Exploring Convolutional Neural Networks for Image Style Transformation

Samarth Kumar, Cliffton Lang, Jannatul Fhirdose Mohammad

December 6, 2024

1 Introduction

Gatys et al. introduced Neural Style Transfer (NST) as a framework for combining the semantic content of an image with the artistic style of another. Their algorithm formulates NST as an optimization problem, minimizing a combined loss function that incorporates both content and style losses. Content is extracted from intermediate layers of a CNN, preserving spatial structure, while style is encoded by a Gram matrix, capturing correlations between the feature maps to represent textures, colors, and patterns [1]. This technique opened new avenues for creative expression in digital art and media. We hope to build upon Gatys' foundational approach and aim to optimize the NST process further. Through the exploration of variations in content and style loss weights, along with computational refinements, we plan to address the algorithm's computational inefficiencies while expanding the applicability of NST towards a large and diverse dataset.

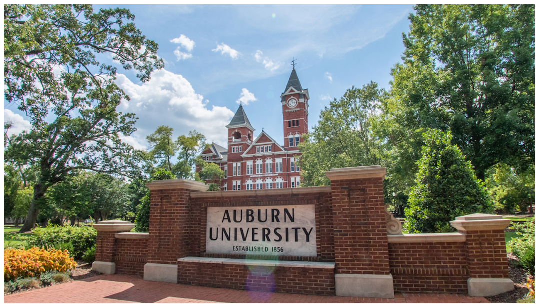
Our implementation is based on the VGG19 Convolutional Neural Network (CNN), a powerful model for image classification in computer vision [2]. VGG19 was pre-trained from the ImageNet dataset, which allows the CNN to perform tasks like pattern recognition and identify structures or textures within images. For a Neural Style Transfer, the model can separate and recombine the content and style of two images. The main goal is to create a new image that combines the content of one image (structure, objects, layouts), with the artistic style of another (colors, texture, patterns). The resulting image, formed as a result of the combination of a content and style image, enables the model to blend the semantics of the content image with the artistic features of the style image.

Deep learning and convolutional neural networks have been transformative in the domain of computer vision, especially for tasks like neural style transfer by using hierarchical representations of images. CNNs, such as the VGG19 model, extract features in lower layers, like texture and edges, to semantic content from deeper layers, which enables isolation and combination of content and style. Though other frameworks, such as ResNet, have been explored for enhancing efficiency, we relied upon the VGG19 model as it was based on the original approach from Gatys et al.

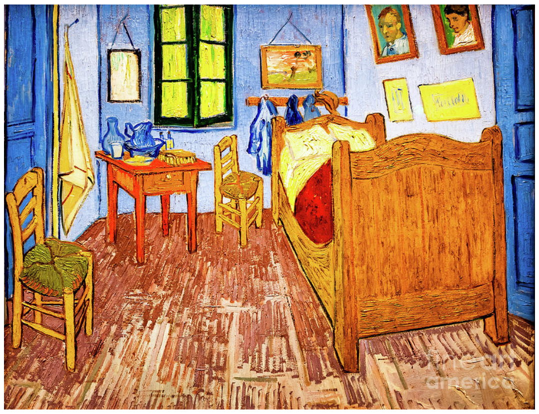
2 Methodology

2.1 Dataset

We incorporated two datasets for this project. First, we used a small dataset that included the same images utilized by Gatys et al. The small dataset consisted of one content image and four style images. Additionally, we added a large dataset for the purpose of scalability and efficiency testing. This large dataset includes two content images and eight style images, where each content image was transformed with the eight styles. Similar to the original dataset, we selected various famous artworks as our style images in order to provide a wide range of artistic styles to train our model with. Content images can be any images with identifiable objects and structures. We chose an image of Auburn University as our content image as an example with architecture and another content image depicting an Alpine sunset, as an instance that is purely a landscape. Each image we selected was in the .jpeg format and found either on the internet or from Gatys et al. for the smaller dataset. Figures (a) and (b), below, represent one of the content and style images used in the larger dataset we selected for our project.



(a) Content Image: Auburn University



(b) Style Image: Vincent Van Gogh's *Bedroom in Arles*

2.2 Algorithm

The loss function quantifies the quality of the generated image by comparing it to the content and style images, ensuring the features are blended efficiently. It comprises two components: *content loss* and *style loss*. Content loss measures the structural and overall similarity between the generated and content images using feature activations from a pre-trained VGG19 model. These activations capture textures, edges, and patterns, with deviations leading to higher loss. Mathematically, content loss is calculated as the mean squared error (MSE) between the feature maps of the content and generated images.

Style loss evaluates how closely the generated image aligns with the style image by comparing their stylistic features using a Gram matrix. This matrix captures correlations between feature maps, representing color patterns and textures. By computing the MSE between the Gram matrices of style and generated images across multiple layers of VGG19, the model minimizes style discrepancies. The total loss is a weighted combination of content and style losses, expressed as:

$$L_{\text{total}} = \alpha L_{\text{content}} + \beta L_{\text{style}} \tag{1}$$

The algorithm ultimately aims to minimize the total loss while balancing content and style losses so that the final image does not prioritize either content or style in terms of influence on the final resulting image. The model followed a sequence of five steps throughout the training process.

