Neural Style Transfer

1 Team Members

- Pranita Sharma
- Ankith Divakar Raja
- Neeharika Sompalli
- Sandeep Hegde

2 Background and Introduction

An application named **PRISMA** developed in 2016 for mobile users, became an instant craze among millions of users. It was also awarded as the App of the year by both Apple and Google. The motivation behind the project is to know the math behind the application. The application has used Neural Style Transfer to convert the image to an stylized image.

Neural style transfer is an optimization technique used to produce a stylized image by taking three images, a content image, a style image such as an artwork by a famous painter, and the input image you want to style. The stylized image is blended together such that the input image is transformed to look like the content image, but painted in the style of the style image.

Arbitrary Style Transfer in real-time paper[3] proposes new advancement over the previous slow iterative optimization process. Other faster approximations with feed forward neural network have been proposed but cannot adapt to new styles.

3 Method

To accomplish the task of re-imagining an image in the style of other, the project is using multiple models to compare and contrast their performance, computational complexity and network's ability to separate style from the input image.

Two models being used are from the VGG (Visual Geometry Group)[1].

- VGG16: Uses 16 convolutional weight layers
- VGG19: Uses 19 convolutional weight layers

The significance of involving both the models is to study the difference in knowledge penetration and style learning due to different depths in neural network.

The third model is GoogleNet[2].

VGG :

VGG uses convolutional layers of size 3 x 3 and max-pooling layers of size 2 x 2. It has multiple 3 x 3 filters one after the other rather than one large kernel-sized filter. Its great performance is due to the fact that multiple stacked smaller size kernel is better than the one with a larger size kernel because multiple non-linear layers increases the depth of the network which enables it to learn more complex features, and that too at a lower cost for a given receptive field.

The VGG convolutional layers are followed by 3 fully connected layers. The width of the network starts at a small value of 64 and increases by a factor of 2 after every sub-sampling/pooling layer. It achieves the top-5 accuracy of 92.3 percent on ImageNet.

GoogleNet :

GoogleNet[2] devised a module called inception module that approximates a sparse CNN with a normal dense construction(shown in the figure). Since only a small number of neurons are effective, the width/number of the convolutional filters of a particular kernel size is kept small. Also, it uses convolutions of different sizes to capture details at varied scales(5X5, 3X3, 1X1). Another salient point about the model is that it has a so-called bottleneck layer(1X1 convolutions in the figure). It helps in the massive reduction of the computation requirement.

3.1 Flow to achieve style transfer:

There will be two inputs -

- Style Image
- Content Image.

The artwork whose style we want to transfer, known as the style image and the picture that we want to transfer the style onto, known as the content image . The optimization method is to take the advantage of back propagation to minimize two defined loss values. The tensor which will be back propagated into is the stylized image that is the final output, which will be called **pastiche**.

The pastiche is initialized to be content image or random noise. It, along with the content and style images, are then passed through several layers of a network that is pre-trained on image classification (VGG /Google Net).

The outputs of various intermediate layers will be used to compute two types of losses:

- Style Loss :Closeness of the pastiche to the style image in style
- Content Loss: Closeness of the pastiche to the content image in content

Those losses are then minimized by directly changing the pastiche image. By the end of a few iterations, the pastiche image will have the style of the style image and the content of the content image or it will be the stylized version of the original content image.

