Style Transfer Neural Networks Report

1. Introduction and Background

Neural Style Transfer (NST) is a technique that synthesizes a new image by combining the structural content of one image with the artistic style of another. The concept was formalized by **Gatys, Ecker, and Bethge (2015)**, who demonstrated that deep convolutional neural networks contain hierarchical feature representations capable of disentangling high-level content information from low-level texture statistics [1]. Their work showed that the correlations between feature activations captured through **Gram matrices** effectively represent artistic style, while deeper feature maps preserve semantic content. This seminal formulation established NST as a new direction at the intersection of computer vision, computer graphics, and computational creativity.

Following the original optimization-based method, which requires hundreds of gradient descent iterations for each stylized image, subsequent research focused on improving efficiency. **Johnson et al. (2016)** introduced a **feed-forward generative model** trained using perceptual losses, enabling real-time style transfer at interactive speeds [2]. Their results demonstrated that learning an explicit transformation network drastically reduces computational cost while maintaining high visual quality. Later, **Ulyanov et al. (2017)** showed that **Instance Normalization** plays a crucial role in stylization quality, providing insights into the relationship between feature statistics and artistic style [3].

A significant milestone in NST research is the development of **arbitrary-style transfer** methods, where a single model can apply an unlimited number of artistic styles without retraining. A key contribution in this direction is **Adaptive Instance Normalization (AdaIN)** proposed by **Huang and Belongie (2017)**, which aligns the mean and variance of content features to those of style features to achieve flexible, fast stylization [4]. Subsequent approaches, such as **Whitening and Coloring Transform (Li et al., 2017)** [5], further generalized this idea by matching full covariance statistics in feature space.

More recent architectures leverage attention and transformer mechanisms to capture long-range dependencies between content and style features. For example, **StyTR² (Deng et al., 2022)** adapted Vision Transformers (ViTs) to NST, achieving high-quality stylization with improved semantic alignment [6]. These methods represent the current frontiers of NST research and illustrate a shift from pure statistical matching to more expressive feature correspondence modeling.

Overall, the evolution of NST, from optimization-based algorithms to fast feed-forward and transformer-based architectures, reflects a broader trend toward more controllable, efficient, and semantically aware image synthesis.

2. Model Explanation

2.1. Open-Source Model

https://github.com/nazianafis/Neural-Style-Transfer
In this project, NST network uses a pretrained VGG19 model to extract content and style features from images. VGG19 is a convolutional neural network (CNN) originally trained on ImageNet, with a total of 19 layers to capture different types of visual information. Early layers respond to colors, edges, and textures, while deeper layers respond to more abstract structures such as shapes and object layouts.

