Computer Vision
Assignment-6-7
Neural Style Transfer

Submitted by:
Kashish Shah-AU1841014
Harsh Patel-AU1841015

***Abstract*—This document illustrates the concept of Neural Style Transfer using different models like VGG16 and VGG19. This principle of Neural style transfer takes two images as input, the content image and the style image. The output will be generated in such a way that the content image will gain the attributes(like color, texture etc) of the style image or in other words, the content image will be painted in the style of the style image.**

***Index Terms*—Gram Matrix, Content-loss, Style Loss, Gradient Descent, Optimizers, Content Layers, Style Layers.**

I. Introduction / motivation, and background

Imagine if you can see yourself in the style of a lion or your favourite cartoon character or any other theme you choose. It may be a dream at this point. But not if you know a little about deep learning. Neural style transfer is a technique that showcases the capabilities of a deep neural network.It is an art of giving a particular style to any image, which is illustrated further in this document. Wish you enjoy exploring it.

II. Literature Review of NST

"Art enables us to find ourselves at the same time"[1].Creating an artistic design was initially a human level task but now using Complex Neural Networks we can reach to human-level performance. Neural Style Transfer is best example where any image can be painted to any form of style. NST using deep neural networks has the power to these tranformations.

Leon A Gatys, Alexaner S. Ecker, Matthias developed an exciting Neural Algorithm of Artistic Style in which they used pre-trained weights of VGG-19 Network. There are 2 images, content image (the image which we have to be painted in a particular style) and style image (the style into which we have to convert).They performed gradient descent on a noisy image to find an image that matches the content and style representation of an image. They used Mean-Square-Error and gram matrix for the loss computation and used back propagation for optimising the result.

III. Architectural Overview

A. VGG16

The VGG16 Architecture consists of 13 Convolution layers and 3 dense layers with 5 max pooling layers after each stack. The initial input image size required is $244 \times 224 \times 1$.
There are 5 stacks of convolution layers, each ending with a max pooling 2d layer. The first and second block(or stack) has 2 convolution layers and others have 3 convolution layers. At the end there are three dense layers.
It takes 138 million parameters to train the model, if all layers are used.
Below is the pictorial overview of VGG16:

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
predictions (Dense)	(None, 1000)	4097000

B. VGG19

The VGG19 Architecture consists of 16 Convolution layers and 3 dense layers with 5 max pooling layers after each stack. The initial input image size required is $244 \times 224 \times 1$.
This architecture is very similar to VGG16. One convolution

layer is added in the third, fourth and fifth block in the VGG16 architecture.

Below is the pictorial overview of VGG19:

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv4 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv4 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv4 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
predictions (Dense)	(None, 1000)	4097000

We are going to use some of the layers from VGG16 and VGG19 as content layers and some as style layers.

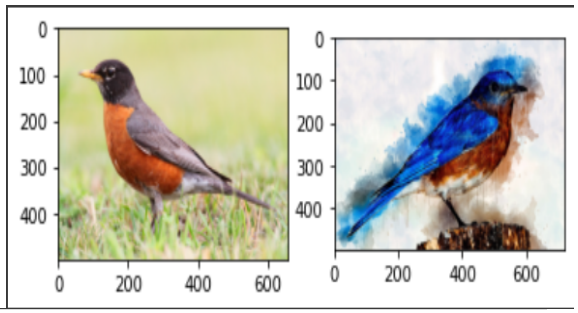
IV. Implementation of Neural Style Transfer