3.2 Extraction of Features by reading images:

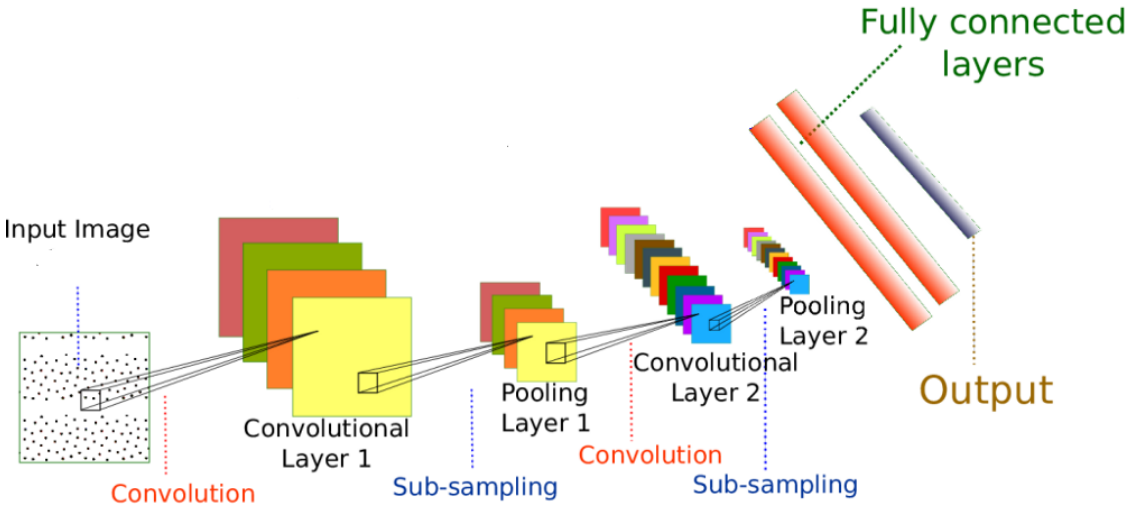
To read an image and output its different features, we used a convolutional neural network (CNN). CNN applies basic principle of convolution. We use the image a N x N filter. Method used is to slide the filter over the image one by one and take the weighted sum of the inputs covered by the filter, transformed by a non-linearity such as sigmoid or ReLU.

Every filter has it's own set of weights, which do not change during the convolution operation. The convolution operation is shown in the below picture, where the green color grid is the feature set resulting from the convolution operation.

So in a Convolution Network, the input image is convoluted with several layers of filters and feature maps are generated and the result is convoluted with new layers. The layers are shown in the below picture.

We have used the following layers in the project:

- Convolution Layer
- ReLU Layer
- Pooling Layer



3.3 Losses:

- **Content Loss:**
  Pass both the pastiche image and the content image through some layers of an image classification network and find the Euclidean distance between the intermediate representations of those images.

  Content Loss equation:

$$L_{content} = \sum_{l} \sum_{i,j} (C^{l}_{i,j} - S^{l}_{i,j})^2 \tag{1}$$

- **Style Loss:** Instead of comparing the raw outputs of the style and pastiche images at various layers, we compare the Gram matrices of the outputs. A Gram matrix results from multiplying a matrix with the transpose of itself:

  Style Loss equation:

$$G^{l}_{i,k} = \sum_{k} F^{l}_{i,k} * F^{l}_{j,k} \tag{2}$$

  It contains non-localized information about the image, such as texture, shapes, and weights - style

$$L_{style} = \sum_{l} \sum_{i,j} (G^{s,l}_{i,j} - G^{p,l}_{i,j})^2 \tag{3}$$

- **Total Loss:**

$$TotalLoss = \alpha * ContentLoss + \beta * StyleLoss \tag{4}$$

  Where α, β represent content weight and style weight. Back propagation will result in minimizing this loss and getting the desired output - a stylized content image.

4 Plan and Experimental

4.1 Plan:

1. Investigate the pre-trained CNN models using style loss and content loss.

2. Find the optimal weights for α, β which produce the best stylized image.
3. Examine the effects on stylized image by using white noise and content image as input image.

4.2 Experimental Set Up:

The following libraries in python are used to conduct experiments:

- Torch
- Torch Vision
- matplotlib
- pillow
- caffe

ARC Cluster GPU nVidia GTX Titan X(12GB) is used to generate the images for faster execution.

5 Results

5.1 VGG 16, VGG 19 and Google Net Inception v1 Results :

We have studied the difference in knowledge penetration and style learning of the VGG16(Visual Geometry Group) , VGG19 and Google Net Inception V1 models. The difference in Style transfer mainly depends on Depth in Neural Network and Density of layers. So the results of VGG16 , VGG19 and Google Net are significantly different.

Layers are changed from 5 to 16 for each model the results are verified.

VGG19 has given best results when the initial pastiche image is taken as content image. VGG19 has multiple stacked smaller size kernels which will help in better quality of style transfer.