2.2. Model architecture

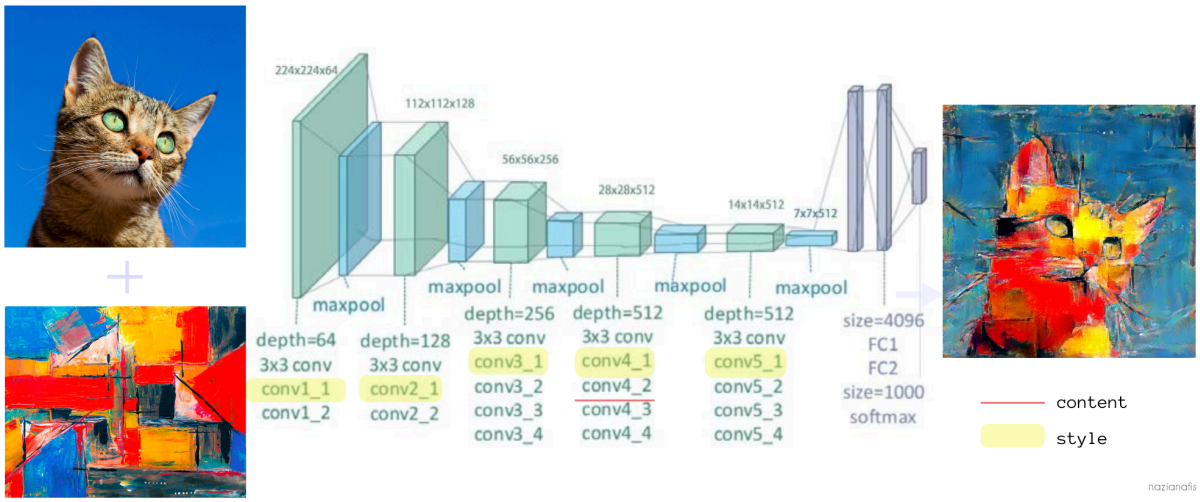

Fig 1. VGG19 model architecture with input and output. Copied from nazianafis's page.

To access the intermediate feature maps needed for style transfer, the VGG19 feature extractor is divided into six sequential blocks (`slice1` to `slice6`). Each block contains a group of convolution and activation layers taken directly from the pretrained VGG19 model. During a forward pass, the input image is passed through these slices one by one, and the model records the output of the layers that are used for style and content representation.

Specifically, the implementation returns:

- **content features** from `conv4_2`
- **style features** from `conv1_1`, `conv2_1`, `conv3_1`, `conv4_1`, `conv5_1`

These layers are selected because they capture texture information at different scales (for style) and structural information needed to preserve the content of the input. All parameters of VGG19

are frozen (`requires_grad=False`) so the network acts only as a feature extractor. The generated image is the only component that will be optimized.

The forward method returns the selected feature maps as a named tuple, making it easy to compute content and style loss using these intermediate activations.

2.3. How the model is trained

Unlike typical neural network training, where model weights are updated, neural style transfer optimizes **only the generated image**. The goal is to adjust the pixel values of the generated image so that it matches the content of one image and the style of another.

The training workflow proceeds as follows:

Step 1: Initialize the generated image
Begin with a copy of the content image (or sometimes random noise). This image is treated as the variable to be optimized.

Step 2: Extract features using VGG19
The content image, style image, and generated image are each passed through the VGG19 model. The model outputs a tuple of feature maps from the selected layers.

Step 3: Compute the loss function
The total loss is a weighted combination of:

- **Content loss** ensures the generated image preserves the spatial structure of the content image.
- **Style loss** computed using Gram matrices of the feature maps, encouraging the generated image to match the textures and colors of the style image.
- **Total variation loss** adds smoothness and reduces artifacts.

Step 4: Optimize the generated image
An optimizer such as L-BFGS (used in this network) or Adam is used to update the pixel values of the generated image by backpropagating the gradients of the total loss. Importantly, **VGG19's parameters remain unchanged throughout the process** since it only serves as a fixed feature extractor.

Through this iterative optimization process, the generated image gradually evolves to combine the structure of the content image with the appearance of the style image.

2.4. NST network architecture

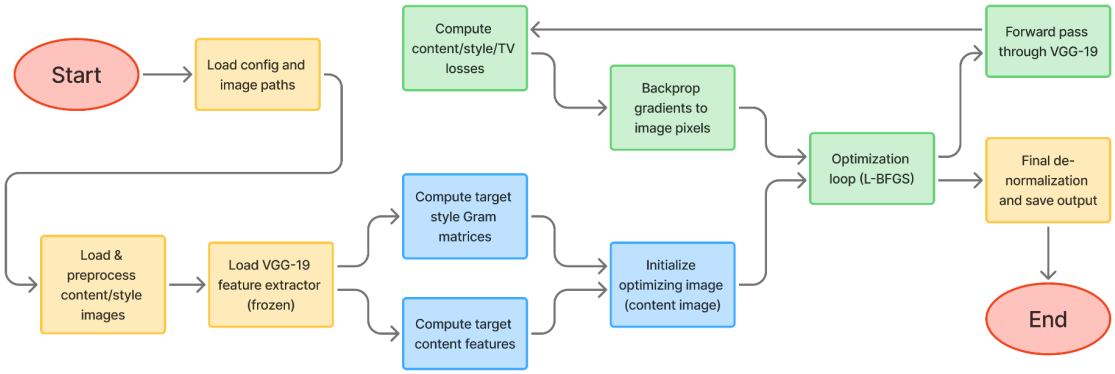

Fig 2. Overall pipeline of the neural style transfer network.

