# Computer Vision Assignment 6-7
# Neural Style Transfer

Hriday Nagrani
*AU1841042*
*hriday.n@ahduni.edu.in*

Panth Patel
*AU1841020*
*panth.p@ahduni.edu.in*

***Abstract*—This is a report for implementation of Neural Style transfer Performed on various content and style images. First we discuss the literature review of neural style transfer and the mathematics behind the loss functions.Then we discuss the architecture overview of different CNN models which were used to performed the style transfer.After that various results and comparisons are showcased which are then followed by the conclusion.**

***Index Terms*—CNN,Deep Learning, Style Transfer, Content loss, Style loss**

## I. Introduction

Converting a normal image into some form of artistic style image has been an area of interest for many decades.

Neural style Transfer is deep learning technique to get an output image having the content of "content image" and style of "style image", so that it feels like out content image has been stylized.

Many apps like DeepArt and Prisma use this techinque. Many artists and designers around the globe use this method to generate new artwork based on old developed Styles.

## II. Literature review of Neural Style Transfer

So as explained in introduction, in neural style transfer we basically derive a image having content of the given content image and style of the style image.

To perform this style transfer we use the fact that the image information is encoded in different layers of the hierarchy in a CNN model.

We define two Loss functions , the Content Loss and the Style Loss. Then the idea is to minimize these loss functions on the output image through SGD and generate a stylised image .

The author of the paper performed gradient descent on a white noise image to produce a style transfered image but we can use content image as base image too and introduce style in it and vice versa.

### *A. Content Loss*

Before defining the content loss, it is important to know how the content of any image can be extracted out of a CNN model.

Basically through the feature visualizations of many activation maps we can observe what layers of the model help generate the basic content(The actual Matter) of the image .

So in laymen terms the content loss will be the error difference of feature outputs of a particular layer for output image(the final image) and content image, and we want to reduce this loss.

The Paper Defines content loss as follows:

"So let $\vec{p}$ and $\vec{x}$ be the original image and the image that is generated and $P^l$ and $F^l$ their respective feature representation in layer l. We then define the squared-error loss between the two feature representations"

$$L_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{ij} (F^l_{ij} - P^l_{ij})^2$$

### *B. Style Loss*

To derive the style loss first we need to understand what exactly we mean by the "Style" of an image in terms of mathematics and understand how we can derive style of an image through a CNN model.

Style in laymen terms means the similar matching patterns across the image which makes the image look stylish and attractive.

We know that to quantify the measure of similarity we can use the operation of correlation .So the style can be defined as "Correlation between activations".

Although notice that here we need to consider correlation across various channels to capture the style .

To measure the correlation of activations across different channels the author defines a gram matrix . Gram matrix

$G^l \in R^{N_l N_l}$, where $G^l_{ij}$ is the inner product between the vectorised feature map in $i$ and $j$ in layer $l$:

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}$$

We can think of this matrix as the "Style matrix". Now to obtain the style of style image onto our output image ,our task would be to reduce the error between the "style(gram) metrices" of these two images.

The paper defines Style loss as follows:

So let $\vec{a}$ and $\vec{x}$ be the original image and the image that is generated and $A^l$ and $G^l$ their respective style representations in layer $l$. The contribution of that layer to the total loss is then

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{ij} (G^l_{ij} - A^l_{ij})^2$$

and the total style loss is

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

Now the total Loss would be just the addition of the style and content losses. Notice we can decide on how much of style or content we want to preserve by passing on a hyper parameter with both the losses. So let $\vec{p}$ be the photograph and $\vec{a}$ be the artwork. The loss function we minimise is

$$L_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha L_{content}(\vec{p}, \vec{x}) + \beta L_{style}(\vec{a}, \vec{x})$$

## III. Architectural overview of VGG16 and VGG19

VGG16 and VGG19 are pre-trained models for image classifications trained on the Imagenet dataset. VGG16 and VGG19 have 16 and 19 layers respectively.

VGG16 has 5 blocks with filter size increasing by a factor of 2 from 64 to 512. First 2 blocks has 2 convolutional layers followed by maxpooling layer and the remaining 3 blocks have 3 convolutinoal layers with maxpooling layer followed by 3 Fully-Connected Layers

For VGG19 first 2 blocks are same as VGG16 and the remaining 3 blocks have 4 Conlvolutional layers each with maxpooling followed by 3 Fully-Connected Layers.

To get content and style representations of images, we look into intermediate layers of pre-trained Models. Intermediate Layers have feature maps that become of higher and higher order as we go deeper into network.
Here we are using the architecture of VGG19 and VGG16 image classification networks. These intermediate layers have the necessary representations of content as well as style of our images. For our input images, we will try and match the corresponding style and content representation at these intermediate layers.

