Neural Style Transfer For Modeling Artistic Images

Manikanta Chunduru Balaji
Department of Computer Science, University of Southern California, Los Angeles, California 90089, USA
(Dated: May 1, 2021)

In this paper we discuss about the Neural Style transfer. Neural style transfer is an optimization technique that takes in two images an input content image and an input style image and renders an image that mimics the content of the content image and also imbibes the style of the reference style image[1]. The implementation of neural style transfer involves the usage of VGG 19 model pretrained on ImageNet. Each layer of VGG 19 extracts feature information from the images, using this feature information we define content loss and style loss. The Content and Style loss functions are used to determine the difference in content and style with respect to the reference images. By using the loss functions with suitable optimiziters like Adam and SGD we can render visualizations that is a combination of two different images.

Keywords: Neural Style Transfer, Convolution Neural Networks, Content Loss, Style Loss

I. INTRODUCTION

In Art, various artists have created a plethora of visual representations. These representations are so vivid that sometimes we never know the difference between a real picture and the original one.There are few art forms like the abstract form which give a visual representation in a very unique way using a different structure, technique and without using any particular methodology of creating artwork. Its significance is not to depict any visual reality. On the other hand, when we see machine-learning algorithms like deep learning, it has been very successful in applications like image restoration, object detection. Also it has been known from recent researches that deep learning network can be used to create artistic images. In this paper I would like to portray how Convolutional Neural Networks can create artistic images by using the content of one image and the style of another image. This way of rendering images is called neural style transfer.

II. RELATED WORK

In Gaty's paper[2] the authors give a comprehensive explanation about neural style transfer. They have explained how artificial systems is able to create artistic images using deep neural networks. A neural representation of content and style of two different images is separated and combined resulting in artistic images. Their findings indicate that content and style of two images are separable. And these content and style aspects is leveraged to create a new image which preserves content of one image and also uses the style of another image for its color, texture. Convolution neural networks is the neural network utilized for this purpose. Here the network has several small computational units for computing feature information of the input images. This process happens hierarchically over a set of layers where each layer extracts a set of features referenced as feature maps. It is explained that for the content representation, the lower layers capture the pixel information and the higher layers capture the higher level contents of the image. For the style representation, a feature space is utilized over different layers of the network to obtain the texture information of the style image without having the actual content.

In this paper[3] the authors have given a more detailed explanation about Gram matrices. They have given an approach on how minimizing Maximum Mean Discrepancy(MMD) is equivalent to Gram Matrices. They have adopted different distribution alignment methods to explain this. They have examined different transfer methods to understand the style loss and how it results in the formation of different styled images. An effect of balance factor is considered to understand how its impact has on the content loss and style loss of an image. Also, different neural style transfer methods are fused to generate new transfer results.

III. IMPLEMENTATION OF NEURAL STYLE TRANSFER

For the project I have collected a couple of pictures from unsplash repository. I have collected random images for content and style references. Also I have used my painting as a reference for style image.

For my implementation i have used the code snippets from [4] [5]

A. Intermediate layers

1. *Content Representation:*

Neural Algorithms have several computational layers that capture feature maps progressively layer by layer. It can be observed, that when algorithms are trained for detection or classification the filtering layers tend to be more explicit as higher we move towards the final layers[2]. The layers that capture the granular details can be linked to the lower layers of the model. However,

FIG. 1. Content Image 1

FIG. 2. Style Image 1

we can observe that the higher layers capturing the exact information of the content in terms of the number of objects present, structure of the objects and their positions. This content representation can be understood by visualizing the feature maps of the different convolutional layers.

2. Style Representation:

For extracting the style of an image, we use each layer of the network as reference for recreating the style. A space over each layer is created over each convolutional layer. This space is utilized to find the correlations between the different features in the filter maps. Capturing these correlations after each layer in the feature space results in finding the exact texture of the referenced style image. Again, this style extraction can be characterized by visualizing the feature space after each convolutional layer. The colors and complexity of the texture progressively improves and becomes similar to the input referenced style image[2]. This is the style representation of the network.

From the research done on neural style transfer, we know that the content and style of image are separable[2]. When we reconstruct an image that matches the content of one image and the style of another image it is very difficult to replicate both the content and style exactly. So, in order to represent the content of one image and style of another image we use the loss function to control these two components. Controlling the loss function gives us images that gives preference either on content or on style. If content is given more emphasis, an image is synthesized with more significance on content of reference image rather than style of the style reference image. If style is given more emphasis than an image is reconstructed with more similarity to the style of the reference image. A trade off between content and style can be used to create a surfeit of visual representations.

B. Image Pre processing

First, the images are loaded using keras preprocessing function. The image is loaded and resized into rows:400 and columns (width*rows)/height. Here width and height is with respect to base image. Then the image is converted into a tensor for further processing. Here vgg19 pre-process input method is utilized for preprocessing.

