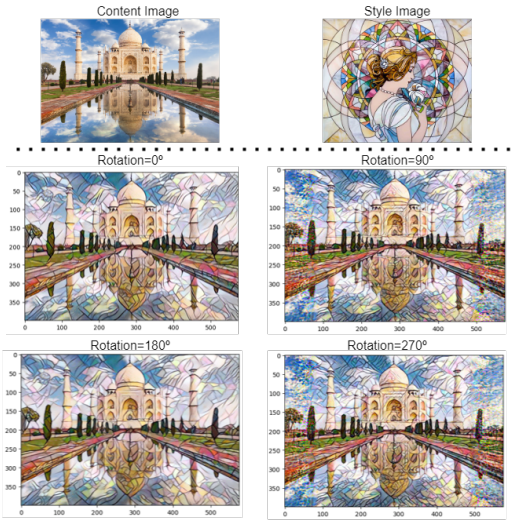


Figure 10. Results of Rotating Gram Matrix

From the above output figure 10, we can see the images rotated by 0° and 180° are similar and the images rotated by 90° and 270° are similar. However, the rotated images are simply distorted and there is not much difference in style between them. Similar results were observed [5].

5.6. Experiment 5: Using Multiple Style Images

In this experiment, we merge two style images into a single content image. The loss function used so far is given below

$$Loss = \alpha L_{\text{content}}(x) + \beta \Sigma L_{\text{style}}(G(x))$$

To add another style image we introduce another term into the loss function which captures the style of the second image. The new loss function is given below.

$$Loss = \alpha L_{\text{content}}(x) + \beta \Sigma L_{\text{style1}}(G(x)) + \gamma \Sigma L_{\text{style2}}(G(x))$$

Beta and Gamma determine the degree of weights of the first and second-style loss from the two-style images.

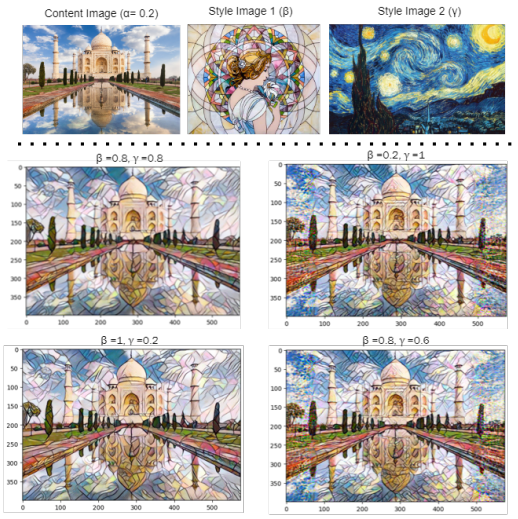


Figure 11. Using two style images

The output in figure 11 shows two styles that can be merged into a single image producing a new image. We also experimented with various values for β and γ to see how the images change based on these style weights.

6. Conclusion

In our investigation, we compared different methods to find the best way to apply artistic styles to images. We discovered that the Adam optimizer was the more efficient out of the two, striking a fine balance between final image style and clarity. Careful hyper-parameter tuning was essential to achieving high-quality results without overextending our computational resources. The VGG19 model, renowned for its detail-capturing ability, although not the fastest, was instrumental in producing style-rich images. We also successfully combined elements from two different styles into a single image, enriching the versatility of our approach. These insights have been pivotal in establishing a reliable method that harnesses the strengths of VGG19 and the Adam optimizer, fine-tuned to enhance the artistic quality of images with efficiency.

7. Discussion

The process of experimentation is never over. While our study has shown potential, it is essential to acknowledge that there is significant room for improvement. More experimentation with different optimizers and starting images will give us more insight into the loss minimization process. More experiments can be done with the manipulation of gram matrices to see how the style patterns change with each change in the gram matrix.

References

[1] Zhikun Ding et al. “An Artificial Intelligence-Based Method for Crack Detection in Engineering Facilities around Subways”. In: *Applied Sciences* 13.19 (2023), p. 11002.

[2] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. “A neural algorithm of artistic style”. In: *arXiv preprint arXiv:1508.06576* (2015).

[3] Xiaobing Han et al. “Pre-trained alexnet architecture with pyramid pooling and supervision for high spatial resolution remote sensing image scene classification”. In: *Remote Sensing* 9.8 (2017), p. 848.

[4] Slav Ivanov. *Picking an optimizer for Style Transfer*. 2017. URL: `https://blog.slavv.com/picking-an-optimizer-for-style-transfer-86e7b8cba84b#.hjzd89nba`.

[5] Son Truong Nguyen, Nguyen Quang Tuyen, and Nguyen Hong Phuc. “Deep feature rotation for multimodal image style transfer”. In: *2021 8th NAFOSTED Conference on Information and Computer Science (NICS)*. IEEE. 2021, pp. 260–265.