Project Report: Neural Style Transfer

CS725 2021: Foundations of Machine Learning
by

Bodayyagari Govardhan Reddy (213070008) and
Kadam Yash Sachin (213079001)

Course Instructor

Prof. Preethi Jyothi

Indian Institute of Technology, Bombay
Mumbai 400 076

1 Introduction

How would a photograph taken today or a painting done today look like if it were painted by a famous artist of the bygone era? This is the question we would like to solve using neural networks. So we are given two images a style image eg. a painting of a famous painter and a content image eg. a photograph of a famous landmark and we would like to transfer the style from style image onto the content image without modifying the actual objects and other properties of the content image.

1.1 Literature Review

Leon A. Gatys et. al. in their paper [1] have worked on similar problem where their major findings were the representation of style and content in models used for image classification. These models internally develop a representation of the raw input image data. The feature maps of layers closer to the input have the required style representation where they capture the texture and colour of the image while as we go deeper the content representation takes place and the model starts to capture the geometrical shapes in the image. Thus, we are employing transfer learning wherein we are re-purposing a neural network trained to classify images to perform style transfer.[2]

By using these content and style representation we can quantify the content and style loss of the image which will help us in generating a stylized image.

1.2 Scope and Objectives

To transfer the style of a famous artist from one of his/her paintings (style image) to another image (content image) using Neural Networks. This needs to be done based on the style and content weight that we provide. To compare the output of two different pre-trained convolutional neural network models.

2 Datasets and models employed

We are going to use VGG19 and ResNet50 models trained on the ImageNet dataset for this project.

1) **VGG19**: VGG is a deep CNN used to classify images. It consists of 19 layers
   Total params: 143,667,240
   Trainable params: 143,667,240
   Non-trainable params: 0
   It uses kernels of size 3x3.[5]

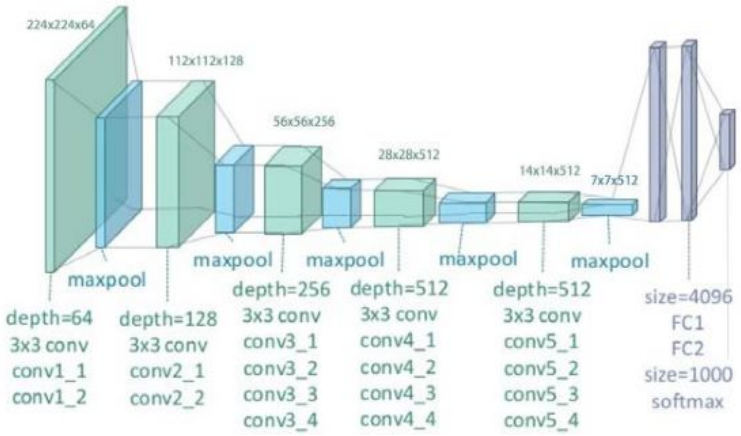


Figure 1: VGG19 architecture

2) **ResNet50**: It has 50 layers and 4 stages:
Total params: 25,636,712
Trainable params: 25,583,592
Non-trainable params: 53,120
It uses kernels of size 3x3 and 1x1.[6]
It contains Identity mapping connection which helps combat the vanishing gradient problem in very deep networks by providing alternative pathway for gradients to backpropagate.

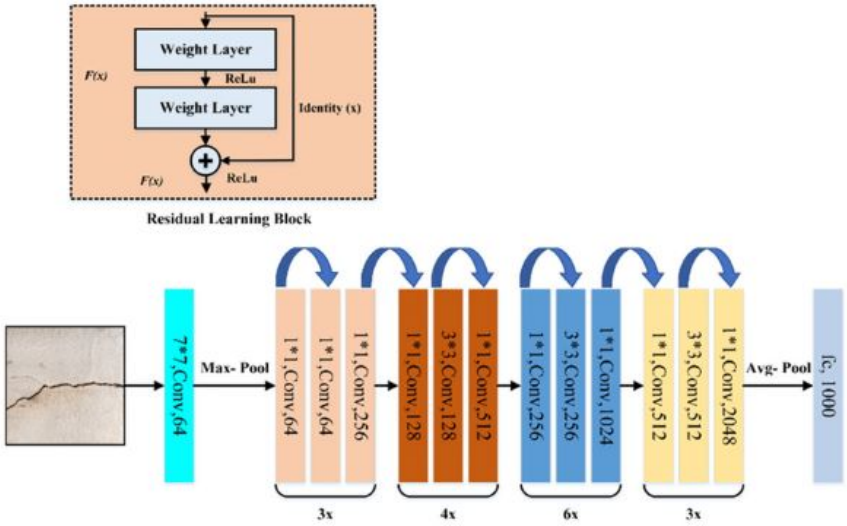


Figure 2: ResNet50 architecture

3 Implementation

Some layers' outputs of the pretrained models are used to quantify the content and style of the image:

- **Content**: We use the deeper layers' output feature maps to represent the content in the image.
- **Style**: By including the feature maps' correlations of multiple layers, we obtain a style representation of the input image, which captures its texture and colour information but not the edges and geometrical shapes. Lower layers are preferred for this task.

The following are the layers selected in this project:

- **VGG19**:
  - Content
    1) block5_conv2
  - Style
    1) block1_conv1,
    2) block2_conv1,
    3) block3_conv1,
    4) block4_conv1 and
    5) block5_conv1
- **ResNet50**:
  - Content
    1) conv5_block3_3_bn
  - Style
    1) conv2_block2_2_bn,
    2) conv3_block1_0_bn,
    3) conv3_block1_0_conv,
    4) conv4_block1_2_bn and
    5) conv5_block1_3_bn

Based on these Content and Style representations we define:

1) **Content loss**:

   We feed the network our input image(x) and content image(p).
   If C_{nn} is the pretrained deep CNN and $C_{nn}(X)$ is the network fed by the input image X. Let ${F_{ij}}^{l}(x) \epsilon C_{nn}(x)$ and ${P_{ij}}^{l}(p) \epsilon C_{nn}(p)$ describe the respective intermediate feature representation of the network with inputs x and p at layer l.
   Then,

$$\mathbf{L_{content}} = \sum_{\mathbf{i,j}} \left(\mathbf{{F_{ij}}^{l}(x)} - \mathbf{{P_{ij}}^{l}(p)}\right)^{\mathbf{2}}$$

2) **Style loss**

We feed the network our input image(x) and style image(a). Then compute the Gram matrices for the two outputs and compare them:

$$\mathbf{E_l} = \frac{\mathbf{1}}{\mathbf{4N_l^2M_l^2}} \sum_{\mathbf{i,j}} (\mathbf{G_{ij}^l} - \mathbf{A_{ij}^l})^\mathbf{2}$$

where, G_{ij}^l and A_{ij}^l are the respective style representation in layer l of x and a. N_l describes the number of feature maps, each of size $M_l = height * width$. Thus, the total style loss across each layer is

$$\mathbf{L_{style}}(\mathbf{a}, \mathbf{x}) = \sum_{\mathbf{l} \in \mathbf{L}} \mathbf{w_l E_l}$$

where we weight the contribution of each layer's loss by some factor w_l. In our case, we weight each layer equally ($w_l = \frac{1}{|L|}$)

3) **Total loss**

$$\mathbf{Total \quad Loss} = \alpha * \mathbf{L_{content}} + \beta * \mathbf{L_{style}}$$

α and β are content and style weights respectively and both are hyperparameters.

We initialize the base input image as the content image itself. This makes the style transfer a lot more quicker than if it were just initialised as a white noise. Then based on the hyperparameters α, β learning rate and number of iterations we go on modifying the base input image using Adam optimizer and gradient descent to reduce the total loss.

4 Results

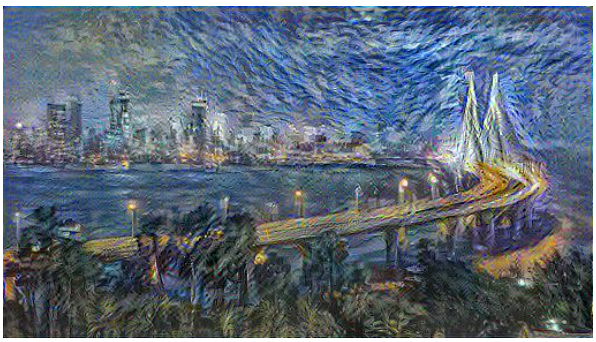
The following results were obtained for the below set of parameters:

- Content image: Mumbai Sealink,
- Style images: a)Starry Night by Van Gogh; b)Krishna(Spring in Kulu) by Nicholas Roerich; c)Pillars of Creation, d)The Scream by Edvard Munch
- number of iterations = 1000, learning rate = 5
- VGG19:
  $\alpha = 10^5$, $\beta = 10^2$
- ResNet50:
  $\alpha = 1$, $\beta = 10^5$

