NEURAL STYLE TRANSFER

213079001 Kadam Yash Sachin
213070008 Bodayyagari Govardhan Reddy

Outline

- Introduction
- Objectives
- Dataset and Models employed
- Implementation
- Results
- Future Scope
- Conclusion
- References

Introduction

- How would a photograph or a painting done today look like if it were painted by a famous artist of the bygone era?
- Can we achieve this using Neural Networks.
- How do we achieve this?
- What are some of its limitations?

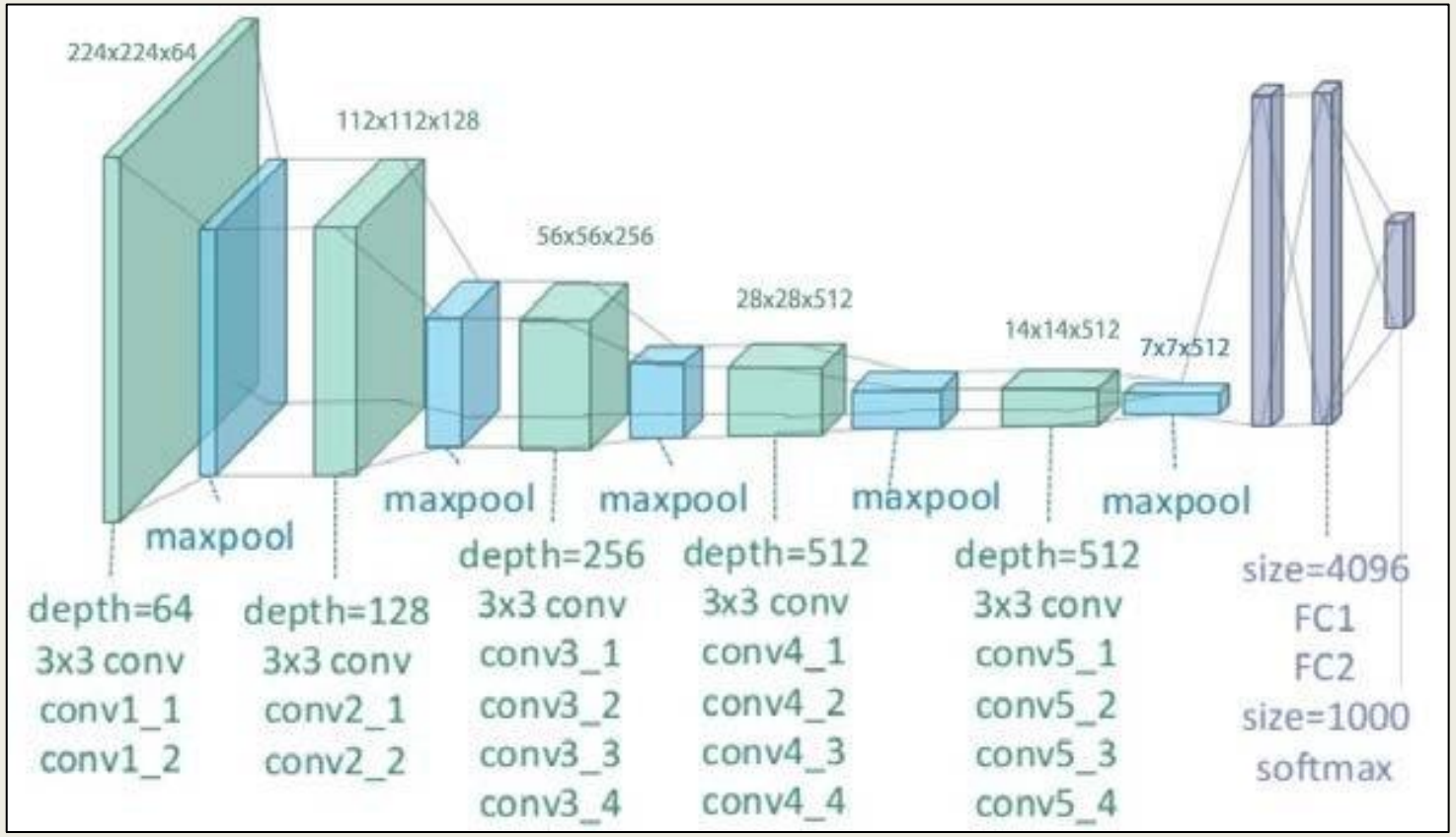
Objectives

- To transfer the style of a famous painter 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 models.