(a) VGG 16 Result (b) VGG 19 result (c) Google Net result

Comparing the stylized image of VGG16, VGG19 and GoogleNet for the same weights GoogleNet had the worst quality of style transfer. For further analysis we continued with only VGG16 and VGG19.

5.1.1 Comparison of VGG16 and VGG19 Losses:

Results for VGG16 and VGG19				
Runs	VGG16 Content Loss	VGG16 Style Loss	VGG19 Content Loss	VGG19 Style Loss
50	354.69	26.93	120.15	18.78
100	144.48	25.89	32.59	17.00
150	94.81	24.70	14.52	15.14
200	56.22	23.29	6.80	13.29
250	26.69	21.73	4.10	11.64
300	11.62	19.74	2.95	10.41

Comparing VGG16 and VGG19; VGG19 has lower losses and we only consider VGG19 further on.

(a) Content Image (b) Style Image

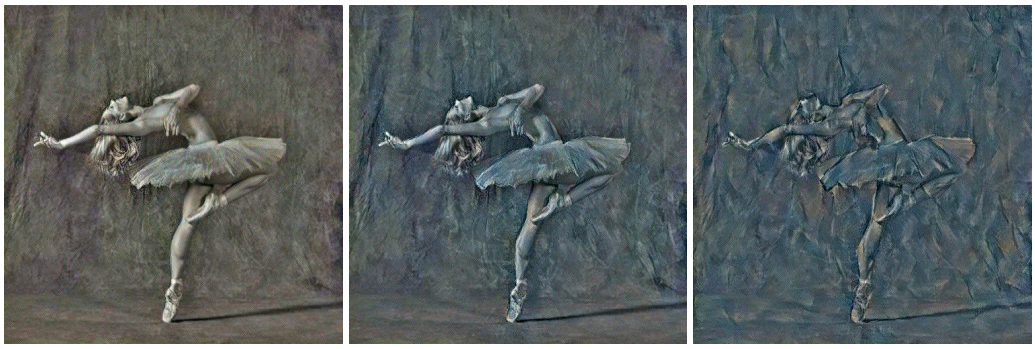
5.2 Weight Optimization:

The content Image and style Image fed into the model are as show above:

Stylized images generated with increasing weight from left to right

5.2.1 Weight optimization for white noise input image:

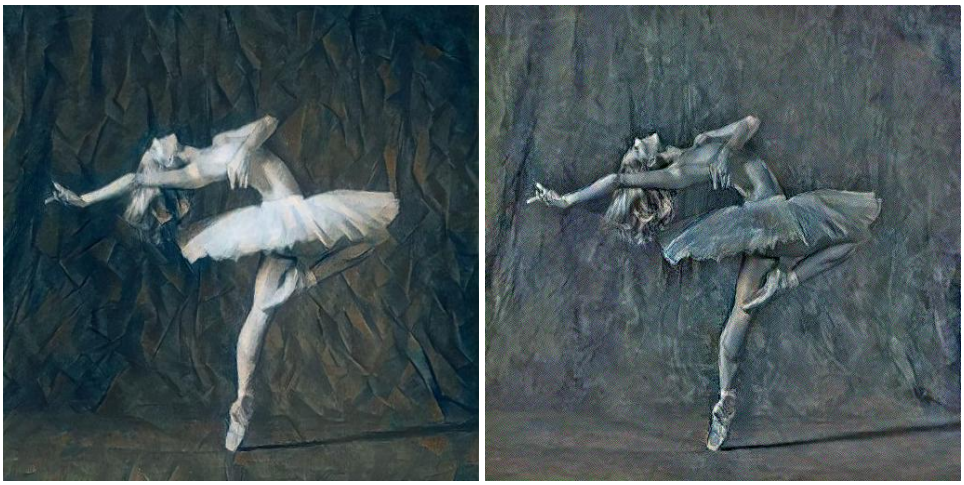
We performed the weight optimization exercise for input image starting for white noise and chose the best one we could identify.



