John Bartlett

IST 652

Project Report – Deep learning & Style

Using deep learning to understand style proved to be a complicated task, mentally and computationally taxing, to say the least. The data in this project comes in a few different forms, images, numpy arrays, and keras objects, but the starting point of the data is just a .jpg. A .jpg doesn’t at face value seem like a dataset, but when you use numpy / keras to process that .jpg into a numpy array it becomes clear that it is indeed a valid type of dataset. The bulk of the images I used in this project were pulled form the Kaggle dataset ‘Best Artworks of All Time’ others were added from external sources mainly google images.

The objective of my project was regretfully transformed as I build out the program. Initially I wanted to train my own feature analysis on a set of abstract art images that I use for my wallpaper I found that it would require far more computer resources then I have access to and a good deal of technical knowledge that I lacked. So, I resided to just trying to replicate, tweak, and understand how to use python to process an images ‘style’ and imprint it on another image, in that respect I was successful.

I will attempt to summarize how an ‘style’ image is processed to understand the style and then imprinted on a ‘base’ image. The program takes 4 parameters, *base image, style image, iterations, and target file.* First, the base image and style image are converted to a format that Keras, and VGG19 modules can read, i.e. a numpy array with a lot of extra details. Keras is an open-source library that provides a Python interface for artificial neural networks, a front end for TensorFlow in python, VGG19 is a 19-layer neural network that can be pretrained with an ImageNet weights file. ***(Output A)***

Once the images are processed, they are concatenated to a list and a pretrained model is loaded into python via VGG19, specific feature and style layers are pulled out of this model to be used on our style and base images. The layers and ‘first draft’ of the result image are then combined to form a single keras ‘loss function.’ Then each of the feature layers loaded from the VGG19 model are combined with the loss function and weighted according to the amount of ‘style loss’ calculated when processing the style image with said layer.

Next comes the generation of the new image, this is where the iterations and computer power issues come into play. To prove that the style being imprinted on the output image closely resembles the style of the ‘style image’ we need to analyze and compare the data between the two. So, a program from the module scipy is used to imprint the style calculated with the loss function to the numpy array generated by the base image. While doing that the ‘style loss’ value and is calculated and the lowest is saved, then this function repeats for a preset number of iterations. At this point I’d like to note that I have a fairly modern 8 core 16 logical processor CPU and this runs with Tensorflow, so some of the work is off loaded to my GPU, however just one iteration takes ~5-10min to run. So this bulky calculation is repeated and gradually improved upon, the array with the lowest loss value is recorded and saved, along the way the loss function is gradually tweaked, which contributes to the improvement of the style. ***(Output B)***

Once a full cycle of however many iterations where run, a numpy array is returned that is mathematically the best representation of the style according to your tweaked loss function. Then the numpy array is saved as an image and plotted via matplotlib alongside the base and style image. Of which the output can be viewed below. ***(Output C)***

Ultimately, I can conclude that yes *elements* of artistic style can be processed and understood using python, that is not to say that the images that are generated could be passed as a specific artist’s style. The chosen style image is critical for a successful style implant, for instance if you chose an post-impressionist artist like Rousseau as your style choice then the only change observed in the result image is a light blurring or paint brush effect over the base image. It really comes down to how ‘style’ is defined, there is a subjective quality to style, aspects of it can come from the collective of an artists entire body of work, its hard to understand style from just one image. There are false flags, like unusual amounts of lines or strong brush strokes.

The images that are generated are unique and interesting, the more abstract a style image is the more dramatic the change is, those are the images that I like the most, and in the sense of trying to abstractify images, I was undoubtedly successful. One caveat that I must mention is that I cannot claim to have written this program from scratch, I am nowhere near that experienced in the various modules used. Majority of my inspiration / influence came from the Kaggle response listed in my sources below, without that I would have spent years attempting this. So in a sense I sort of used this project as an avenue for me to further my python experience and to get my feet wet with image processing and some machine learning, to try and deepen my understanding of that sub-set of data science, in this respect I think I was successful.

