Home Assignment 6-7

Yash Arvindkumar Patel
School of Engineering and Applied Science
Ahmedabad University
Ahmedabad, India
AU1841141

Bhavesh Oza
School of Enginnering and Applied Science
Ahmedabad University
Ahmedabad, India
AU1949003

***Abstract*—In this assignment we tried to carry out the implementation of Neural Style Transfer by understanding the feature Map of the convolutional layer and carry out the combination of style image and content image on the input image and converting it into artistic design.**

***Index Terms*—CNN(Convolution Neural Networks), Neural Style Transfer, Style Loss, Content Loss, Gram matrix**

I. Introduction/ Motivation and Background

This assignment is designed to carry out artistic representation of two images with the help of feature extraction from the black box of convolutional neural network. With the use of transfer learning we can visualize the layer's output i.e. activation map and can use them to extract useful features. In this assignment we have used VGG16 and VGG19 pretrained models to extract the features of given images and by combining them we change the style of the image given the same content of that image. This combination of style and Content image is called Neural Style Transfer.

II. Literature Review

So far Neural Networks have carried out much complex thinkings to simpler level and now it has been expected from Neural nets to bring some artistic exposure from the given images in a simpler manner. For that Leon Gatys in his paper carry out how we can use style of one image and content of other image and by combining them to make the generated image more innovative which have been mentioned in his paper.
"In fine art, especially painting humans have mastered the skills to create unique visual experiences through exploring a complex interplay between the content and style of an image."
From the implementation view point, there were several reference articles in which authors have mentioned how we can use the VGG16 and VGG19 pretrained model to carry out NST. They have also mentioned the important part of codes from scratch with mathematical intuition. Some of them have carried out euclidean distance of pixels at same position to calculate the loss function for content loss while some have used Mean Squared Error to calculate the same. For Style loss they have used the Gram matrix which we will explain in the Implementation part of this report.

III. Architectural overview of VGG16 and VGG19

This section contains the architectural overview of VGG16 and VGG19 pretrained models. How many layers are there and what they are doing to carry out feature maps. Also we would see the number of parameters and the weights have been shared among the Convolutional layer.
There has been a very thin line between both the models as they are very similar and are easier to understand then other model slike inceptionNet because there is a pattern of layers observed in both the models and we will discuss that in depth in below sections.

A. Overview of VGG16

This model has been proposed by K. Simonyan and A. Zisserman from University of Oxford and this model has achieved 92.7 percent accuracy on ImageNet dataset (which has about 14.5 million images and more than 1000 classes) in ILSVCR2014 (ImageNet Large Scale Visual Recognition Challenge-2014).
The important thing about this model is that it has used about 13 convolution layers and 5 max pooling layers in a pattern like 2-3 convolution layer and then one max pooling layer. Also we can point out that there has been no use of large size kernels and the idea is to make model more deeper and not to use higher dimensional kernels so that we can carry out more complex features. Vgg16 model has carry out one improvement from older models like AlexNet by the use of this deep network and lesser dimension of kernel.

- Input dimension of image for vgg16 model is $(224 * 224 * 3)$ i.e. image should be in rgb mode and no need to convert it into grayscale.
- The first and second convolutional layer contains 64 channels and kernel size is of $3 * 3$ and the padding is 'same' as the input so the output dimension should be $(224 * 224 * 64)$ for each layer in first two layers. All the convolution layer will use relu activation function.
- Then there is a max pooling layer of size $(2 * 2)$ and output will be of $(112 * 112 * 64)$
- After that there are two convolutional layers with 128 channels, filter size $(3 * 3)$ and 'same' padding and this results in output dimension of $(112 * 112 * 128)$

- Then there is max pooling of size $(2*2)$ which outputs in $(56*56*128)$ and followed by three convolutional layers of 256 channels with filter size of $(3*3)$ and 'same' padding which outputs in $(56*56*256)$ size of each layer.
- Then again there is a max pooling layer of size $(2*2)$ and output size will be $(28*28*256)$ and then followed by three convolution layers with 512 channels and filter size of $(3*3)$ along with 'same' padding which gives $(28*28*512)$ shape of output feature map for each layer.
- Then again followed by $(2*2)$ max pooling and then three convolutional layers of 512 channels and filter size of $(3*3)$ and 'same' padding which output the feature map of $(14*14*512)$.
- Then comes the last max pooling layer of $(2*2)$ and final output is of shape $(7*7*512)$ and this will be flatten into a single dimension array.
- Then there comes a dense layer of 25088 length and relu activation function followed by two dense layers of 4096 length using relu activation function.
- Then the final layer of model using softmax activation function of 1000 neurons.