Datasets and Models

- Pretrained Models trained on ImageNet dataset are used:

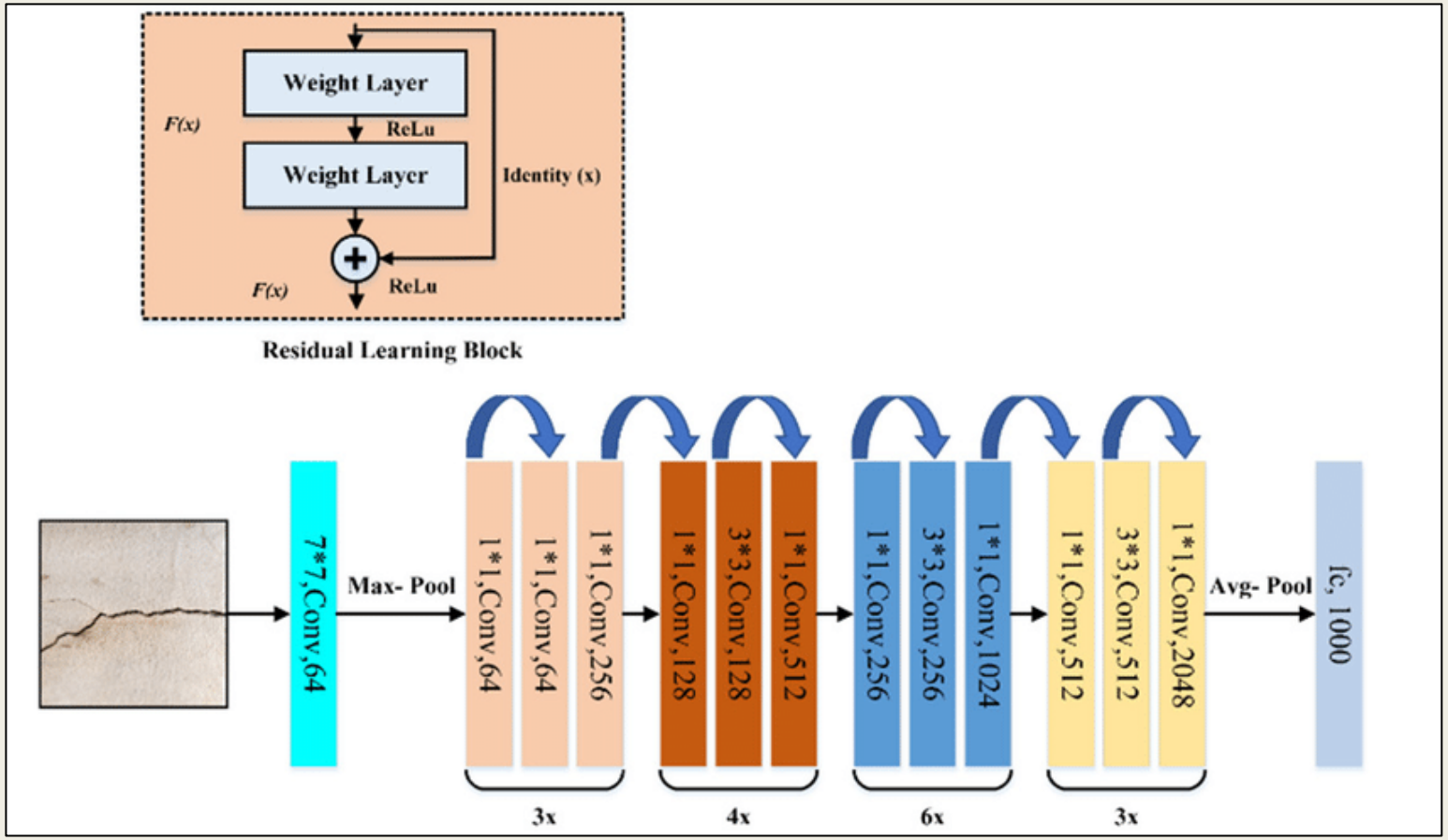
1. VGG19:



Datasets and Models

- Pretrained Models trained on ImageNet dataset are used:

2. ResNet50:



Implementation

- Some layers' outputs of the pretrained models are used to quantify the:

1. **Content:** We use the deeper layers' output feature maps to represent the content in the image.
2. **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.