I also added another block in the end to plot the loss curves by iterations as **Fig 4**.
More detailed explanations can be found in the following colab notebook.
https://colab.research.google.com/drive/1QiPBdz0j3gCa3EoPT8pGmZ63mPy01jAj?usp=sharing

3. Test and Discussion

3.1. Test with provided images

Fig 3. Content image + style image + result image.

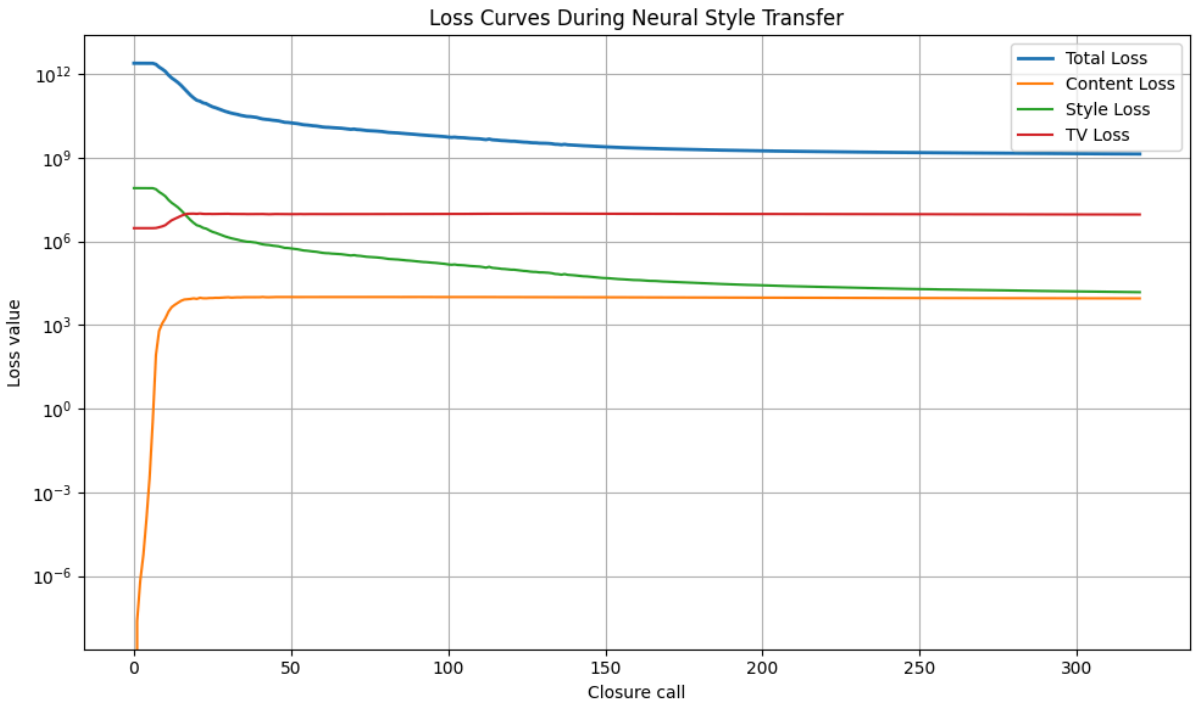

Fig 4. Four loss curves by iterations for the provided images' testing.