C. Model used

For my project I will be using the VGG19 which was invented by Visual Geometry Group[6]. Basically, this neural network consists of 19 layers. The VGG19 is a model with pretrained weights on ImageNet. ImageNet is a dataset of over thousands of images, and the VGG model is pre trained for these images. Using this model trained on object detection, we will extract content and style of two different images to render images that matches the content of one image and style of another image.

1. Model Architecture

VGG 19 model has 16 convolutional layers, 5 max pool layers. There is also one input layer. The architecture of VGG 19 is shown in Fig 3.

D. Defining Methods

Before training the model it is important to understand the different functionalities involved for Neural style transfer. For this purpose we define the loss functions which are utilized in the training process.

Model: "vgg19"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, None, None, 3)]	0
block1_conv1 (Conv2D)	(None, None, None, 64)	1792
block1_conv2 (Conv2D)	(None, None, None, 64)	36928
block1_pool (MaxPooling2D)	(None, None, None, 64)	0
block2_conv1 (Conv2D)	(None, None, None, 128)	73856
block2_conv2 (Conv2D)	(None, None, None, 128)	147584
block2_pool (MaxPooling2D)	(None, None, None, 128)	0
block3_conv1 (Conv2D)	(None, None, None, 256)	295168
block3_conv2 (Conv2D)	(None, None, None, 256)	590080
block3_conv3 (Conv2D)	(None, None, None, 256)	590080
block3_conv4 (Conv2D)	(None, None, None, 256)	590080
block3_pool (MaxPooling2D)	(None, None, None, 256)	0
block4_conv1 (Conv2D)	(None, None, None, 512)	1180160
block4_conv2 (Conv2D)	(None, None, None, 512)	2359808
block4_conv3 (Conv2D)	(None, None, None, 512)	2359808
block4_conv4 (Conv2D)	(None, None, None, 512)	2359808
block4_pool (MaxPooling2D)	(None, None, None, 512)	0
block5_conv1 (Conv2D)	(None, None, None, 512)	2359808
block5_conv2 (Conv2D)	(None, None, None, 512)	2359808
block5_conv3 (Conv2D)	(None, None, None, 512)	2359808
block5_conv4 (Conv2D)	(None, None, None, 512)	2359808
block5_pool (MaxPooling2D)	(None, None, None, 512)	0

Total params: 20,024,384
Trainable params: 20,024,384
Non-trainable params: 0

FIG. 3. VGG 19 Architecture

1. *Content Loss*

To find the content loss we find the mean squared error loss between the original image and the reconstructed image. Content loss basically gives the distance of the content from the original image to reconstructed image. The below formula gives the content loss between the feature map F of our content image and the feature map P of our reconstructed image.

$$L_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F^l{}_{ij} - P^l{}_{ij})^2 [2] \tag{1}$$

2. *Style Loss*

In order to compute style loss we need to understand the concept of gram matrices. For this lets first understand the significance of dot product. Dot product can be viewed as the length of the projected vector a on vector b times the length of the vector b[7]. Its significance can be seen as how similar two vectors actually are. So in the feature space consider any two vectors. Now their dot product gives us the information between them. Higher the dot product the more correlated the features are, lesser the dot product the more different the features will be. When we take the dot product of all feature vectors the result is a gram matrix.

The mean square loss between the gram matrix of the reference style image and reconstructed image gives us the style loss. If there are 'l' number of layers the gram matrix for all layers is computed using the below formula.

$$\mathcal{G}_{ij} = \sum_k F^l{}_{ik} F^l{}_{jk} [2] \tag{2}$$

The contribution of style loss of a layer to the total loss is found between the gram matrices of the style image 'G' and the gram matrices of the reconstructed image 'A'.

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G^l{}_{ij} - A^l{}_{ij})^2 [2] \tag{3}$$

The Total loss is found using:

$$L_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l [2] \tag{4}$$

'wl ' represents the weighting factors of the contribution of each layer and 'El' represents the loss type.

3. *Deprocessing*

The deprocessing method is used to reshape the tensor into a valid image. Then zero centering is done by mean pixel. Zero centering is done so as to linearly transform the data so that data is centered at the origin. Here zero centering is done by subtracting each data point with the mean value. Finally, the image is converted from BGR to RGB format in order easily plot and understand the difference with respect the style and content images.

E. Training Pipeline

For training first we define the content layers and style layers required for neural style transfer. Then we use these layers and pass to a loss function. This loss function extracts the feature information for both content layer and style layers from the activations of the VGG 19. The content loss is determined by the above formula as in Section D. The content loss is calculated with respect to reconstructed image and content image. Similarly the style loss is calculated as per formula given in Section D.