Fig. 1. VGG16 model architecture

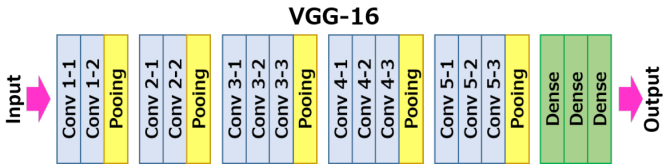
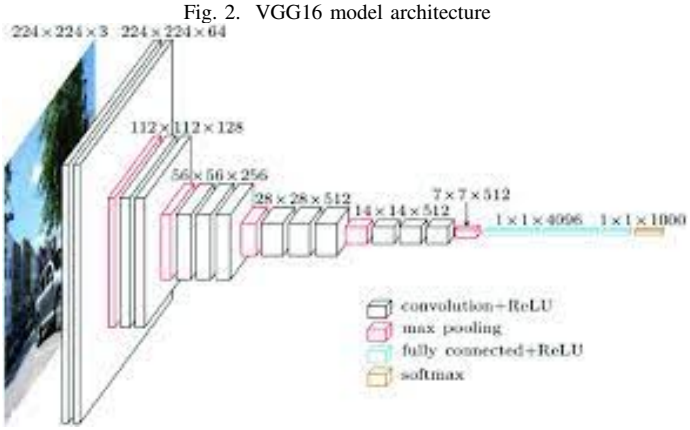


Fig. 2. VGG16 model architecture

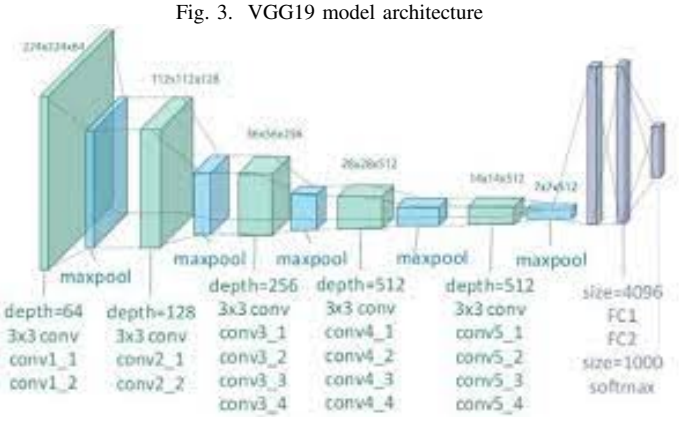


B. Overview of VGG19

Follow on to VGG16, VGG19 has come up with more deeper network. There have 16 convolutional layers, 5 max pooling layers and 3 dense layers of which two have 4096 neurons and last have 1000 neurons.
Architecture of Vgg19 is almost similar to that of Vgg16 the only changes are in the number of convolutional layers in Vgg19 is greater then that of Vgg16 and how that will affect the output will be discussed in the results part.

- Here also the input shape of image should be $(224*224*3)$ that means image should be in rgb form and not in grayscale.
- First two layers are of convolutional layer of 64 channels with filter size $(3*3)$ and 'same' padding which results into $(224*224*64)$ as output shape. All the convolutional layers will use relu activation function.
- After that there is one max pooling layer of $(2*2)$ shape which results into $(112*112*64)$ and followed by two convolutional layers of 128 channels with filter size $(3*3)$ and 'same' padding which results into $(112*112*128)$ as output feature map shape.
- After that there is again one maxpooling layer of $(2*2)$ shape and followed by four convolutional layers with 256 channels and filter size of $(3*3)$ which results in $(56*56*256)$ as final feature map shape.
- Continuing the above sequence of $(2*2)$ maxpooling and four convolutional layers of 512 channels and filter size of $(3*3)$ results into $(28*28*512)$ and followed by the same sequence again of $(2*2)$ max pooling and convolutional layer with 512 channel of filter size $(3*3)$ which results into $(14*14*512)$.
- After the final maxpooling layer of $(2*2)$ which results final feature map of shape $(7*7*512)$ will now be flatten and followed by the two dense layer of 4096 neurons with relu activation function.
- Last layer of vgg19 is same as vgg16 which is of 1000 neurons and softmax activation function.

Fig. 3. VGG19 model architecture



IV. Implementation of Neural Style Transfer

In this section we will dive deep into how the neural transfer is possible and how we can copy style of one image and content other image and merge them to results into some more artistic and innovative image.
The main idea behind the Neural Style Transfer is to calculate the difference between the input generated image and the content/style image to get features from feature map of both the images and generate another image which can have style

of one and content of other.
Now our aim is to get the feature maps when we put our style image and content image in the vggNet model and after getting the feature maps we would like to train our input generated image where can set the feature maps of first layer from each block as similar to style image's feature maps and last layer from last block of content image and input generated image should have minimum distance.
Now to get our needs of minimum distance between the feature maps we need to define *Loss Functions* which shows the distance between the pixels in feature maps of generated-style and generated-content.