Implementation

The following are the layers selected in this project:

- VGG19:

1. Content layer:
    1. *block5_conv2*
2. Style layers:
    1. *block1_conv1,*
    2. *block2_conv1,*
    3. *block3_conv1,*
    4. *block4_conv1,*
    5. *block5_conv1*

- ResNet50:

1. Content layer:
    1. *conv5_block3_3_bn*
2. Style layers:
    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*

Implementation

- 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 $F_{ij}^{l}(x) \in C_{nn}(x)$ and $P_{ij}^{l}(p) \in C_{nn}(p)$ describe the respective intermediate feature representation of the network with inputs x and p at layer l.
- Then, $L_{content} = \sum_{ij} \left(F_{ij}^{l}(x) - P_{ij}^{l}(p) \right)^2$

Implementation

- Based on these Style and Content representation we define:

2. **Style Loss**:

- We feed the network our input image and style image.
- Then compute the Gram matrices for the two outputs and compare them:
- $E_l = \frac{1}{4N_l^2 M_l^2} \sum_{ij} \left(G_{ij}^l - A_{ij}^l\right)^2$

where G_{ij}^l and A_{ij}^l are the respective style representation in layer l of input image and style image. N_l describes the number of feature maps, each of size $M_l = height * width$.

- $L_{style} = \sum_{l \in L} w_l E_l$, where w_l is weight of each layer, $w_l = \frac{1}{|L|}$.

Implementation

- Based on these Style and Content representation we define:

3. **Total Loss**:

- $Total\ Loss = \alpha Lconten_t + \beta Lstyle$
- α and β are content and style weights respectively and both are hyperparameters.

Implementation

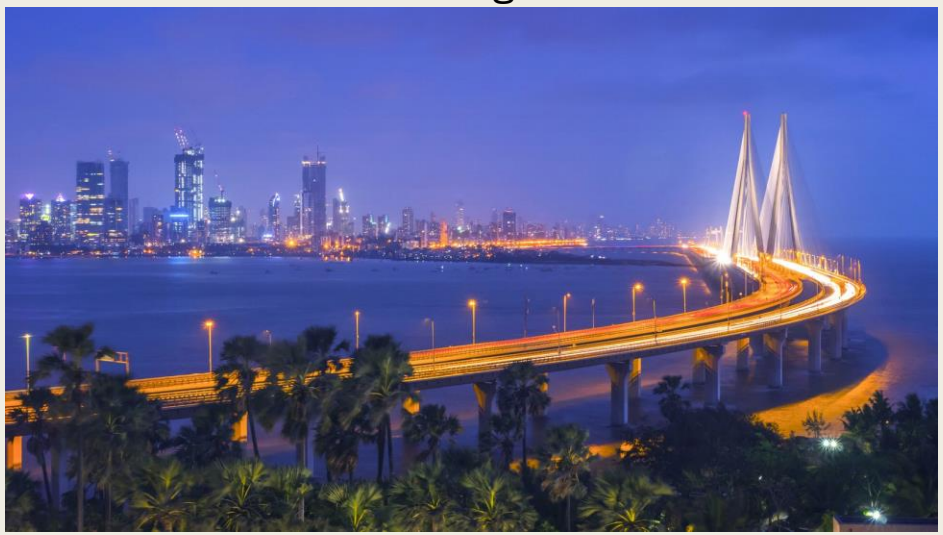
- We initialize the base input image as the content image itself.
- Then based on the hyperparameters α, β, learning rate and number of iterations we go on modifying the base input image using Adam optimizer and gradient descent to reduce the total loss.

Results

- Content image: Mumbai Sealink,
- Style images: Starry Night by Van Gogh; Krishna(Spring in Kulu) by Nicholas Roerich; Pillars of Creation.
- All the following results were obtained by using the following hyperparameters: 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.

