Style Transfer using Neural Networks: Implementation and Hyperparameter Analysis

Gaurav Patil
012629189
CSCI611 - Spring 2025

March 25, 2025

GitHub Repository for Submission

Project code and additional assets are available at:

GitHub Repository - Assignment 4

Contents

1 Introduction

Style transfer is a technique in computer vision that applies the style of one image to the content of another. This method was popularized by Gatys et al. (2016), who used a pre-trained convolutional neural network (VGG-19) to extract and recombine content and style features. This report documents the implementation of neural style transfer and the effect of different hyperparameters on the quality and appearance of the generated image.

2 Input Images

Figure 1 shows the content and style images used for generating stylized outputs.

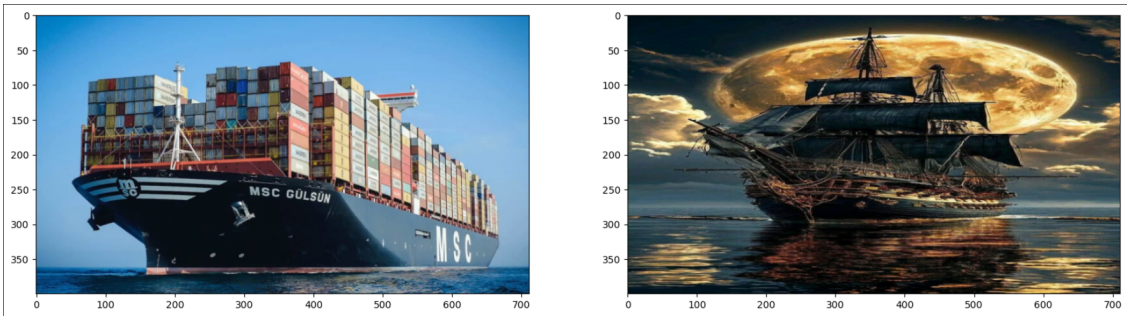

Figure 1: Left: Content Image (Cargo Ship), Right: Style Image (Fantasy Ship with Moon)

3 Methodology

3.1 Feature Extraction

The VGG-19 network, pre-trained on ImageNet, is used to extract both content and style features. Specific layers are chosen to represent content and style:

- **Content representation:** ‘conv4_2‘
- **Style representation:** ‘conv1_1‘, ‘conv2_1‘, ‘conv3_1‘, ‘conv4_1‘, ‘conv5_1‘

3.2 Gram Matrix

The style is captured using a Gram matrix, which computes the correlations between feature maps. For a feature map tensor of shape (C, H, W), it is reshaped into $(C, H \times W)$ and multiplied by its transpose to yield a (C, C) Gram matrix.

3.3 Loss Functions

- **Content Loss:** Mean squared error between content features of the target and original content image at ‘conv4_2‘.

- **Style Loss:** Mean squared error between the Gram matrices of the target and style images, aggregated across selected layers, each weighted differently.
- **Total Loss:** Weighted sum of content and style loss:

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

3.4 Optimization

The target image (initialized as a clone of the content image) is updated using the Adam optimizer. Only the target image's pixels are updated; the model remains unchanged.

4 Implementation Details

4.1 Libraries Used

- PyTorch
- torchvision
- matplotlib
- PIL

4.2 Code Snippets

Gram Matrix:

```
def gram_matrix(tensor):
    b, d, h, w = tensor.size()
    features = tensor.view(d, h * w)
    gram = torch.mm(features, features.t())
    return gram
```

Layer Mapping for Feature Extraction:

```
layers = {
    '0': 'conv1_1',
    '5': 'conv2_1',
    '10': 'conv3_1',
    '19': 'conv4_1',
    '21': 'conv4_2',  # content layer
    '28': 'conv5_1'
}
```

5 Hyperparameter Experiments and Results

5.1 Tested Configurations

We experimented with the following hyperparameter settings:

- Content weight (α): 1, 10, 100
- Style weight (β): 1×10^6, 1×10^7, 1×10^8
- Steps: 2000, 3000, 4000, 5000