At higher level, taking intermediate layers from a pre-trained image classification network, it must understand the image quiet well. For this it takes raw image as input and build and representation through transformations that turn raw image pixels into a complex understanding of features within the image. Hence, they're really able to capture the invariances and define features within classes. Thus, the model serves as complex feature extractor by accessing intermediate layers.

For extracting style we take 'block1conv1', 'block2conv1', 'block3conv1', 'block4conv1' and 'block5conv1' which are first convolutional layers of block 1 to 5. For content we take 'block5conv2' which is 2nd Convolutional Layer of Block 5. For VGG16 we use style layers as same but change the content layer to 'block4conv2' which is the 2nd Convolutional layer of 4th block.

## IV. Implementation of neural style transfer

1) Loading Content And style Images
2) Deciding the hyper parameters of all the losses , i.e total loss weight, content loss weight, style loss weight.
3) Resizing image into desired dimensions.
4) Pre-processing content,style and combined image
5) Defining content and style layers
6) Defining gram matrix to be used in style loss
7) Defining content loss,style loss and the total loss
8) Reducing Content loss MSE between content and combined images and reducing style loss between style and combined images using stoichastic gradient descent.
9) De-processing the combined image and Saving.

## V. Results

We took 4 styles as shown below and produced the output image on the image of a cat. Each style was used 4 times for constructing the Image with one dimension as 400 and one as 512 on both models VGG16 as well as VGG19.

Fig. 1. Original Image

### A. Style 1

Here we have taken the famous painting of the Great Wave of Kanagawa as our styling.

Fig. 2. The Great Wave of Kanagawa

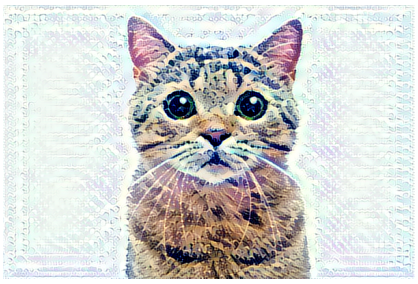

Fig. 3. VGG16 with Dimension 400

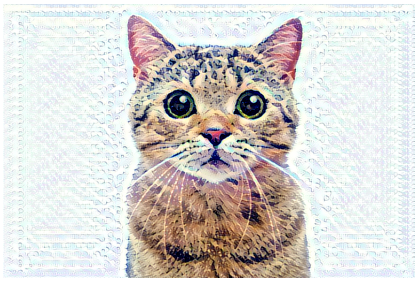

Fig. 4. VGG16 with Dimension 512

Fig. 5. VGG19 with Dimension 400

Fig. 6. VGG19 with Dimension 512

Fig. 7. Color Art

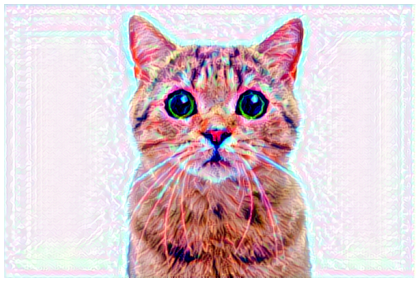

Fig. 8. VGG16 with Dimension 400

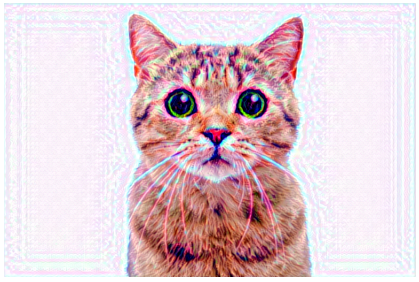

Fig. 9. VGG16 with Dimension 512

### *B. Style 2*

Here we have taken the an abstract color art as our styling. And we can see that the reproduced image is a mixture of colors

### *C. Style 3*

In this we have taken styling as an image of a folded book but the results obtained were interesting.

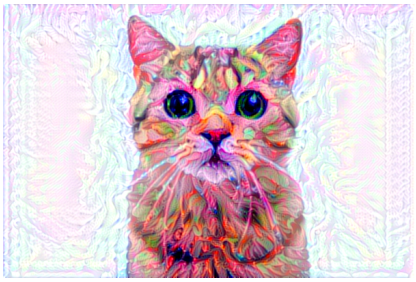

Fig. 10. VGG19 with Dimension 400

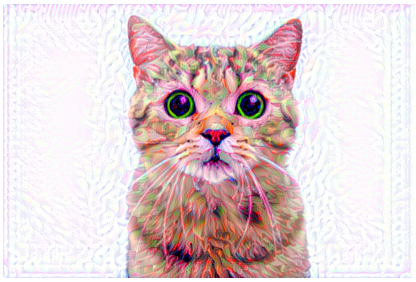

Fig. 11. VGG19 with Dimension 512

Fig. 12. Image of the Book

Fig. 13. VGG16 with Dimension 400

### *D. Style 4*

Here we have used a Picasso Classic as our styling but the results obtained were quiet similar to the input image. But we noticed that the reproduced image has a box like structure in it just like the style image.

