Ghiblify

Haruka Masamura, Sasha Floru, Lynda Umuhoza, Ari Codjoe

Introduction

In this project, our objective was to update the artistic style of a given image to resemble that of Studio Ghibli animations. To achieve this, we implemented an existing paper titled "Image Style Transfer Using Convolutional Neural Networks" by Gatys et al.(2016), which introduced a neural algorithm of artistic style that can separate and recombine the image content and style of natural images. The paper's primary objective was to demonstrate how the algorithm can be used to transfer artistic style of famous works of art, such as Van Gogh's paintings, onto arbitrary photographs. We chose this paper as it presented a fascinating idea that also allowed room for creativity in our own implementation. While this paper focuses on Van Gogh's works, we aimed to fine-tune the model for Studio Ghibli images.

Methodology

We used a pre-trained Convolutional Neural Network (CNN) called VGG-19 for feature extraction. To define the content and style representations, we use the intermediate layers of this VGG-19 model, whose early layers capture basic features like edges, textures, and patterns, while the later layers capture more complex and abstract features like object parts. By accessing these intermediate layers, we can describe the content and style of input images.

We represent the style of an image using a gram matrix, which captures the distribution of features across different feature maps. This matrix allows us to extract the style information and apply it to a new image.

To create the new image, we use a feed-forward neural network. The network takes an input image and generates an output image that minimizes the mean square error between the features of the input image and the content target, and the gram matrices of the output image and the style target. We set the style and content target values, then run gradient descent to optimize the output image. The result is a new image that combines the content of the input image with the style of the reference image.

Data

Our model requires only 2 images to train—a content image and a style image. The model's VGG19 component extracts the style from the style image and the content from the content image. The model then leverages these learned features to produce a novel image. Here, we picked three style images, and three content images to feed into the model (see figures below).

Figure 1: The figure above represents content images passed into the model. We focused on landscapes but chose to add a an outlier of a boutique to see how our model performed on different environments. From left to right, we will refer to each image as content image 1 (for the foremost left) and so on.

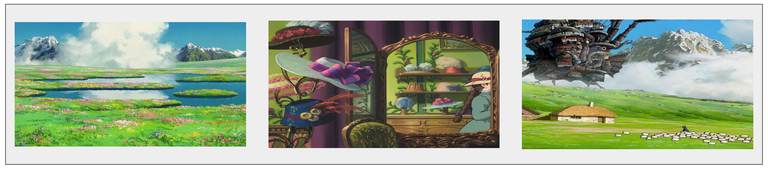

Figure 2: The figure above represents style images passed into the model. From left to right, we will refer to each image as style image 1 (for the foremost left) and so on.

Results

Figure 3: the visualizations of the combinations described in Figure 4. The top left represents Content_1 and Style_1. Top right is Content_2 and Style_2. Bottom left is Content_3 and Style_3. Bottom right is Content_2 and Style_3.

	Loss			
Train Step	**Content_1 + Style_1**	**Content_2 + Style_2**	**Content_3 + Style_3**	**Content_2 + Style_3**
100	153145.16	284699.53	256125.31	604531.75
200	54692.25	198928.13	87166.17	281916.53
300	27894.02	159895.58	39259.41	98415.66
400	17883.50	139571.41	24582.72	47360.80
500	12386.21	127870.67	18267.14	32837.95
600	9257.34	120426.12	14715.22	25584.07
700	7509.90	115344.89	12536.91	21514.36
800	6496.36	111647.96	11044.98	19096.82
900	5798.30	108828.66	9987.55	17479.39
1000	5303.94	106593.88	9198.89	16292.55

Figure 4: This table displays the loss results of our model. As can be seen, combinations that exclusively look at landscape (the first and third columns) produce optimal results compared to others. (see figure 3 for visualizations)

Discussion

The main challenge of this project was becoming familiar with Pytorch and the pretrained VGG19 model. We encountered a bug which caused a substantial drop in accuracy, which was easily observed visually as it was evident that the layers were not effectively identifying image features. The debugging process required an in-depth understanding of Pytorch functions we used. In particular, *create_feature_extractor*—a function we use to extract the intermediate layers of the VGG model—does not perform in-place activations by default, which was causing negative values to linger in the feature representations. With some alterations to our model representation as well as layers used to represent image features, we were able to resolve the bug and achieve visually appealing results which demonstrate successful style transfer onto our desired image.

Despite the successful transfer, our model comes with limitations. While the model is able to successfully transfer image patterns, there are clear limitations with transferring the texture-related information of images with numerous objects and retaining the content image's color information. For our purpose of creating Ghibli style images, a continuation of this project could experiment with training GANs rather than VGG-19.