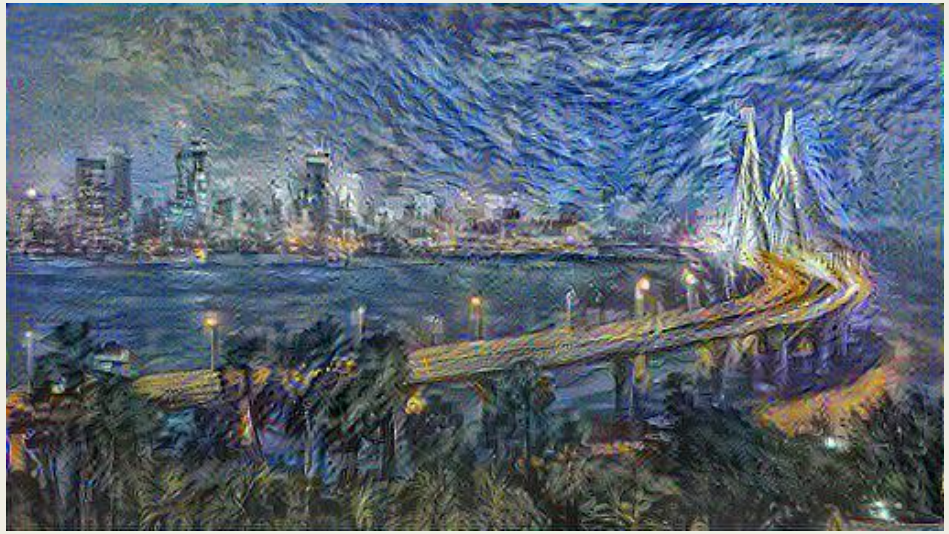
Results

Content Image

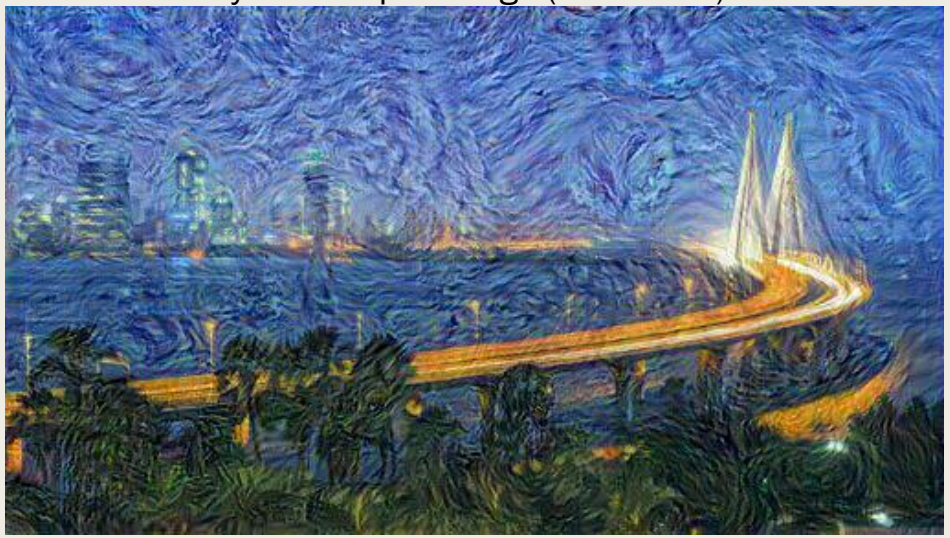


Style Image

Stylized output Image (VGG19)

