INFO8010: Style Transfer

Altaha Yaman,[1] **Verdonck Antoine**,[2] and **Louis Dasnois**[3]

[1] *yaman.altaha@student.uliege.be (s215427)*
[2] *antoine.verdonck@student.uliege.be (s152481)*
[3] *Louis.Dasnois@student.uliege.be (s181779)*

I. INTRODUCTION

In this project we used convolutional neural networks to achieve artistic style transfer on images. The system uses neural representations to separate and recombine the content and style of arbitrary images, providing artistic image creation. Since this project has already been done, several resources helped us in our task such as Neural Algorithm of Artistic Style [1] which gave us a clear approach to solve this problem. We used some pretrained models (VGG19, ResNet50), as well as a VGG19 network that we implemented ourselves.

II. RELATED WORK

Thanks to Convolutional Neural Networks (CNN), deep learning networks achieve some pretty impressive results in the world of computer vision . In fact, CNN is really useful for classification, object detection, or even semantic segmentation. With convolutional neural networks trained on a classification task, we could separate the content and style representations of images and then combine them to create stylized images. We based our work on the original paper for this style transfer method [1], which used a VGG19 neural network. After we failed to reproduce the results using a ResNet50 network, we also quickly reviewed a more recent paper [2] about the same technique applied to more modern CNNs that make use of residual connections to improve gradient flow. It turns out that the residual connections themselves lead to bad results for style transfer, although some mitigations are possible.

III. METHODOLOGY

The following steps are involved:

A. Preprocessing

Preprocessing is done in the way that was used for training the networks.

- Resize the images to a fixed size for further processing.
- Normalize the pixel values for both the content and style images to ensure they are in a similar range.

B. Loss Functions

We define a loss function to quantify the divergence in style and content between two images, using the trained CNN.

- Content loss:

  Calculate the squared difference between the feature representations of the content image and the generated image.

  - Extract the feature maps of the content image and the generated image at a specified layer of the pre-trained CNN.
  - Compute the total squared error between the feature maps to measure the content variation.

- Style loss:

  Capture the style of the style image by comparing the Gram matrices of their feature representations. The Gram matrices encodes the correlations between the activations of the different filters of a layer.

  - Extract the feature maps of the style image and the generated image at multiple layers of the CNN.
  - Calculate the Gram matrices of the feature maps, which represent the correlations between their activations.
  - Compute the total squared error (normalized for the layer sizes) between the Gram matrices to measure the style difference.

- Total loss:

  The total loss is the weighted sum of the style and content losses. The best results were achieved with a style loss of weight 1 and content loss of weight 10^{-2}.

C. Optimize the generated image

- Initialize a random noise image or a copy of the content image as the generated image. Starting with a copy of the content image make the optimization process much faster and reduces the need for the content loss. In some cases, it can almost be removed.

- Define an optimization process to minimize the total loss function. We used the Adam optimizer with default parameters and a learning rate of 0.02, doing 5000 iterations.
- Use backpropagation to compute the gradients of the loss with respect to the generated image.
- Update the generated image by taking a step in the direction of the negative gradients.

  Finally, repeat the optimization process for a certain number of iterations to improve the generated image.

IV. RESULTS

We tested this method using various networks: pretrained VGG19, pretrained ResNet50, and our own VGG19 network both untrained and after training, as it was suggested by [2] that the architecture of the networks is more important as a feature extractor than the weights themselves. We trained on the Caltech101 dataset with only 5 epochs, because the dataset is small we were running out of time. VGG19 being a rather big model, that training was rather ineffective. Results are shown in figures 1-9. We see that the only good result is obtained using the pretrained VGG19. ResNet50 gave quite poor results despite selecting the same layers as in [2]. Finally, our untrained model did not do great, but also did not do too bad considering it required no training. One could say it did better than ResNet.

V. DISCUSSION

The implementation of "A Neural Algorithm of Artistic Style" [1] involves utilizing a pre-trained VGG network to extract content and style representations from images. By minimizing a combined loss function, the algorithm generates visually appealing images that blend content and style characteristics. The paper's approach has shown impressive results and opens up possibilities for various applications in image synthesis and artistic rendering. We can get great results using well trained networks with the right architectures, but if either the architecture changes or the network is not well trained, the results are not great. "Rethinking and Improving the Robustness of Image Style Transfer" [2] suggests ways to make style transfer more effective using networks that have residual connections like ResNet.

[1] Matthias Bethge Leon A. Gatys, Alexander S. Ecker. A neural algorithm of artistic style, 2015.

[2] Nuno Vasconcelos Pei Wang, Yijun Li. Rethinking and improving the robustness of image style transfer, 2021.