**OUTPUT:**

This is the best way I could think of to include output. I have ran this code a lot of times pointing to different images, A and B come almost the same for each iteration of different pictures. In C section I included the ‘final’ output which contains a small sample of images from the output.

**A)**

Model loaded.

Tensor("block5\_conv2/Relu:0", shape=(3, 25, 37, 512), dtype=float32)

Layer Feature for Content Layers :: Tensor("block5\_conv2/Relu:0", shape=(3, 25, 37, 512), dtype=float32)

Base Image Feature :: Tensor("strided\_slice:0", shape=(25, 37, 512), dtype=float32)

Combination Image Feature for Content Layers:: Tensor("strided\_slice\_1:0", shape=(25, 37, 512), dtype=float32)

Layer Feature for Style Layers :: Tensor("block1\_conv1/Relu:0", shape=(3, 400, 602, 64), dtype=float32)

Style Image Feature :: Tensor("strided\_slice\_2:0", shape=(400, 602, 64), dtype=float32)

Combination Image Feature for Style Layers:: Tensor("strided\_slice\_3:0", shape=(400, 602, 64), dtype=float32)

Layer Feature for Style Layers :: Tensor("block2\_conv1/Relu:0", shape=(3, 200, 301, 128), dtype=float32)

Style Image Feature :: Tensor("strided\_slice\_6:0", shape=(200, 301, 128), dtype=float32)

Combination Image Feature for Style Layers:: Tensor("strided\_slice\_7:0", shape=(200, 301, 128), dtype=float32)

Layer Feature for Style Layers :: Tensor("block3\_conv1/Relu:0", shape=(3, 100, 150, 256), dtype=float32)

Style Image Feature :: Tensor("strided\_slice\_10:0", shape=(100, 150, 256), dtype=float32)

Combination Image Feature for Style Layers:: Tensor("strided\_slice\_11:0", shape=(100, 150, 256), dtype=float32)

Layer Feature for Style Layers :: Tensor("block4\_conv1/Relu:0", shape=(3, 50, 75, 512), dtype=float32)

Style Image Feature :: Tensor("strided\_slice\_14:0", shape=(50, 75, 512), dtype=float32)

Combination Image Feature for Style Layers:: Tensor("strided\_slice\_15:0", shape=(50, 75, 512), dtype=float32)

Layer Feature for Style Layers :: Tensor("block5\_conv1/Relu:0", shape=(3, 25, 37, 512), dtype=float32)

Style Image Feature :: Tensor("strided\_slice\_18:0", shape=(25, 37, 512), dtype=float32)

Combination Image Feature for Style Layers:: Tensor("strided\_slice\_19:0", shape=(25, 37, 512), dtype=float32)

**B)**

**Example of an iteration:**

Start of iteration 0

RUNNING THE L-BFGS-B CODE

\* \* \*

Machine precision = 2.220D-16

N = 722400 M = 10

This problem is unconstrained.