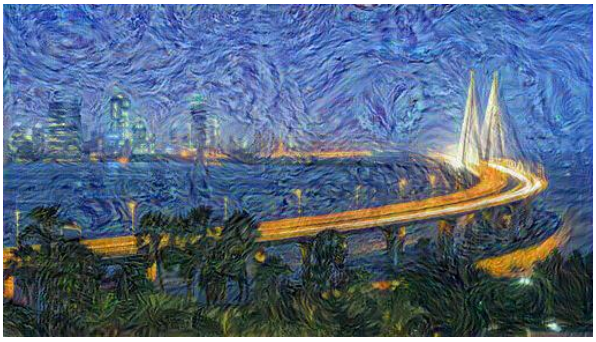
It takes both the models approximately 10 minutes to run the style transfer. As we are resizing the image to be smaller than or equal to 512 pixels. The results obtained for some of the style images are shown in the fig 3, 4 and 5.

(a) Content Image

(b) Style Image



(c) VGG19 Stylized Image



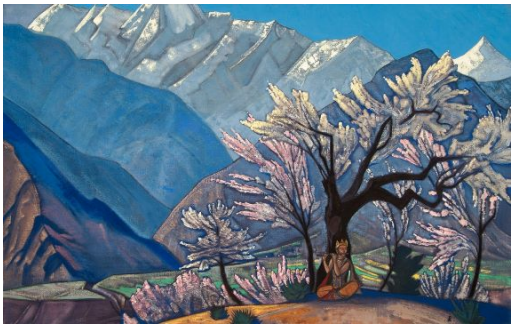
(d) ResNet50 Stylized Image

Figure 3: a)Content image = Mumbai Sealink, b)Style Image = Starry Night Vincent Van Gogh, and the respective stylized images obtained using c)VGG19 and d)ResNet50

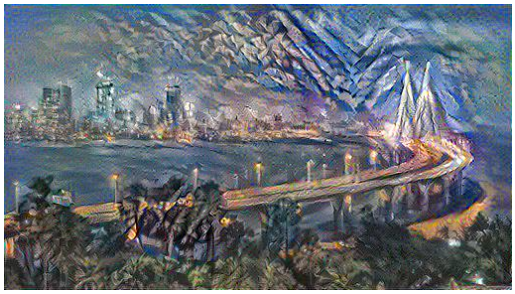
In the fig 3 results obtained in VGG19 is more visually appealing and feels more natural to be labelled as a painting but results obtained from ResNet50 are also satisfactory. More research needs to be done on the style layers used in ResNet50. In addition, since VGG19 is a relatively simpler model (compared with ResNet) the feature maps actually work better for style transfer.

Here in fig 4 as well we see that the image obtained by using VGG19 is more appealing. The batch normalization and the identity mapping in the ResNet model make the stylised images look a bit blurry. But it is quite subjective here.

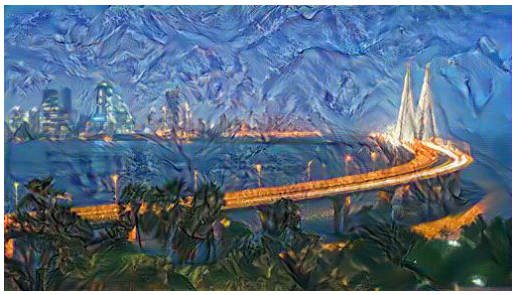
In this fig 5 the difference between VGG19 and ResNet50 is quite clear the style has not transferred to the extent we want to but has made the image a bit blurry and just transferred the simple textures in case of ResNet50. But in the case of VGG19 we see a well stylized image.



(a) Style Image



(b) VGG19 Stylized Image



(c) ResNet50 Stylized Image

Figure 4: a) Style Image = Krishna (Spring in Shimla) by Nicholas Roerich and the VGG19 and ResNet50 Stylized images b) and c) respectively.

(a) Style Image

(b) VGG19 Stylized Image

(c) ResNet50 Stylized Image

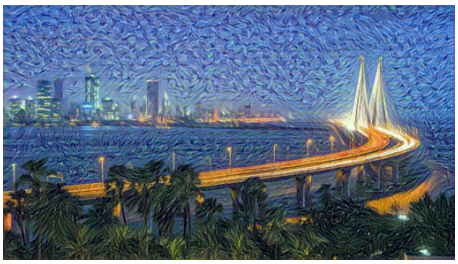
Figure 5: a) Style Image = Pillars of Creation and the VGG19 and ResNet50 Stylized images b) and c) respectively.