5.2 Output Comparisons

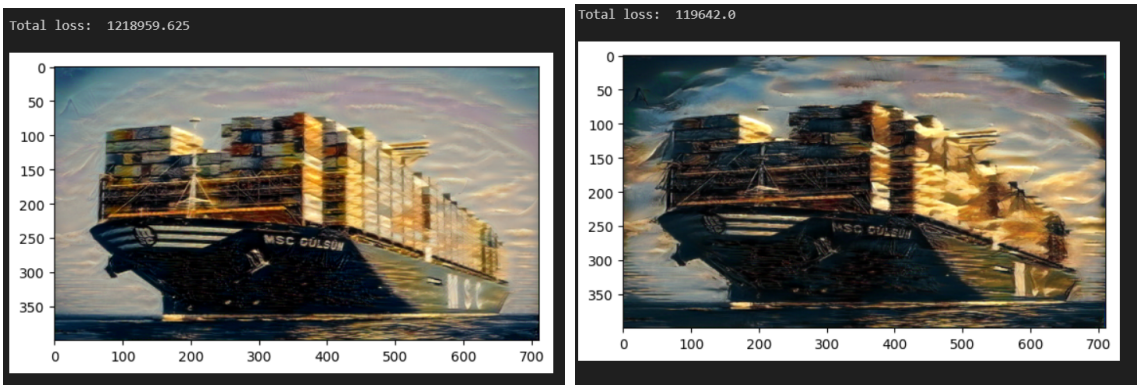

Figure 2: Left: α=1, β=10^6, steps=5000; Right: α=10, β=10^6, steps=5000

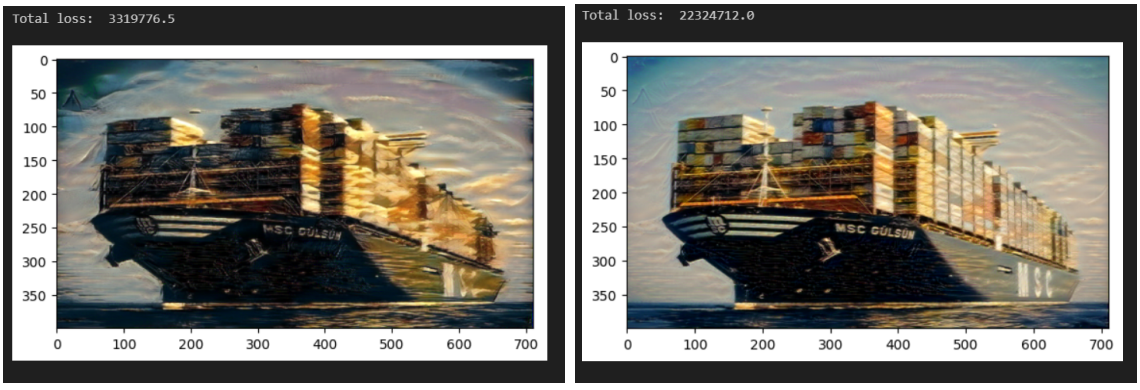

Figure 3: Left: α=1, β=10^7, steps=2000; Right: α=1, β=10^7, steps=3000

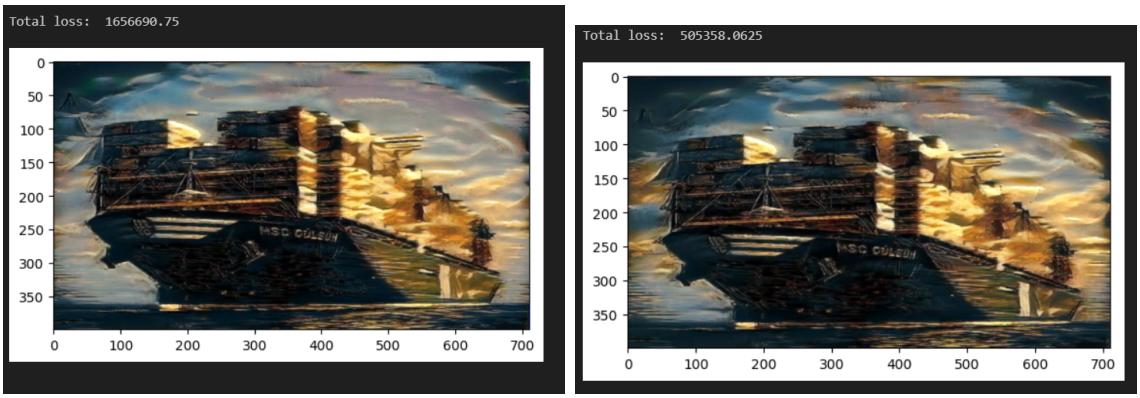

Figure 4: Left: α=1, β=10^7, steps=4000; Right: α=1, β=10^7, steps=5000

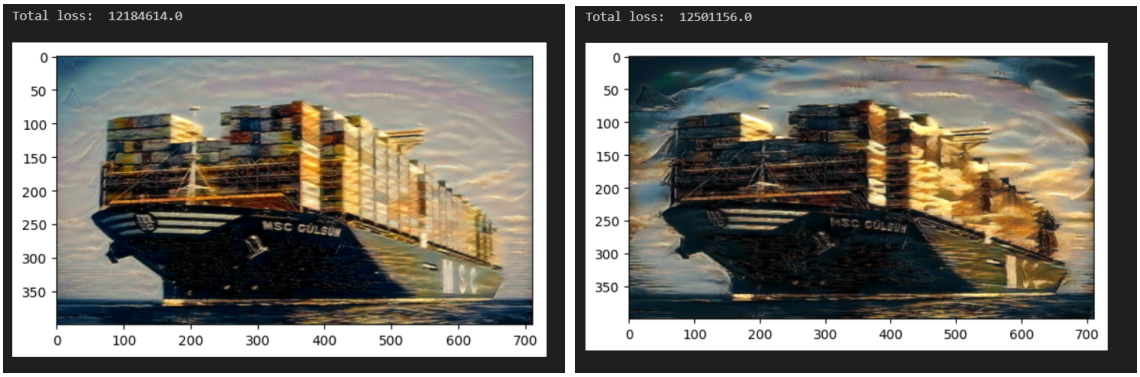

Figure 5: Left: α=10, β=10^7, steps=5000; Right: α=1, β=10^8, steps=5000

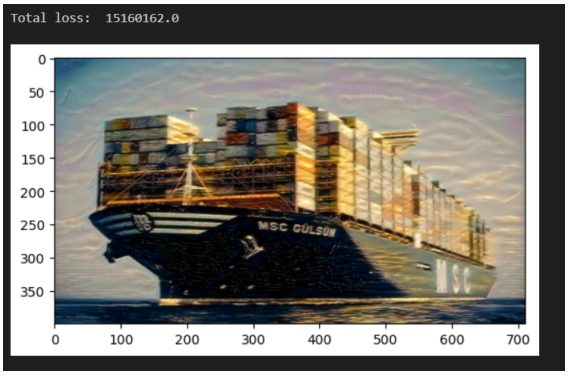

Figure 6: Extreme emphasis on content: α=100, β=10^7, steps=5000

5.3 Observations

Effect of α (Content Weight):

- **Low α (1):** The output focuses more on capturing style patterns, but content structure is more distorted.

- **Medium α (10):** The content of the original image is preserved better. Style appears more subtle and less aggressive.
- **High α (100):** Strong content preservation. Stylization is weak and often flattened.

Effect of β (Style Weight):

- **Low β (1×10^6):** Stylization is subtle. Texture transfer is moderate. Content dominates.
- **High β (1×10^7):** Strong stylization, vivid textures, and patterns from the style image.
- **Very High β (1×10^8):** Overpowers content. Output becomes highly abstract and textured.

Effect of Steps:

- **Low steps (2000):** Output may still be noisy or under-optimized.
- **Medium steps (3000–4000):** Stylization is more defined, with fewer artifacts.
- **High steps (5000):** Converged output with stable textures and structure, especially when hyperparameters are balanced.

6 Conclusion

We successfully implemented neural style transfer using a pre-trained VGG-19 model. Our experiments show that:

- Higher content weight helps preserve structure.
- Higher style weight enhances textural style.
- More steps improve stability but with diminishing returns beyond 5000.
- Extreme values in either direction (very high α or β) can negatively affect balance and visual quality.

Hyperparameter tuning is key to achieving aesthetically pleasing results. The balance between content and style weights significantly influences the output quality.