At X0 0 variables are exactly at the bounds

At iterate 0 f= 3.04496D+23 |proj g|= 5.85436D+17

At iterate 1 f= 2.91150D+23 |proj g|= 6.88253D+17

At iterate 2 f= 2.09611D+23 |proj g|= 1.01333D+18

At iterate 3 f= 1.29806D+23 |proj g|= 2.54615D+18

At iterate 4 f= 6.59839D+22 |proj g|= 4.02260D+17

At iterate 5 f= 5.65701D+22 |proj g|= 3.24175D+17

At iterate 6 f= 3.62522D+22 |proj g|= 5.11230D+17

At iterate 7 f= 2.58794D+22 |proj g|= 1.63019D+17

At iterate 8 f= 1.91891D+22 |proj g|= 1.37131D+17

At iterate 9 f= 1.59136D+22 |proj g|= 2.93585D+17

At iterate 10 f= 1.18031D+22 |proj g|= 8.68859D+16

At iterate 11 f= 1.04358D+22 |proj g|= 8.38809D+16

At iterate 12 f= 8.21148D+21 |proj g|= 6.54336D+16

At iterate 13 f= 7.48190D+21 |proj g|= 3.77909D+17

At iterate 14 f= 6.04335D+21 |proj g|= 8.43959D+16

At iterate 15 f= 5.86367D+21 |proj g|= 7.19248D+16

At iterate 16 f= 5.47507D+21 |proj g|= 4.34615D+16

\* \* \*

Tit = total number of iterations

Tnf = total number of function evaluations

Tnint = total number of segments explored during Cauchy searches

Skip = number of BFGS updates skipped

Nact = number of active bounds at final generalized Cauchy point

Projg = norm of the final projected gradient

F = final function value

\* \* \*

N Tit Tnf Tnint Skip Nact Projg F

\*\*\*\*\* 16 21 1 0 0 4.346D+16 5.475D+21

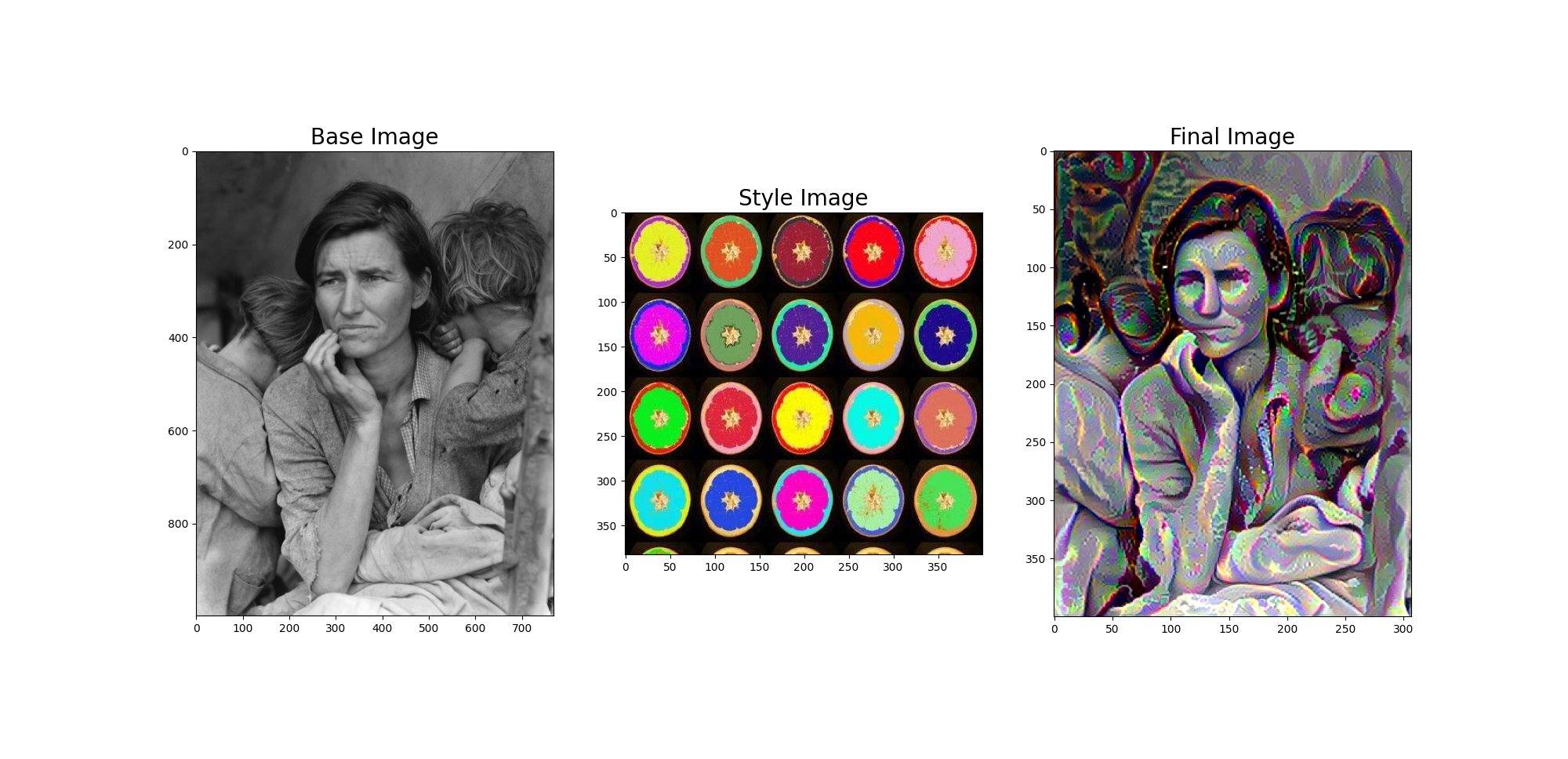
F = 5.4750654734910515E+021

STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT

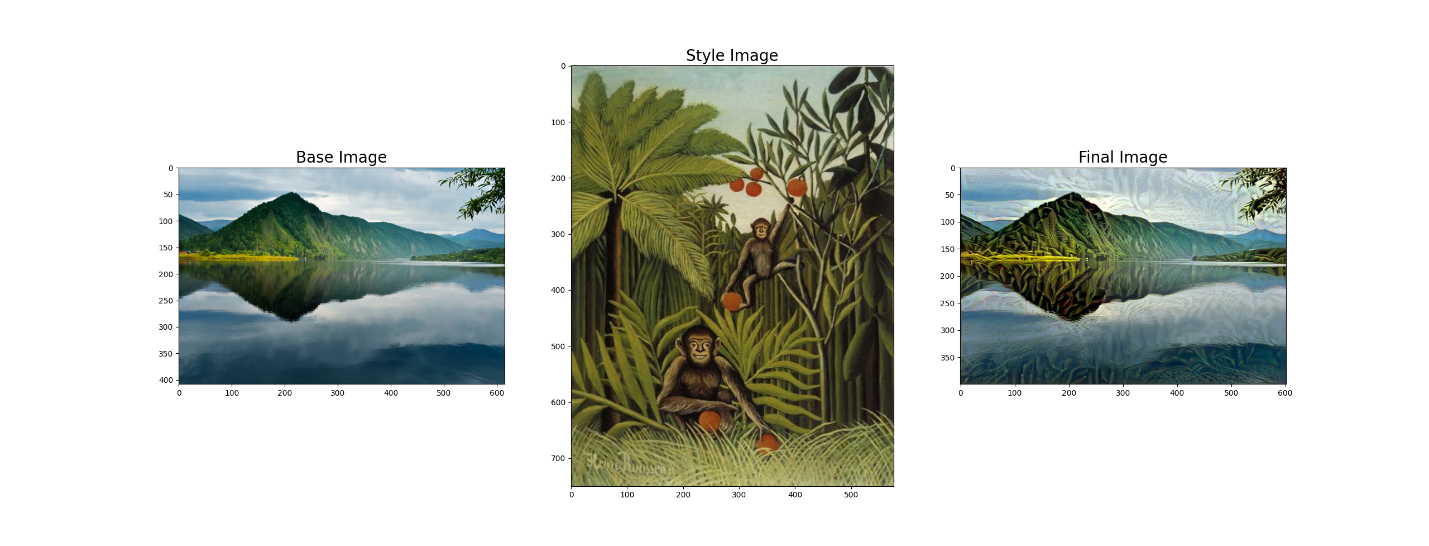
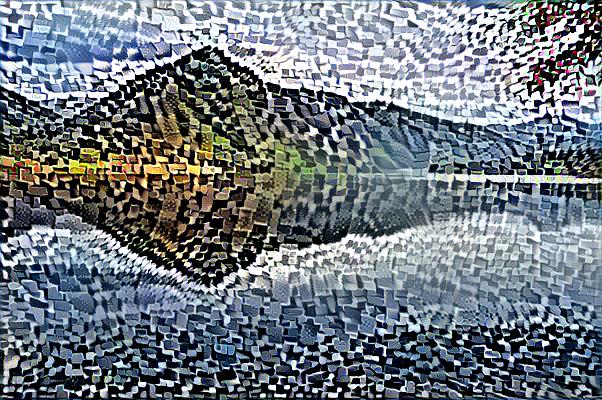