Trying different combination of style layers resulted in different amounts of style loss and in some cases the style did not transfer well.

(a) 1 style layer

(f) 1 style layer

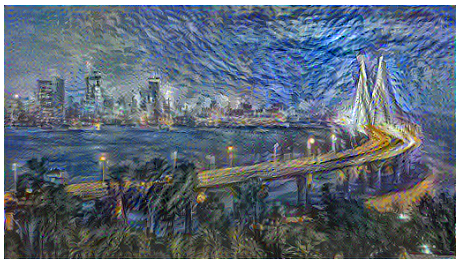
(b) 2 style layers



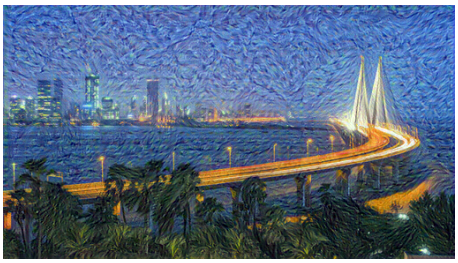
(g) 2 style layers

(c) 23 style layers

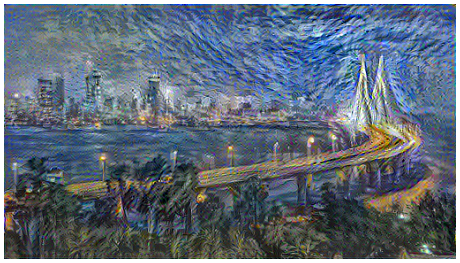
(h) 3 style layers



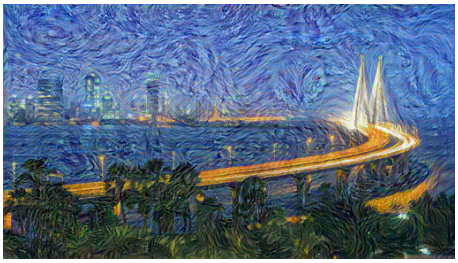
(d) 4 style layers



(i) 4 style layers



(e) 5 style layers



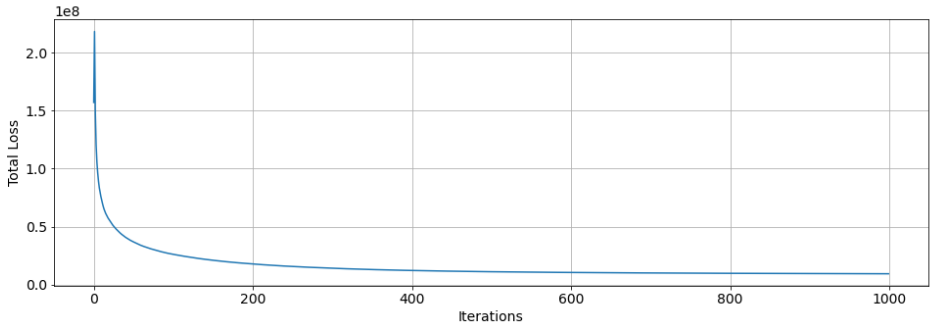
(j) 5 style layers

Figure 6: a, b, c, d and e are for VGG19, and f, g, h, i and j are for ResNet50 and the respective style layers used and the loss obtained are mentioned in the Table 1

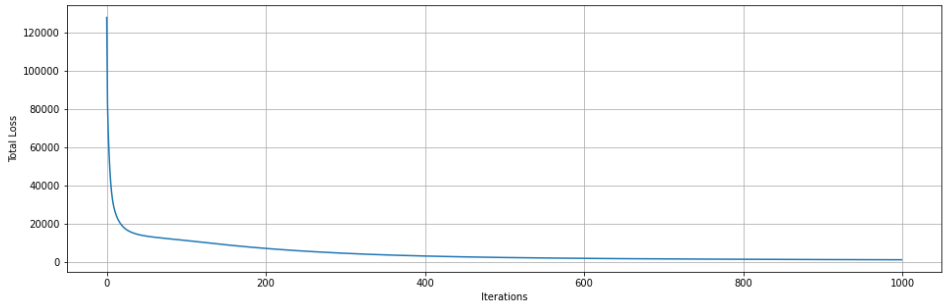
Table 1: Style Layers Chosen and the Style Loss obtained