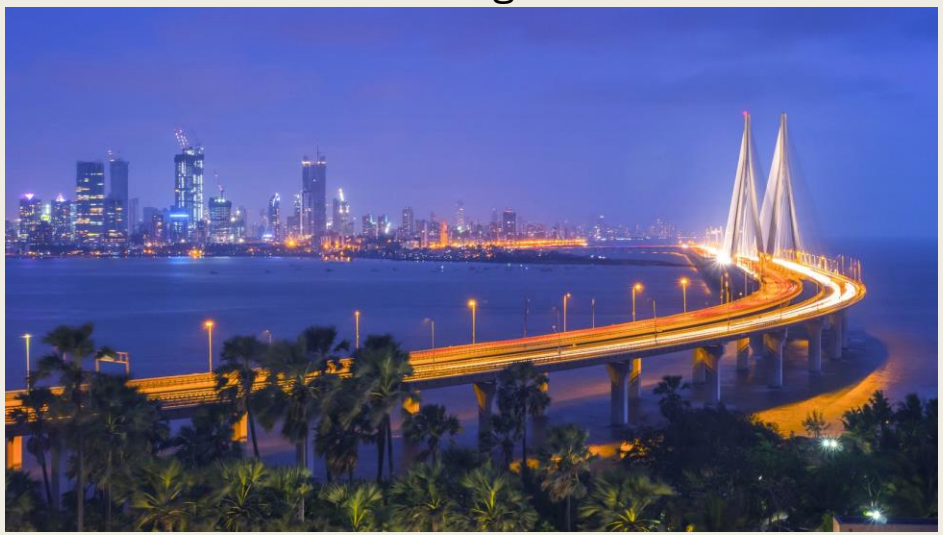


Stylized output Image (ResNet50)