The style loss is calculated with respect to reconstructed image and style image. Finally the total loss is calculated by using the total loss formula given in section D.

Also we compute the gradient using tf.gradient with respect to loss and the reconstructed image. For training purpose we have used SGD and Adam optimizer. Using this optimizer we apply the gradients for about 40 iterations and above.

IV. RESULTS

A. Results obtained using SGD Optimizer

The below figures shows the reconstructed images by using different combinations of content and style layers. The layers used are of the several convolutional layers present in the VGG 19 model.

For the SGD optimizer intial learning rate is set to 100 with decay rate of 0.96.

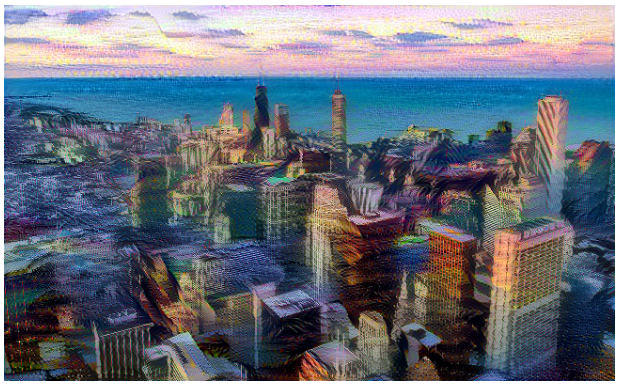

FIG. 4. Iteration 40. For this reconstructed image i have used the style layers block1 conv1, block2 conv1, block3 conv1, block4 conv1, block5 conv1 and content layer block5 conv1 respectively.

FIG. 5. Iteration 40. For this reconstructed image i have used the style layers block1 conv1, block2 conv1, block3 conv1 and content layer block2 conv2 respectively.

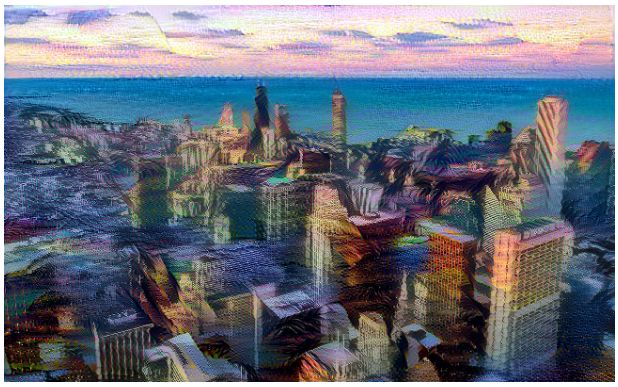

FIG. 6. Iteration 40. For this reconstructed image i have used the style layers block1 conv1, block2 conv1, block3 conv1, block4 conv1, block5 conv1 and content layer block5 conv2 respectively.

FIG. 7. Iteration 40. For this reconstructed image i have used the style layers block1 conv1, block2 conv1, block3 conv1, block4 conv1, block5 conv1 and content layer block3 conv4 respectively.

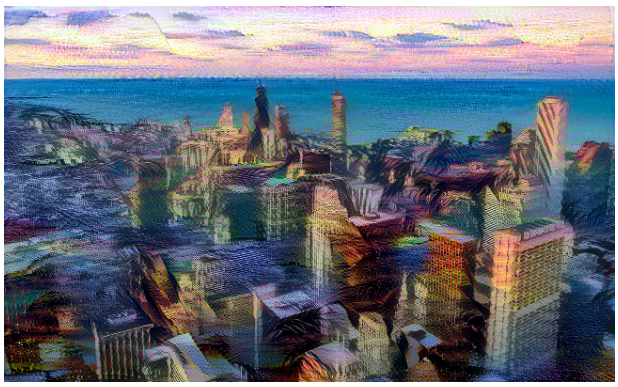

FIG. 8. Iteration 40. For this reconstructed image i have used the style layers block1 conv1, block2 conv1, block3 conv1, block4 conv1 and content layer block5 conv1 respectively.

B. Results obtained using Adam

The below results are obtained using Adam optimizer. Here the learning rate is set to 5. The results only utilize

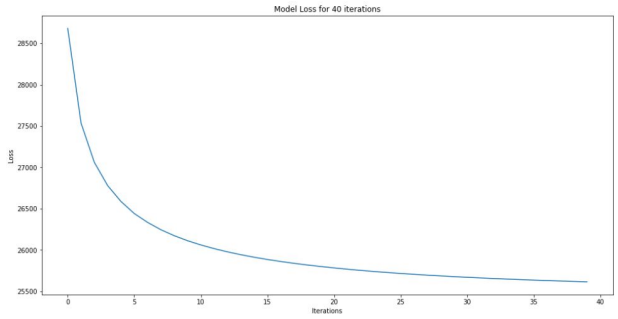

FIG. 9. Loss after 40 iterations for the image reconstructed in Figure 4

the style and content loss.