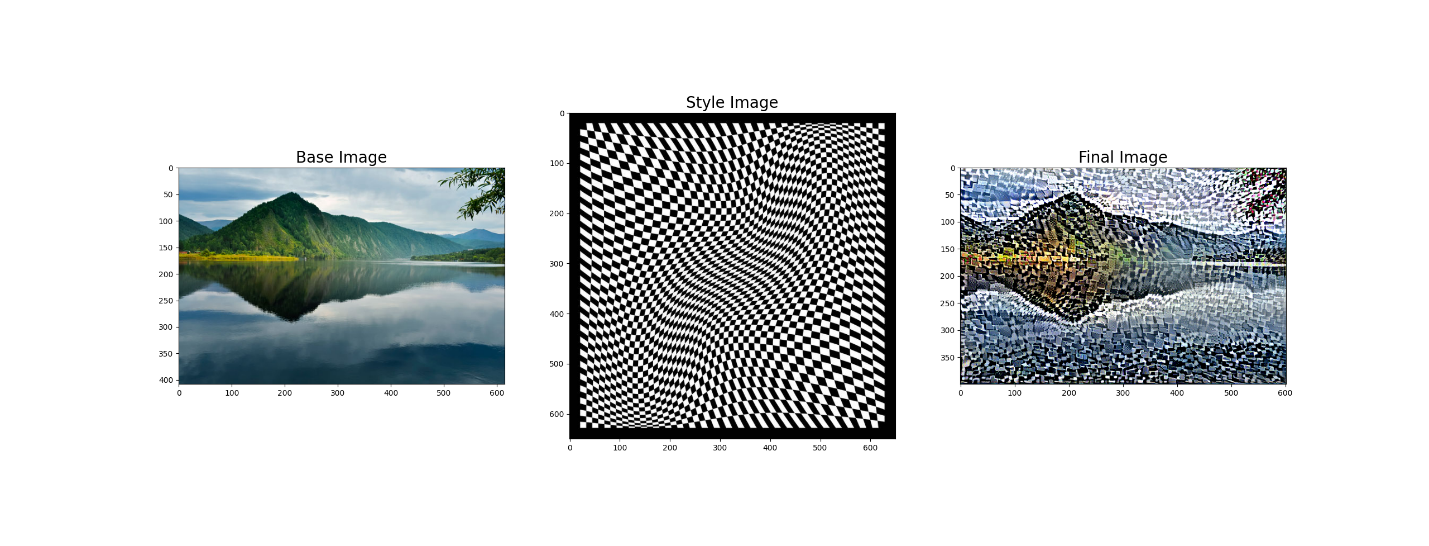
Current loss value: 5.4750655e+21

**C)**

****

**Migrant Mother and a Warhol**



Landscape and Francisco Goya’s “Saturn Devouring His Son”Landscape and a Rousseau****Landscape and an abstract grid.****

**Dataset / Articles / Sources / Templates:**

[**https://www.mathworks.com/help/deeplearning/ref/vgg19.html**](https://www.mathworks.com/help/deeplearning/ref/vgg19.html)

[**https://www.tensorflow.org/tutorials/generative/style\_transfer**](https://www.tensorflow.org/tutorials/generative/style_transfer)

<https://www.kaggle.com/ikarus777/best-artworks-of-all-time>

<https://theconversation.com/using-computers-to-better-understand-art-56887>

<https://www.kaggle.com/basu369victor/style-transfer-deep-learning-algorithm>