Exploring the Algorithm used for Neural Style Transfer

Sarah MacCallum

500684093  
*Department of Computer Science*  
*Ryerson University*Toronto, Canada

*Abstract*— This document describes the process of applying Neural Style Transfer for an image using TensorFlow Core library. The idea is to have the input image be outputted with the style that an artist would have interpreted the image to be. It looks at the values used to accomplish the neural style transfer and manipulates them to see why the values have been chosen.

Keywords— Adam Optimizer, Visual Geometry Group Neural Network, Convolutional Neural Network

# Introduction

This paper on Neural Style Transfer, looks to understand the reason for the certain methods that have been decided on for the process of transferring the style of one image to another. It will explore the use of TensorFlow Core’s Adam Optimizer, which utilizes a stochastic gradient descent, and Visual Geometry Group (VGG) Neural Network as the Convolutional Neural Network (CNN) as the current way to transfer the style of one image to the desired input image. These two activities working together seems to be the most efficient way to accomplish Neural Style Transfer when using a Graphical Processing Unit (GPU) to assist in the process. However, with no access to a GPU, this paper will look at the results of running the algorithm on the input images that utilizes only a Centralized Processing Unit (CPU).

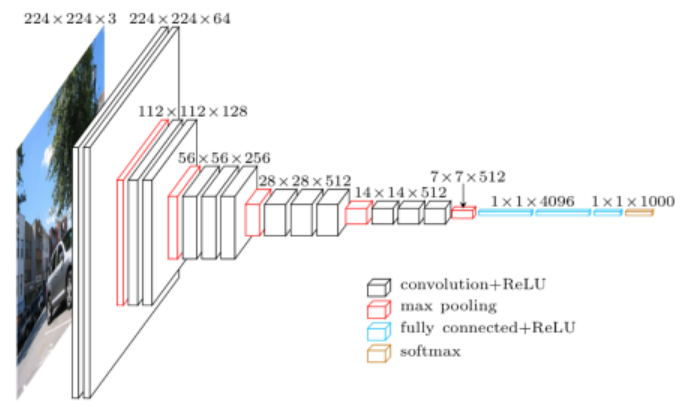
# Background

## Adam Optimizer

The Adam Optimizer is “an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments.” [1] A stochastic gradient descent, is a process of choosing a single data point at random each iteration to be able to compute the gradient. Adam Optimizer is described to be “straightforward to implement, computationally efficient, have little memory requirements, invariant to diagonal rescaling of the gradients, and well suited for problems that are large in terms of data and/or parameters.” [1] By being able to handle large amounts of data it makes the optimizer ideal for handling two images at the same time. The Adam Optimizer takes in a few variables worth noting, learning\_rate, beta\_1, and epsilon. The learning\_rate is a floating-point value used as a bias correction term. Beta\_1 is a float value that is used to help calculate the first moment vector. This vector is the vector that contains the estimate for the mean while the second moment vector is used to hold the uncentered variance of the gradient. Epsilon is a number that is used as a constant for stability purposes. It tends to be a small number too. The program uses the Adam Optimizer created by TensorFlow Core as the implementation of the algorithm.

## Visual Geometry Group Neural Network

The Visual Geometry Group Neural Network is a convolutional neural network model that was put forward by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition.” [2] It takes in an RGB image of 224x244 dimension, as seen in Figure 1, and applies a series of convolutions and ReLUs to achieve the final output.



1. VGG architecture.

VGG also “incorporates 1x1 convolutional layers to make the decision function more non-linear without change the receptive fields” [3] which leads to the ability for the algorithm to contain more layers improving the accuracy of its overall results.

# Implementation

This section will cover the process of the algorithm being used to complete the Neural Style Transfer. The first step is to import both the styled image and the content image and reduce their size to be a maximum of dimension of 512 pixels. Define a VGG and then select layers from the network to represent the content and style of the image. This is done to give us the description of the style and content of the input images. The next step is to calculate the Gramian matrix. The equation for this is:

This equation calculates it by taking the outer product of the feature vector with itself at each location then averaging the outer product at all locations. The easy implementation of this is accomplished through utilizing a TensorFlow core function: tf.linalg.einsum. The next step is to build the model that will return the style tensor and the content tensor, their gram matrix of the layers.