(a) Too much Content (b) Moderate Style and Content (c) Too much Style

5.3 Input Image Selection:

Using the optimized weights from the previous exercise the stylized image generated with content image as input image (a) and white noise as input image (b).



(a) Starting from content image

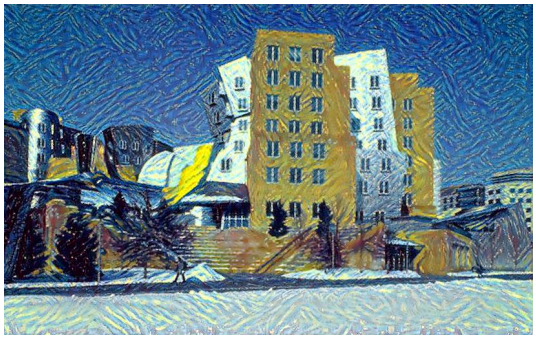
(b) Starting from white noise

5.4 VGG 19 Results:

We ran the final model, with different content and style images using all the optimized parameters. To validate no bias in the optimization's, the same experiments was repeated with different content and style images.

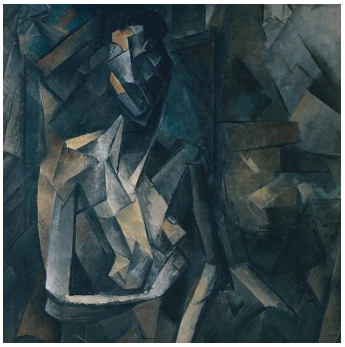
(a) Content image

(b) Style image

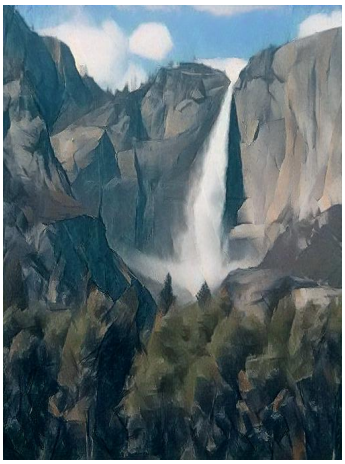


(c) Stylized Image

(a) Content Image



(b) Style Image



(c) result

(a) Content Image

(b) Style Image

(c) result

6 Conclusion

Training neural networks and performing style transfer are two completely different paradigm. In training neural networks we update our weights and biases, but in style transfer, we keep the weights and biases constant, and instead, update our stylized image to reduce the losses.

Google Net's model failed to do great style transfer as it approximates a sparse CNN with a normal dense construction and uses convolution of different kernel sizes. VGG provides great performance due to presence of multiple stacked smaller size kernel than the one with a larger size kernel. From the models VGG16 and VGG19 used; VGG19 gives better results.

7 Future Work

Though it was proved that VGG19 performs better than Google Net Inception V1, Google Net has improved their Inception models to v2 and v3 by iterative improvement over the previous versions which has better speed and accuracy. Our Future work is to explore the style transfer techniques using Google Net Inception V3 and also using other libraries in Python as Open CV. After all optimization's we sometimes feel that the style transfer is either more or less for a particular set of images. So in future work we would like to dynamically adjust weights after the stylized image is generated.

8 GitHub

https://github.ncsu.edu/sdhegde/NeuralStyleTransfer

9 References

1. Karen Simonyan , Andrew Zisserman:Very deep convolutional networks or large-scale image recognition
2. Christian Szegedy, Wei Liu et. al.:Going Deeper with Convolutions
3. Leon A. Gatys, Alexander S. Ecker, Matthias Bethge:A Neural Algorithm of Artistic Style
4. Justin Johnson, Alexandre Alahi, Li Fei-Fei:Perceptual Losses for Real-Time Style Transfer and Super-Resolution
5. Leon A. Gatys, Alexander S. Ecker, Matthias Bethge:Image Style Transfer Using Convolutional Neural Networks
6. Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur:Exploring the structure of a real-time, arbitrary neural artistic stylization network
7. Yanghao Li, Naiyan Wang, Jiaying Liu, Xiaodi Hou:Demystifying Neural Style Transfer
8. https://pytorch.org/tutorials/index.html