Style Transfer with VGG19 in PyTorch

Zeba Samiya

Date : 03/26/2025

Course: CSCI 611 Applied Machine Learning

Instructor: Bo-Shen

1. Introduction

This project explores the implementation of **Neural Style Transfer** using the VGG19 convolutional neural network in PyTorch. The objective is to synthesize a new image that preserves the structure and semantic content of one image (content image) while incorporating the artistic style of another image (style image).

The project follows the approach proposed by *Gatys et al., 2016*, in which the output image is iteratively optimized to minimize a weighted combination of content and style losses. The pre-trained VGG19 model is used as a fixed feature extractor to compute activations and Gram matrices from specified layers.

We tested and tuned various hyperparameters, including content and style weights, learning rate, and style layer importance, to generate high-quality stylized outputs. This report documents the experimental setup, optimization strategy, visualizations, and conclusions based on the implementation.

2. Dataset Description and Preprocessing

Content and Style Images:

The content image used was a photograph of a Labrador Retriever standing on a green field. The style image selected was Vassily_Kandinsky composition vii, known for its bold brush strokes and swirling textures.

Preprocessing Steps:

1. **Image Loading**: Images were loaded from a local disk using the Python Imaging Library (PIL).

2. **Resizing**: All images were resized such that their largest side was ≤ 400 pixels to reduce computation.

3. **Normalization**: Images were normalized using ImageNet mean and standard deviation to match VGG19's expected input distribution.

4. **Transformation**: Converted to PyTorch tensors with an added batch dimension.

3. Challenges in Neural Style Transfer

Neural Style Transfer poses several implementation and tuning challenges:

- **Hyperparameter Sensitivity**: The balance between content and style loss greatly affects output quality.
- **Training Stability**: Using too high a learning rate can destabilize optimization and produce noisy results.
- **Style Layer Selection**: Different VGG layers capture different types of style features (from global color to fine texture).
- **Loss Convergence**: Proper tuning is required for the loss to converge meaningfully over many iterations.

4. Experimental Setup and Trials

We conducted iterative optimization trials, experimenting with different content and style weights, learning rates, and layer weights. Below is the configuration of the final best-performing run.

4.1 Trial 1

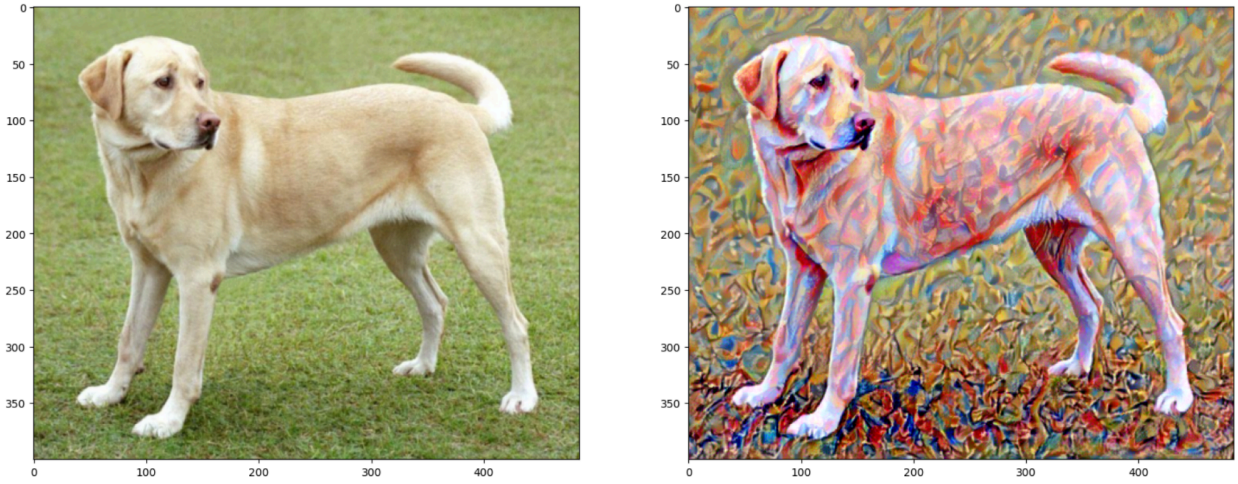
Objective: Establish a baseline stylization result using balanced content and style weights to assess initial quality and convergence behavior.