FIG. 10. For this reconstructed image i have used the style layers block1 conv1, block2 conv1, block3 conv1 and content layer block2 conv2 respectively

FIG. 11. Content Image 2

It is observed from my analysis that the usage of Adam optimizer results in images gives more significance in texture information as show in 13 which is a combination of 11 and style image 12.

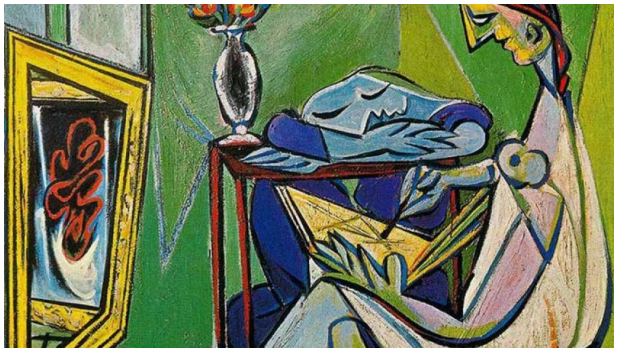

FIG. 12. Style Image 2

FIG. 13. Iteration 40. For this reconstructed image i have used the style layers block1 conv1, block2 conv1, block3 conv1, block4 conv1, block5 conv1 and content layer block5 conv1 respectively.

V. DISCUSSION

In this section we will discuss the results obtained by using SGD and Adam optimizer. From Figure 4 we can observe that by using many style layers the texture information is rendered to the reconstructed image. However in Figure 5 we can see that by usage of first few layer activations for style the texture is not rendered very well to the resultant image.

Similarly, In Figure 4 we can see that by usage of block5 conv1 for content layer the rendered image only has the basic structure but has lost the clarity of pixels. On a similar note in Figure 5 by emphasizing on lower layer for content we can see that the pixel information is rendered well in the reconstructed image.

In Figure 6 we can observe that by emphasizing more on the content by using block5 conv2 the image clarity is lost.

From Figure 7 and 8 we can see how the images vary depending on significance given for style or content. The more the significance given for style the image has more texture similar to the style image. The more the significance given to content of the reference image the clarity of the rendered image is lost.

Comparing results obtained using SGD 5 and using Adam 13 we see that even after using same layers for content and style the results are different. The Adam optimization results in a styled image where clarity is lost but SGD optimization results in styled image preserving good clarity.

VI. CONCLUSIONS

In this paper i have implemented the Neural style transfer. Neural style transfer is a technique of rendering an image that matches the content of the reference content image and the style of the reference style image.

1. Content and Style Representations are the main concepts of neural style transfer.

2. VGG 19 model with pre trained weights is used.

3. The Content loss and Style loss is utilized for rendering the images. Controlling the content and style loss using different layers of the network renders different types of images.

4. Adam and SGD optimizers are utilized in training loop to generate the final images.

5. An emphasis on style renders an image with more texture. An emphasis on content renders an image with more significance on content

6. By using more layers for style loss the texture information is captured to greater extent.

7. By using more content layers the clarity of the image is lost. On the other hand by using the intial layers of the model the clarity is preserved.

From this project I was able to explore the neural style transfer using convolutional neural networks. More research has to be done to understand the importance of layers in transfer learning.

CODE AVAILABILITY

The code is available in the following link `https://github.com/Manikantacb/Neural-Style-Transfer`

DATA AVALILABILITY

The data is available in the following link `https://drive.google.com/drive/folders/1OqX-nqBiWyHtgTOZzKugiZbvfyLxdMEx?usp=sharing`.

ACKNOWLEDGMENTS

I would like to thank my professor, Dr. Marcin Abram for giving the opportunity to work on this project. This project has given me a motivation to learn and has made me explore new concepts. I would like to thank all the instructors and my fellow classmates who have helped me throughout this learning journey.

[1] Neural style transfer (2019).
[2] L. A. Gatys, A. S. Ecker, and M. Bethge, A neural algorithm of artistic style (2015), arXiv:1508.06576 [cs.CV].
[3] Y. Li, N. Wang, J. Liu, and X. Hou, Demystifying neural style transfer (2017), arXiv:1701.01036 [cs.CV].
[4] fchollet, Neural style transfer (2016).
[5] R. Yuan, Neural style transfer: Creating art with deep learning using tf.keras and eager execution (2018).
[6] V. Basu, Style transfer deep learning algorithm (2019).
[7] R. Asokan, Neural networks intuitions: 2. dot product, gram matrix and neural style transfer (2019).