1. **Image Initialization:** The model begins with the content image, or a randomly initialized image. This image is to be modified during training, leading to a resulting image that blends the characteristics for the desired content and style.

2. **Forward Pass and Loss Calculation:** The content, style, and generated images pass through the VGG19 network for extracting the feature activations from each layer. The feature activations are then used to calculate the content and style losses.

3. **Backpropagation and Gradient Descent:** Using the calculated losses, the model performs backpropagation to compute the gradients of the loss with respect to the pixel values of the generated image. These gradients indicate how the generated image should be adjusted to reduce the loss. Gradient descent is then used to update the pixel values of the image in the direction that minimizes the loss.

4. **Iterative Refinement:** This step is repeated over many iterations (normally hundreds or thousands), where the image is slowly refined in order to minimize content and style losses. The generated image approaches the desired content and style combination as the training continues.

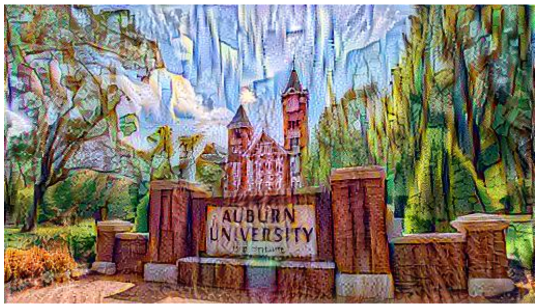
5. **Final Output:** After the model completes the training process, the final image should demonstrate the key features from the content image with the artistic style from the style image. To ensure that pixel values conform within a valid range and that the image is ready to display or be processed further, the final image is clipped.

2.3 Configuration

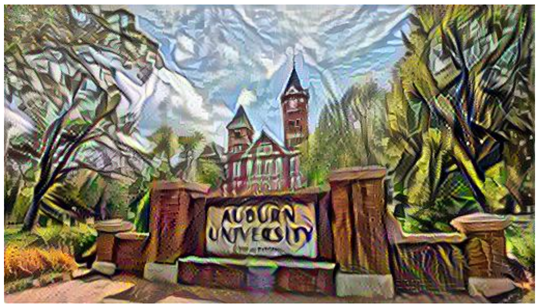
We conducted our experiment mainly in a Jupyter notebook environment and programmed in the Python language. We employed TensorFlow as the primary Deep Learning framework for constructing and training our neural network. This configuration provided a dynamic and interactive environment for experimentation and visualization.

3 Evaluation

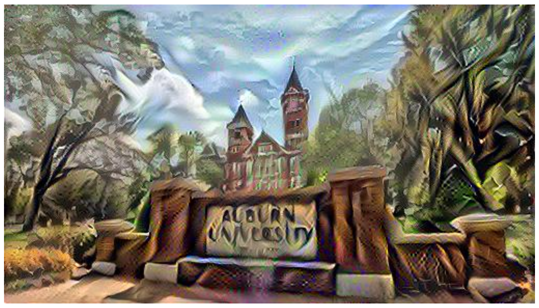
The model was able to effectively balance content and style loss when producing visually compelling images that combined the structural elements of the content image with the artistic features of the style images. Below, we present a few of the examples of our combined outputs, using style images like Vincent Van Gogh's *Bedroom in Arles* and others with the content image of Auburn University. The combined images demonstrate the model's ability to retain key content features such as building structures and layouts while infusing them with the artistic characteristics—color palettes, brushstrokes, and patterns—from the style images.



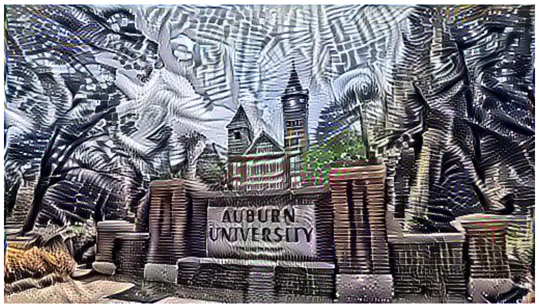
(c) Combined image using Vincent Van Gogh's *Bedroom in Arles* as the style image



(d) Combined image using Pablo Picasso's *The Weeping Woman* as the style image



(e) Combined image using Salvador Dali's *Spider of the Evening* as the style image



(f) Combined image using Maurits Cornelis Escher's *The Dream* as the style image

For a quantitative approach, we measured our model's performance using content and style losses. The plot shows the evolution of style and content loss over the training steps. The content loss is represented by a solid line, while the style loss is shown by a dotted line. Both losses are plotted with the same color to indicate their pairing for the same image. We notice a significant decrease in style and content loss after the first epoch, indicating that the model's ability to retain content and apply the desired style has rapidly improved. After the initial drop, the losses plateaued and remained steady across the remaining iterations. The steady behavior suggests that the model has efficiently learned to balance the content and style features, and further training primarily fine-tunes the model without significant loss reduction. The stabilization in loss curves indicates that the model has converged and that further adjustments would have minimal impact on loss improvement.