Once the information is returned, the next step is to run the gradient descent by using the mean square error followed the weighted sum of the loss between the content and style.

This is where the Adam Optimizer is used to apply the gradient descent. The Adam Optimizer takes in the input of learning rate, beta\_1 and epsilon. These values are used to help specify the decay rate, beta\_1 and epsilon, and the learning rate specifies the weight update to be completed. GradientTape, which records operations for automatic differentiation, is used to update the image each time the process is called. Ideally the amount of iterations to be run should be one thousand, but most results are received from four hundred iterations.

# Results

The image of the Green Sea Turtle being transformed to have the style of The Great Wave off Kanagawa was executed with both one thousand iterations and four hundred iterations to be able to compare the outputs.



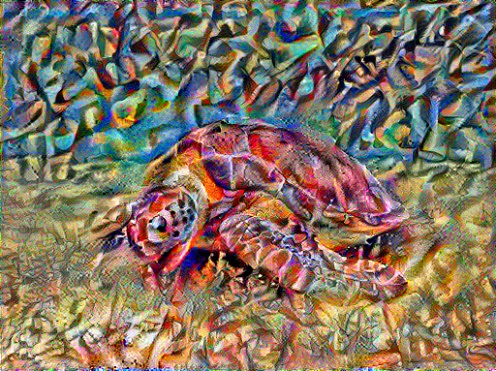


Above are the starting images, input and style, and below are the results. The top image is the output after four hundred iterations followed by the output after one-thousand iterations.





The difference seems to be very subtle between the two, however, the amount of time it took to compute each had a wide difference. The four hundred iteration one took approximately thirty-four minutes to run while the one thousand iterations took approximately one and a half hours. As stated before, this is all only being run on a CPU. These types of programs run at a much faster speed when being able to utilize a GPU as the sources I’ve compared to have theneural style transfer completing in one thousand iterations in under ten seconds. However, due to the constraints of the devices I had access to, the decision was made to complete the remainder of the tests with only four hundred iterations to be able to verify outputs at a faster rate. The one thousand iteration image has slightly more clarity than the four hundred iteration image. To verify that the process was working as expected I ran the program a few more times with different styles and input images.



The green sea turtle was able to have the style of a different picture applied to it successfully. This image had the application of one of Vassily Kandinsky’s painting applied to it.



The image above of myself in graffiti alley, also successful received the style from a different image, Vincent Van Gogh’s Self-Portrait.

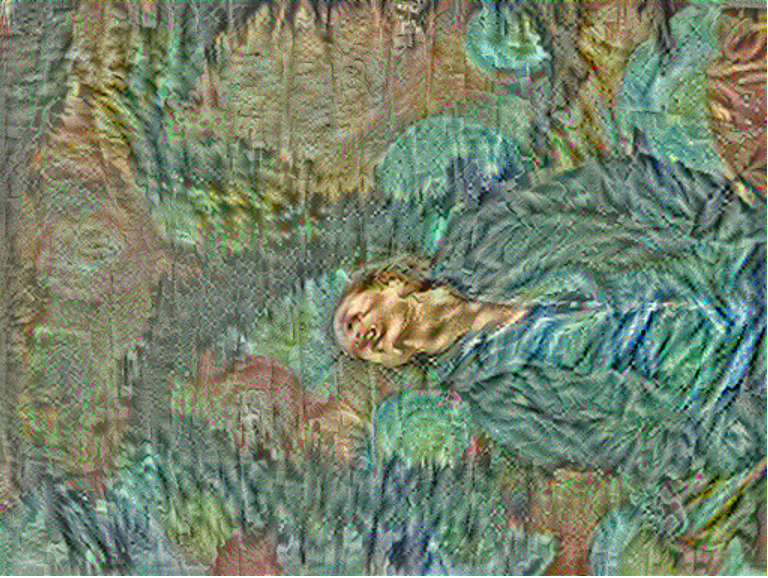
While each image successfully had a style transfer, it is visible that some of the images have lost some of the data. Within each of the images, there is a small amount of, sometimes very intense, red green or blue colours in places where it would not be expected to be found. The style transfer for these pixels, did not work as expected in the end but the over all image has the correct style applied.