- *Loss function for content* is the distance between the feature map of last layer of last block from content image and generated image. Here we take the feature map of last layer from last block because we can get the most complex features of content image from there only. Loss will be defined as:

$$L_{content} = \frac{1}{2} * \sum_{i,j} (A^l_{i,j}(g) - A^l_{i,j}(c))^2$$

  Here l is *layer*, i is the *feature map number*, j is the *position* in it, *A(g)* is the *generated feature map* and *A(c)* is the *content feature map*.
- Now loss function for style is quite different from content loss as this has more feature maps and we need to correlate all that feature map and generate a correlation matrix and then we can use that to get *Style loss* and the matrix is called *Gram Matrix*.
- Now we can look forward to how to calculate this Gram matrix,

$$G^l_{i,j} = \sum_k A^l_{i,k}(I) * A^l_{j,k}(I)$$

  where A(I) is the activation, k is the *position for summation of multiplication of feature maps*, i is the *feature map number* and j is the *position during multiplication*.
- After getting Gram matrix we can look after Style loss,

$$L_{style} = \frac{1}{M^l} * \sum_{i,j}^{l} w^l * (G^l_{i,j}(s) - G^l_{i,j}(g))$$

  Here l is *layer*, i is the *feature map number*, j is the *position* in it, *A(g)* is the *gram matrix of style loss*, *G(g)* is the *gram matrix of generated* and w_α is the *weight*.
- Also there has been weight of Content loss which has notation of w_β and all the loss have sum up and final equation would look like,

$$L_{final} = w_\alpha * L_{content} + w_\beta * L_{style}$$

- After getting loss we can optimize the loss with respect to the input generated image and minimize the loss so that the content of image has similarities with content image and style would have similarity with style image.

V. Results

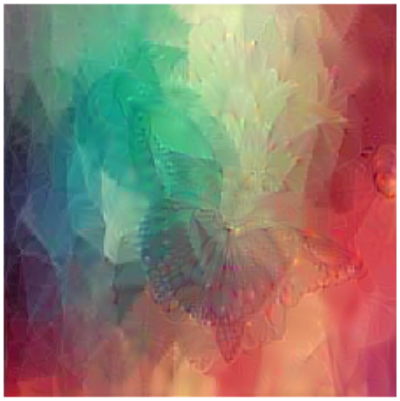
We have carried out the Neural Style Transfer on 2 style and one content image with the help of VGG16 and VGG19 model.

- In this first part we use Butterfly image as the content image and Rainbow Design as style image. In the optimization part we have used SGD optimizer where we declared initial learning rate as 100.0 so that it can start converge faster. Also we have taken epochs as 3000 and the generated images are taken after completion of all epochs.

Fig. 4. Content Image of Butterfly

Fig. 5. Style Image 1

Fig. 6. Generated Image using VGG16



- In this second part we use Butterfly image as the content image and triangle Design as style image and optimizer and epoch number is same as the first example.
- Although results are not as good as we have expected because the loss is pretty higher (around 100) and after 3000

Fig. 7. Generated Image using VGG19

Fig. 8. Content Image of Butterfly

Fig. 9. Style Image 2

epochs it starts repeating the loss and not decreasing, that means the difference of pixels between the feature map of generated input and content image is not collapsing properly.

VI. Discussions and Conclusion

- For the first experiment we can clearly saw the generated outputs having content of butterfly and style of style image.
- Also we can see that generated output for VGG19 is more clearer than VGG16 and the reason is we saw in the architecture of both the models that VGG19 has more deeper layers of convolution and from there we can get more complex features from the content image. For VGG16 there are 13 convolutional layers and for content we take the feature map of last layer of last block and thus the last convolutional layer of VGG19 will be more deeper than VGG16's last convolutional layer.
- The results are not that clearer in second experiment to look different but there are some differences seen on both the generated images. This can be due to model is not able to extract good features from our content image or else the difference between input generated and content has not been minimized properly.

Fig. 10. Generated Image using VGG16

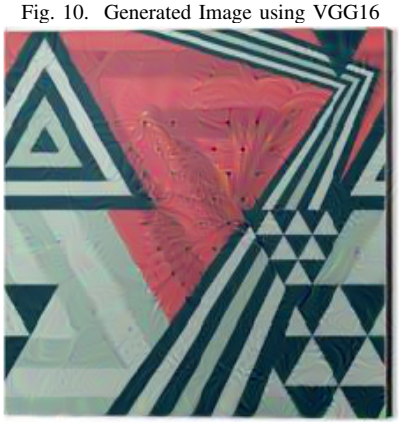
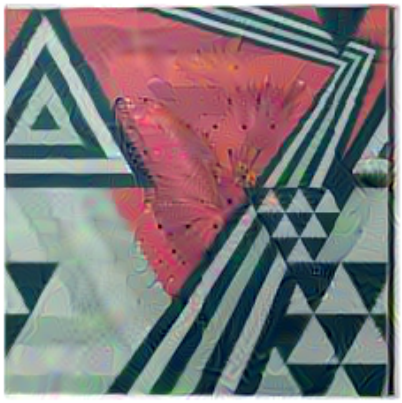


Fig. 11. Generated Image using VGG19



VII. References

- https://keras.io/examples/generative/neural_style_transfer/
- https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text_dataset_from_directory
- https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-neural-style-transf
- https://medium.com/tensorflow/neural-style-transfer-creating-art-with-deep-learning-using-tf-keras-a
- https://github.com/tensorflow/models/blob/master/research/nst_blogpost/4_Neural_Style_Transfer_with_Eager_Execution.ipynb