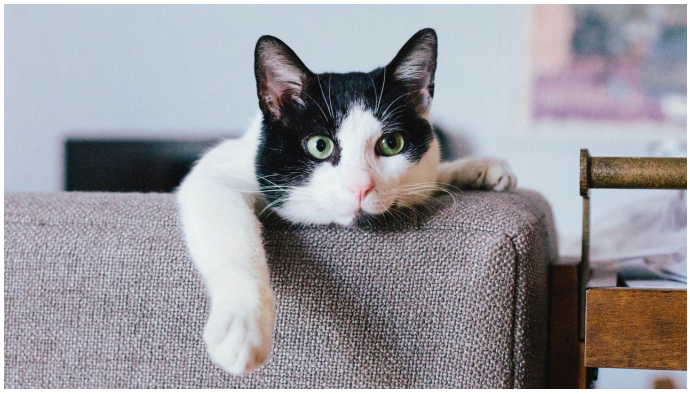

FIG. 1. Original content image

FIG. 2. Original style image

FIG. 3. Style transfer result using pretrained VGG19

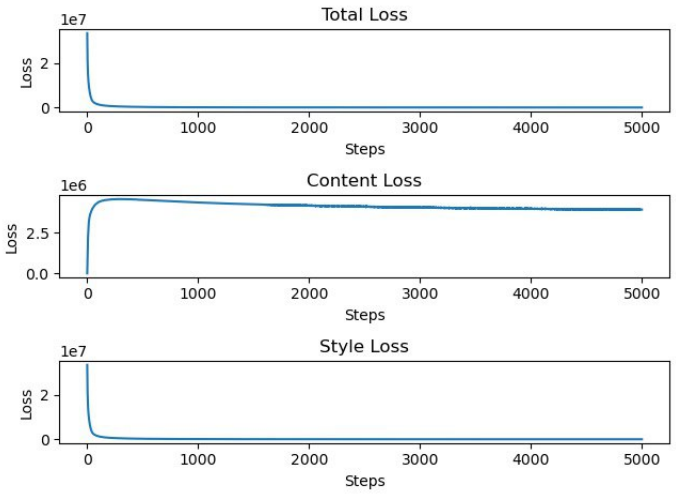

FIG. 4. Loss during optimization of generated image with pretrained VGG19

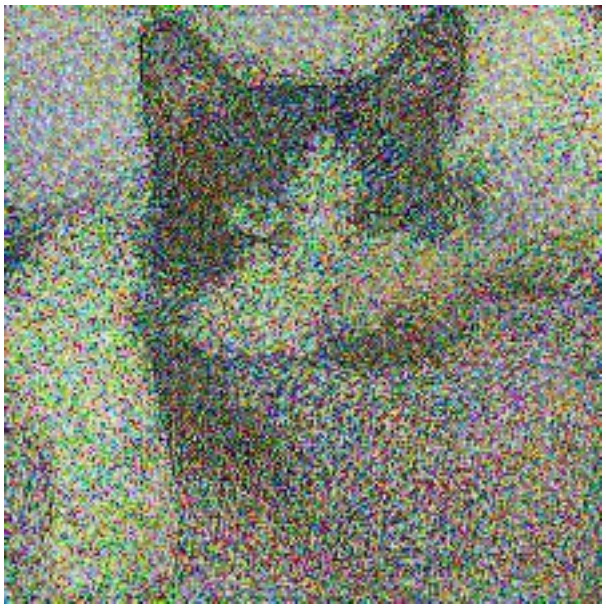

FIG. 5. Style transfer result using pretrained ResNet50

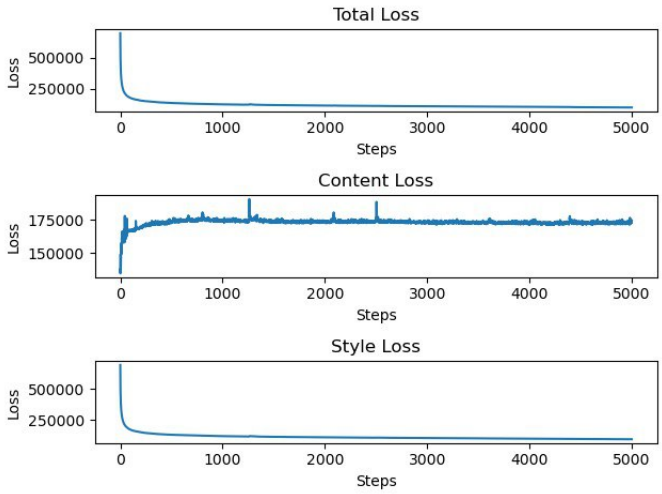

FIG. 6. Loss during optimization of generated image with pretrained ResNet50

FIG. 7. Style transfer result using untrained VGG19

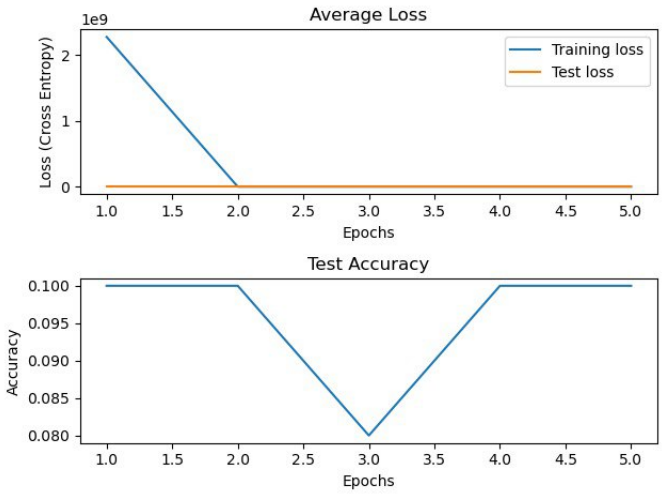

FIG. 8. Loss and test error of our VGG19 network during training on Caltech101

FIG. 9. Style transfer result using our (badly) trained VGG19