Another item to pay attention to was the values given to the Adam Optimizer. For the images above the values were learning rate at 0.02 and beta\_1 at 0.99. To determine why these values were decided upon, they were changed to see what the difference would be after four hundred iterations. The first image had a small value change of learning rate to 0.01 and beta\_1 to 0.8.



The difference between the two doesn’t seem to be that bad, the application of the style transfer isn’t as intense after four hundred iterations, but the image seems to be a bit clearer.

The next change to be made was learning rate from 0.01 to 0.1 and beta\_1 to 0.28. This time the value change was a bit more drastic than before.



The image seems to have more distortion done and the face has become a bit more difficult to tell that it is a face due to the intense swirl on the face. The colours also seem to have become duller in this manipulation. The background seems to have lost some of the style and has a smoother texture than would be expected.

The next manipulation left learning rate at 0.1 and moved beta\_1 back to its original value at 0.99 to see what the affect learning rate had on the optimizer.



In this image the texture is more consistent across the entire image, but the person tends to blend more into the background this time not remaining as clear as one would hope.

The next manipulation moved learning rate back to 0.02 and changed the beta\_1 to 0.01 to see the affect that beta\_1 had alone.



The image kept a stronger hold of the colours in the original image while the style of painting in the self portrait of Vincent Van Gogh has transferred over manipulating the texture of the image.

Next, the values were moved to the default values for the Adam Optimizer. Learning\_rate moved to 0.001, beta\_1 moved to 0.9 and epsilon moved to 1e-07.



The default values at four hundred iterations, has the beginning of the transformation of the painting style applied. The colours remain more based on the original image instead of the style that was to be applied but the texture of the painting strokes has shown up slightly on the picture.

The last manipulation done, was to change only the epsilon value to 1e-02 while learning\_rate and beta\_1 were at the original set values.



This image has a very similar output to the first settings. The image has the texture applied and the colours adjusted well for the style that is being applied to it. Plus, the image is still clear as to what it is supposed to be.

# Conclusion

Overall this algorithm’s application of Neural Style Transfer works well and efficiently. The Visual Geometry Group Neural Network and the Adam Optimizer work well together to accomplish the Neural Style Transfer of images.

The program, while it takes a while to run on a CPU, still gives the desired output of the style transfer, however for faster results it is suggested to utilize a GPU, which can be completed on some computers and some online resources will allow the program to run on a GPU like Google Colab. However, it is useful that the program applies stochastic gradient descent to help lessen the amount of memory being used while running the program.

The values that are chosen by most of learning\_rate at 0.02, beta\_1 at 0.99 and epsilon at 1e-1 is the set of values that appears to transfer the style correctly. The Images with these values have the paint stroke style applied well and the colours are also transformed to match the correct style while not losing sense of what the image is supposed to be. Other values tend to lead to either an unclear image or not a big enough transfer of the style.

This current method is the leading way to accomplish this neural style transfer and based on the results, it is clear as to why it is the case as the accuracy is high for correct style transfer in the results. The program may take a while to run but the output received at the end is what is desired.

##### References

* + - 1. J. Ba and D.P Kingma, “Adam: A method for Stochastic Optimization,” *arXiv.org*, 30-Jan-2017. [Online] Available: https://arxi.org/abs/1412.6980. [Accessed: 13-Dec-2019]
      2. D. Frossard, “VGG in TensorFlow,” *Davi Frossard*, 17-Jun-2016. [Online]. Available: https://www.cs.toronto.edu/~frossard/post/vgg16/. [Accessed: 13-Dec-2019].
      3. J. Wei, “VGG Neural Networks: The Next Step After AlexNet,” *Medium*, 04-Jul-2019. [Online]. Available: https://towardsdatascience.com/vgg-neural-networks-the-next-step-after-alexnet-3f91fa9ffe2c. [Accessed: 13-Dec-2019]