Fig. 14. VGG16 with Dimension 512

Fig. 15. VGG19 with Dimension 400

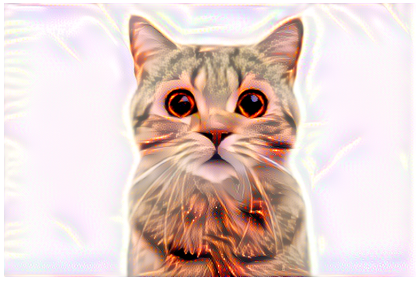

Fig. 16. VGG19 with Dimension 512

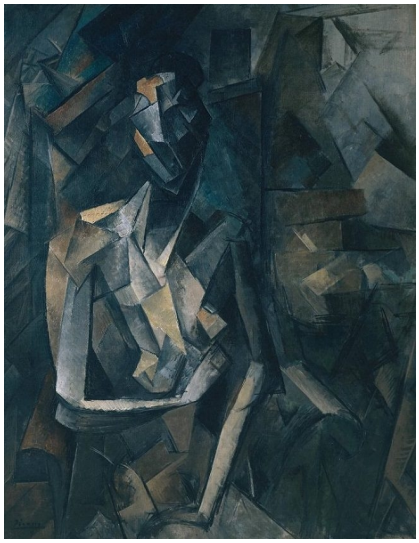

Fig. 17. Pablo Picasso Classic

Fig. 18. VGG16 with Dimension 400

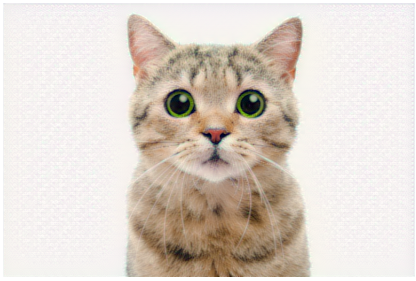

Fig. 19. VGG16 with Dimension 512

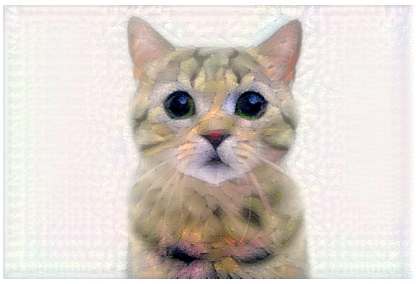

Fig. 20. VGG19 with Dimension 400

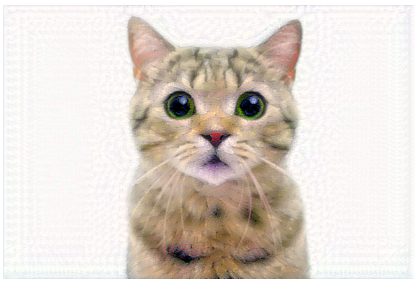

Fig. 21. VGG19 with Dimension 512

## VI. Conclusion

From the above set of results what we can say that output with VGG19 and one of the dimension as 512 pixels comes out to be the best stylized. We use the same layers for extracting style out of the image in both the models but use different layers for extracting content.

In VGG19 the content layer is one block deeper than the content layer in VGG16 and so we can tell that more finer and detailed features are extracted while using VGG19 and hence the image gets stylized better in VGG19.

Talking about the dimension, we maintained the aspect ratio of the image by reducing the height to 400 and 512 in the respective cases and adjusting the width accordingly. We can say that information loss is more in case of resizing one dimension to 400 than 512. And so the stylizing is better in case of reducing one dimension to 512 than 400.

But the catch is that what is better, stylizing image with dimension 400 VGG19 or stylizing image with dimension 512 with VGG16. We can see the above results and conclude that the latter has lesser details but more style due to information loss in the image.

## References


[1] https://medium.com/tensorflow/neural-style-transfer-creating-art-with-deep-learning-using-tf-keras-and-eager-execution-7d541ac31398,"Neural Style Transfer: Creating Art with Deep Learning using tf.keras and eager execution"

[2] https://arxiv.org/abs/1508.06576 , "A Neural Algorithm of Artistic Style"

[3] https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-neural-style-transfer-ef88e46697ee, "Intuitive Guide to Neural Style Transfer"

[4] https://medium.com/codait/art-ai-the-logic-behind-deep-learning-style-transfer-1f59f51441d1, "Art AI: The Logic Behind Deep Learning 'Style Transfer'"