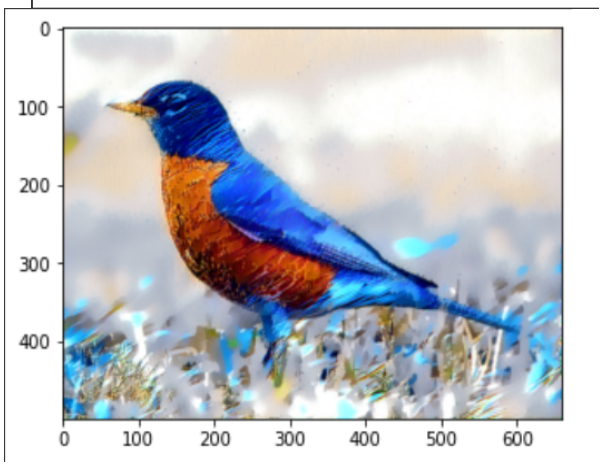
Implementation for VGG16

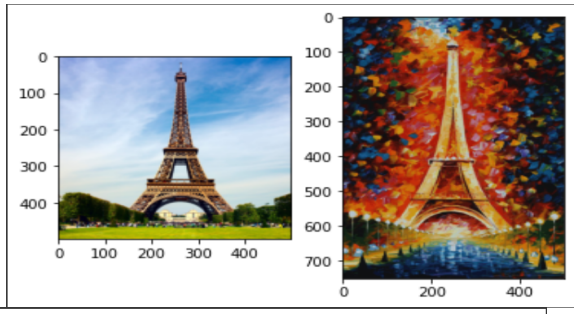
- Initially we took two images as inputs: style and content image.
- We created a model using the pre-trained weights of VGG16.
- We used the layers:block1conv1, block2conv1, block3conv1, block4conv1, block5conv1 as style layers.
- We used the layers:block4conv3 as content layer.
- We pre-processed the content and style images using their corresponding layers.
- We computed loss between the output of content layer and white noise image. This loss is the content loss.
- For the style loss, we computed the feature correlations between the feature maps(output of style layers).
- $TotalLoss = \alpha \times ContentLoss + \beta \times StyleLoss$.
- Then we performed gradient descent on a white noise image to find another image that matches the feature responses of the original image.

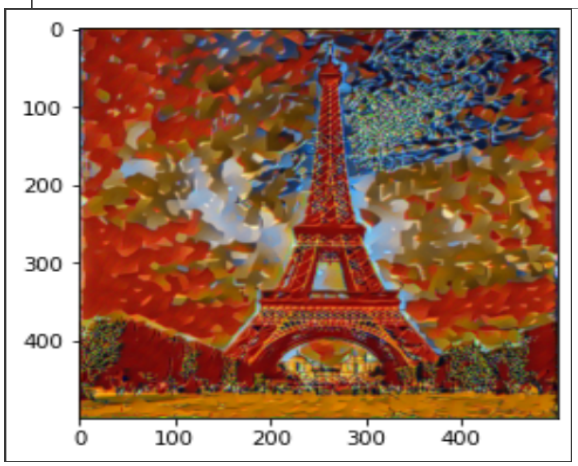
V. Results

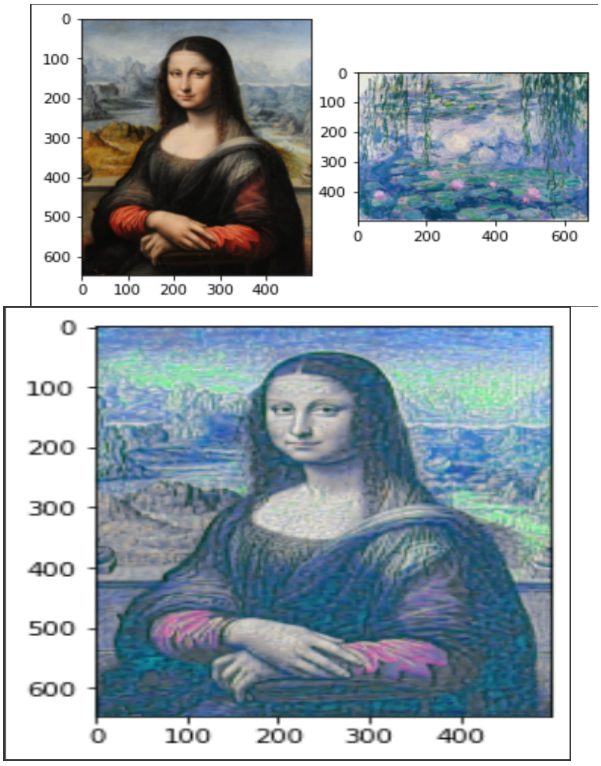
Content Image-left Style Image-right











VI. Conclusion

Using the Neural style transfer, we transformed content image in the style of an another image by minimising the loss function obtained by the combination of content loss and the style loss.

Here, we are able to manipulate the content and style representations to get different outputs and because of that these representations are separable. The content and style representation in the CNN are separable.

References

[1] https://medium.com/artists-and-machine-intelligence/neural-artistic-style-transfer-a-comprehensive-look-f54d8649c199

[2] https://arxiv.org/abs/1508.06576