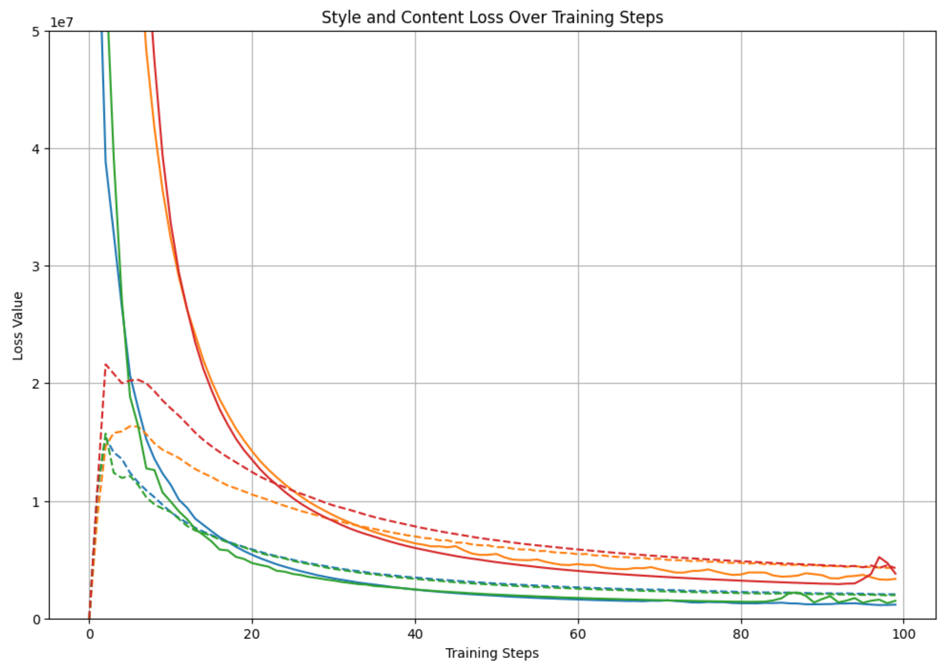


Figure 1: Style and Content Loss Curves for 100 Training Steps

4 Discussion

4.1 Strengths and Weaknesses

The neural style transfer algorithm is highly effective at combining the artistic style of one image with the structural content of another, producing visually compelling results. Its ability to run efficiently on modern hardware enables fast processing, making it suitable for generating high-quality outputs in a relatively short amount of time. The algorithm excels at clearly preserving the key elements of the content image while seamlessly integrating the textures, colors, and patterns from the style image, resulting in outputs that are both consistent and visually stunning.

Despite its strengths, the algorithm has certain limitations. A separate optimization process must be performed for each pair of content and style images, requiring a time-intensive setup for new combinations. Additionally, if the model is over-optimized or run for too many iterations, it can lead to the degradation of fine details within the content image, such as text or intricate structures, as these details become overwhelmed by stylistic elements. This sensitivity highlights the need for careful parameter tuning to ensure balanced outputs.

4.2 Summary

This project implemented a neural style transfer algorithm based on the foundational work of Gatys et al. (2016). The approach used a pre-trained VGG19 convolutional neural network to extract hierarchical features from input images, separating content and style. Content features preserved the structural layout of the original image, while style features, captured through Gram matrices, encoded textures and artistic patterns.

A combined loss function, balancing content and style losses, guided the optimization process. Content loss measured deviations in structural features between the content and generated images, while style loss quantified alignment with the texture and style of the style image. The relative weights of these losses (α and β) were tuned to control the trade-off between content preservation and stylistic transformation.

The iterative optimization framework produced high-quality outputs, blending the content of one image with the artistic attributes of another. By running hundreds of iterations, the algorithm achieved a balance between retaining key content features and seamlessly integrating style. The visual results demonstrated the model's capacity to combine content and style effectively across various images.

Quantitative evaluations supported the success of this approach, showing rapid reductions in content and style loss during the initial optimization phase, followed by stabilization. These trends mirrored those observed in Gatys' original work. However, the computational cost of the iterative process remained significant, with runtime scaling with image resolution and the number of iterations.

Despite these challenges, the implementation showcased the robustness of neural style transfer for artistic applications. Future work could focus on improving computational efficiency, exploring alternative architectures, or developing real-time solutions, further expanding its potential use cases and applicability.

References

[1] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 2414–2423.

[2] A. Kaushik, "Understanding the vgg19 architecture," in *OpenGenus IQ*, 2024. [Online]. Available: https://iq.opengenus.org/vgg19-architecture/