Key Configuration:

```
style_weights = {
   'conv1_1': 1.0,
   'conv2_1': 0.8,
   'conv3_1': 0.5,
   'conv4_1': 0.3,
   'conv5_1': 0.1
}
```

- **Content Layer**: `conv4_2`
- **Content Weight (α)**: 1
- **Style Weight (β)**: 1e6
- **Learning Rate**: 0.003
- **Iterations**: 2000

- **Optimizer**: Adam
- **Initialization**: Copy of content image



4.2 Final Stylization Trial

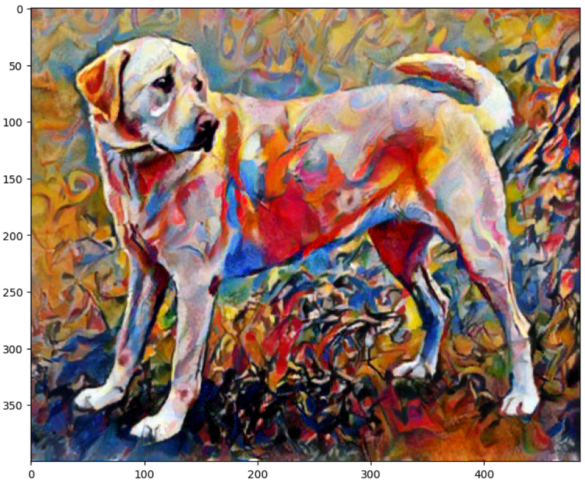
Objective: Create a visually balanced stylized output preserving the dog's form while incorporating Vassily_Kandinsky's signature artistic strokes.

Key Configuration:

Style Layer Weights:

```
style_weights = {
  'conv1_1': 1.0,
  'conv2_1': 0.75,
  'conv3_1': 0.2,
  'conv4_1': 0.2,
  'conv5_1': 0.2
}
```

- **Content Layer**: *conv4_2*
- **Content Weight (α)**: 1
- **Style Weight (β)**: 1e7
- **Learning Rate**: 0.005
- **Iterations**: 4000
- **Optimizer**: Adam
- **Initialization**: Copy of content image



Findings:

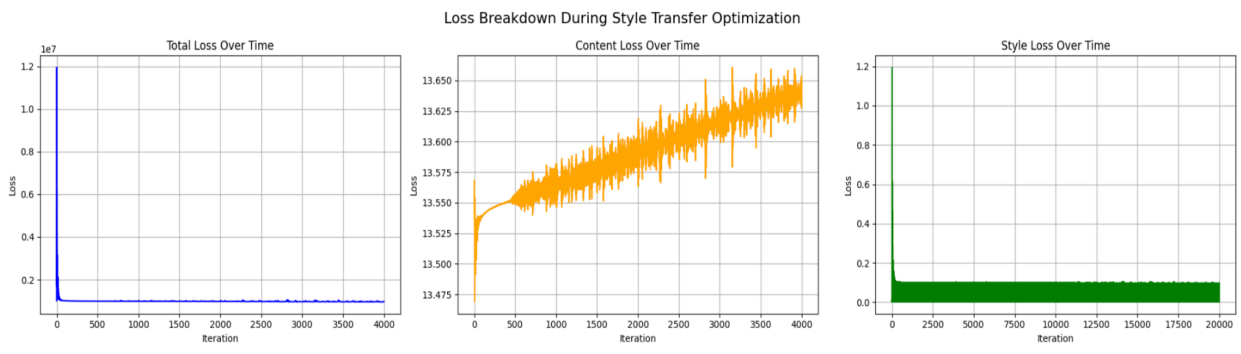
This configuration produced a smooth convergence curve and a final image that effectively blends structure and style. Increasing the weight on *conv1_1 and conv2_1* emphasized larger, swirling patterns from the style image while preserving the dog's form from the content image.

5. Results and Inference

This figure presents individual loss components over the course of training:

- **Total Loss** reflects the combined effect of style and content losses, weighted by their respective coefficients.
- **Content Loss** remains stable, preserving semantic structure.
- **Style Loss** dominates early and decreases gradually as the output adopts artistic features.
-

Tracking each loss separately helped in tuning hyperparameters and verifying convergence behavior.



Side-by-Side Visualization

The stylized output clearly preserves the structure and shape of the dog while successfully incorporating the brushstroke and swirl patterns from the painting.

Content Image: Serves as the structural base of the output. It contains the core layout, shapes, and objects.

Style Image: Provides the artistic texture, color palette, and brushstroke patterns.

Stylized Output: Combines the spatial structure of the content with the visual aesthetics of the style image.

6. Conclusion

This project successfully implemented the neural style transfer algorithm using PyTorch and a pre-trained VGG19 model. By fine-tuning hyperparameters and layer selections, we generated stylized outputs that artistically blend structure and texture.

The experiment demonstrated how different layers contribute to stylistic features, and how content-style weighting influences the output. Our final results aligned well with the original goals, and future improvements could enhance speed and resolution.