Although the original implementation ran **1000** iterations, the plot shows that most losses converge much earlier, around **300** iterations. After this point, the losses decrease only slightly and the output image does not change noticeably. Based on this observation, I reduced the number of iterations from 1000 to 300 for all subsequent experiments. This adjustment significantly improved runtime while still producing visually satisfactory stylization results.

3.2. Test with Own Data

3.2.1. Content Images

Fig 5. C1 - Content picture of UCI building of Engineering School.
Fig 6. C2 - Content picture of my cat Darf watching TV.

3.2.2. Style Images

Fig 7. S1 - Cartoon style picture. Michel H. Devoret. Nobel Prize in Physics 2025.
Fig 8. S2 - Van Gogh style picture.

3.2.3. Result Images

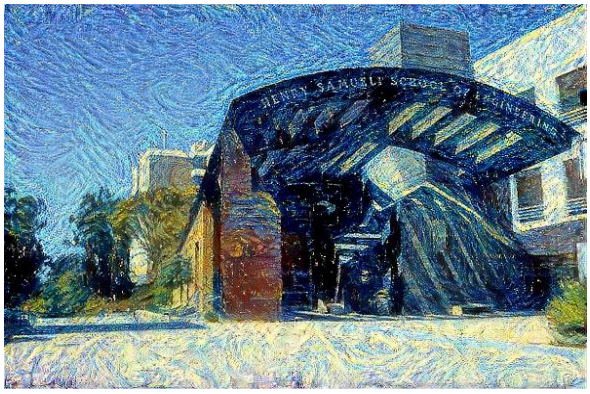

Fig 9. Output pictures C1-S1 and C1-S2.

Fig 10. Output pictures C2-S1 and C2-S2.

3.2.4. Result Discussion

From the results, we can see that the Van Gogh style produces much stronger stylization compared to the cartoon style. This is likely because *Starry Night* contains rich textures and high-frequency patterns that are easy for the VGG19 feature extractor to capture and transfer. As a result, the generated images show clear swirling brush strokes and strong color patterns. In contrast, the cartoon style image is very simple and has large flat regions with fewer textures. Since neural style transfer relies on matching feature statistics extracted by the convolutional layers, a simple style image provides fewer style cues. This makes the final output look closer to the original content image, with only mild changes in color and line appearance.

Overall, detailed and texture-rich styles transfer more effectively, while simple styles lead to more subtle results.

4. Conclusion and Future Work

In this project, I explored and tested an open-source neural style transfer implementation from Github open-source by Nazia Nafis. By analyzing and running the code, I was able to understand how a pretrained VGG19 model can be used to extract content and style features and how the generated image is optimized through iterative updates. The experiments showed that styles with rich textures, such as Van Gogh's *Starry Night*, transfer much more strongly than simpler cartoon styles. I also found that about 300 iterations were enough to produce stable results, which helped reduce runtime.

For future work, it would be interesting to try faster feed-forward style transfer models, test more varied style images, and experiment with different loss weight settings to improve stylization quality. Extending the method to video or trying more modern approaches such as AdaIN could also make the technique more flexible and efficient.

Reference

[1] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge. *A Neural Algorithm of Artistic Style.* CVPR 2016 / arXiv:1508.06576.
[2] Justin Johnson, Alexandre Alahi, Li Fei-Fei. *Perceptual Losses for Real-Time Style Transfer and Super-Resolution.* ECCV 2016.
[3] Dmitry Ulyanov, Andrea Vedaldi, Victor Lempitsky. *Instance Normalization: The Missing Ingredient for Fast Stylization.* arXiv:1607.08022, 2017.
[4] Xun Huang, Serge Belongie. *Arbitrary Style Transfer in Real-Time with Adaptive Instance Normalization.* ICCV 2017.
[5] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, Ming-Hsuan Yang. *Universal Style Transfer via Feature Transforms.* NIPS 2017.
[6] Yifan Deng, Hang Zhou, Di Liu, et al. *StyTR²: Unbiased Image Style Transfer with Transformers.* CVPR 2022.