Style Layers used vs. Total Loss obtained			
Style Layers used (VGG19)	Style Loss (VGG19)	Style Layers used (ResNet50)	Style Loss (ResNet50)
1) block1_conv1	Total loss: $8.83*10^5$, style loss: $1.23 * 10^4$, content loss: $8.71 * 10^5$	1) conv2_block2_2_bn	Total loss: $5.50*10^1$, style loss: $5.33 * 10^1$, content loss: 1.73
1) block1_conv1 2) block2_conv1	Total loss: $3.42*10^5$, style loss: $8.40 * 10^4$, content loss: $2.58 * 10^5$	1) conv2_block2_2_bn 2) conv3_block1_0_bn	Total loss: $2.42*10^2$, style loss: $2.40 * 10^2$, content loss: 2.18
1) block1_conv1 2) block2_conv1 3) block3_conv1	Total loss: $6.13*10^5$, style loss: $3.10 * 10^5$, content loss: $3.03 * 10^5$	1) conv2_block2_2_bn 2) conv3_block1_0_bn 3) conv3_block1_0_conv	Total loss: $2.47*10^2$, style loss: $2.45 * 10^2$, content loss: 2.16
1) block1_conv1 2) block2_conv1 3) block3_conv1 4) block4_conv1	Total loss: $1.02*10^7$, style loss: $3.92 * 10^6$, content loss: $6.24 * 10^6$	1) conv2_block2_2_bn 2) conv3_block1_0_bn 3) conv3_block1_0_conv 4) conv4_block1_2_bn	Total loss: $3.30*10^2$, style loss: $3.28 * 10^2$, content loss: 2.26
1) block1_conv1 2) block2_conv1 3) block3_conv1 4) block4_conv1 5) block5_conv1	Total loss: $9.34*10^6$, style loss: $3.59 * 10^6$, content loss: $5.74 * 10^6$	1) conv2_block2_2_bn 2) conv3_block1_0_bn 3) conv3_block1_0_conv 4) conv4_block1_2_bn 5) conv5_block1_3_bn	Total loss: $1.15*10^3$, style loss: $1.11 * 10^3$, content loss: 2.98

We see that as we add more and more style layers the style loss goes on increasing but in the fig 6 we see the corresponding stylized images quality also increases. Thus, using more style layers improves the output quality although it increases the loss. The different combinations of style layers need to be explored in the future.
In the fig 7 we see the loss decreases to very low value at a very small number of iterations but the image obtained at these lower iterations is not stylized well.



(a) VGG19 Loss Vs Iterations



(b) ResNet50 Loss Vs Iterations

Figure 7: Total Loss Vs number of Iterations

5 Conclusions

The results obtained by using VGG19 were more appealing to us than ResNet50 but this is quite subjective. The choice of hyperparameters such as content weight and style weight heavily influenced the final image. While the total loss decreased to a low value at fairly low number of iterations a more refined image was obtained at a significantly higher number of iterations. Different combinations of content and style layers were tried but the ones mentioned in the slides gave the best results.

6 Future Scope

Additional CNN models such as Inception, EfficientNet, etc. can also be tried. The models used currently apply style transfer to the entire image, in certain applications we might want to apply to a single object in the image a Generative Adverserial Network can be used to achieve this. Also the time taken to generate a single image is quite high in this method so it cannot be used on a video. Other techniques need to be explored for video files.

References

[1] A Neural Algorithm of Artistic Style, Leon A. Gatys and Alexander S. Ecker and Matthias Bethge.

[2] 4 Pre-Trained CNN Models to Use for Computer Vision with Transfer Learning, Orhan G. Yalçın, https://towardsdatascience.com/4-pre-trained-cnn-models-to-use-for-computer-vision-with-transfer-learning-885cb1b2dfc

[3] Neural Style transfer with eager execution, Nikhil Agrawal https://github.com/nikhilagrawal2000/Neural-Style-Transfer-with-Eager-Execution

[4] Neural Style Transfer using ResNet50 with tf.keras, Leo Barbosa, https://github.com/Leo8216/Neural-Style-Transfer-using-ResNet50-with-tf.keras-

[5] Understanding the VGG19 Architecture, Aakash Kaushik https://iq.opengenus.org/vgg19-architecture/

[6] Detailed Guide to Understand and Implement ResNets, Ankit Sachan https://cv-tricks.com/keras/understand-implement-resnets/