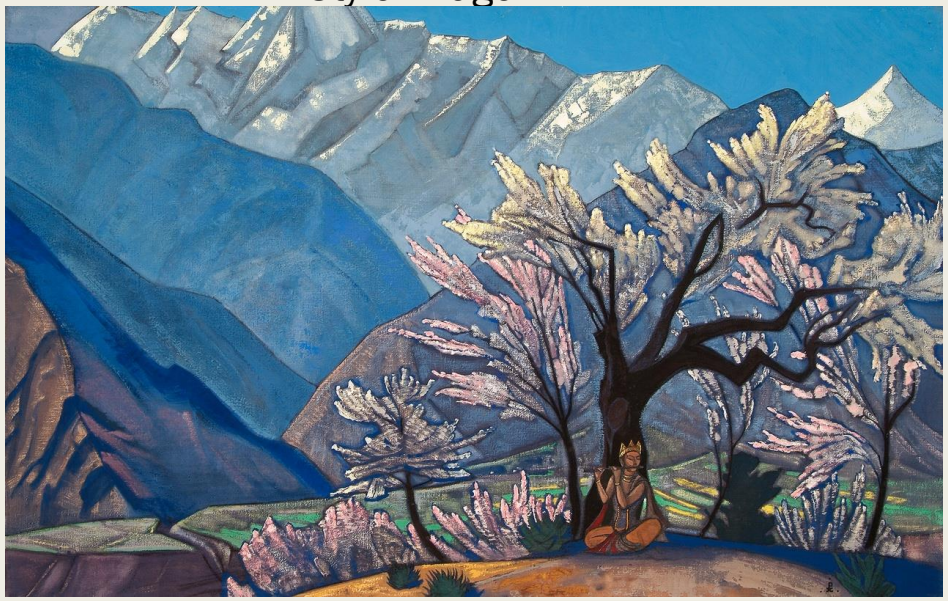


Results

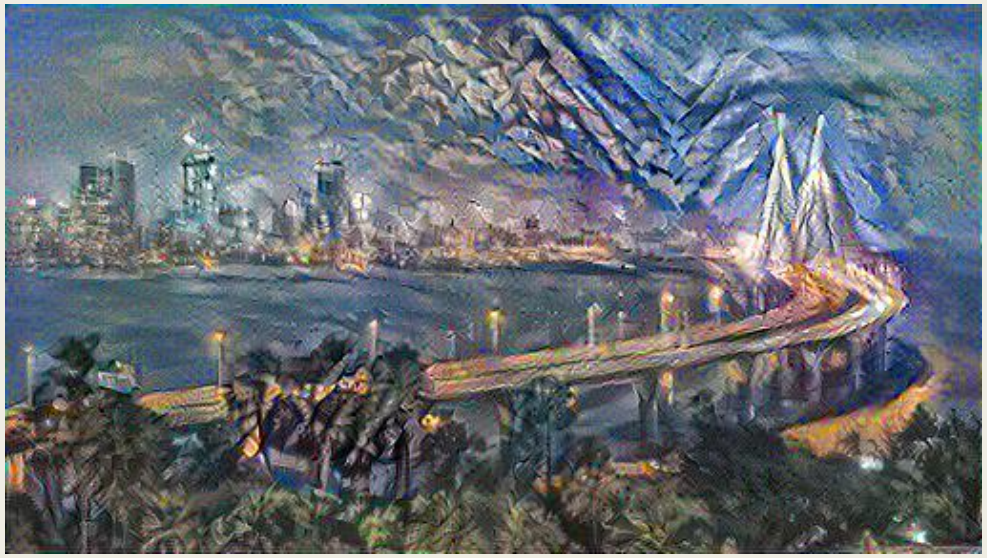
Content Image



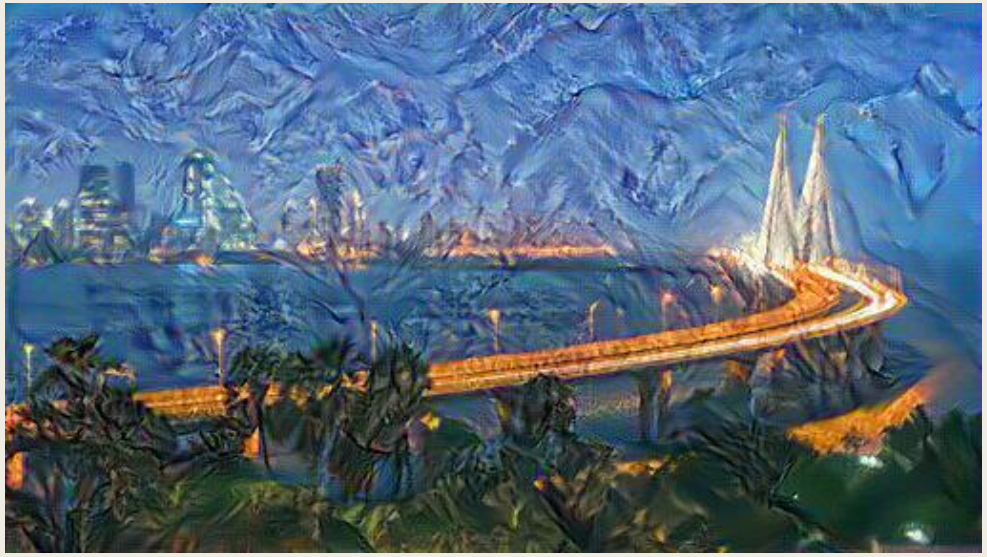
Style Image



Stylized output Image (VGG19)

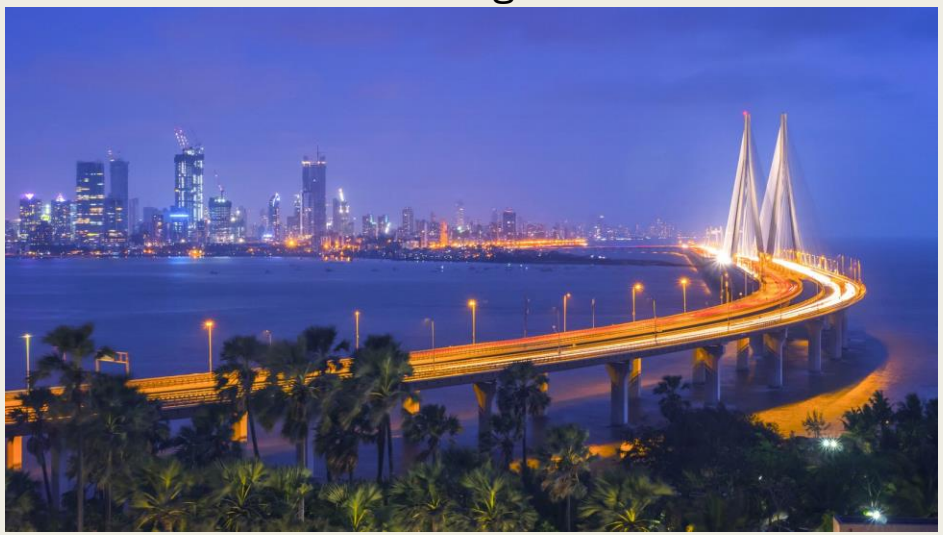


Stylized output Image (ResNet50)



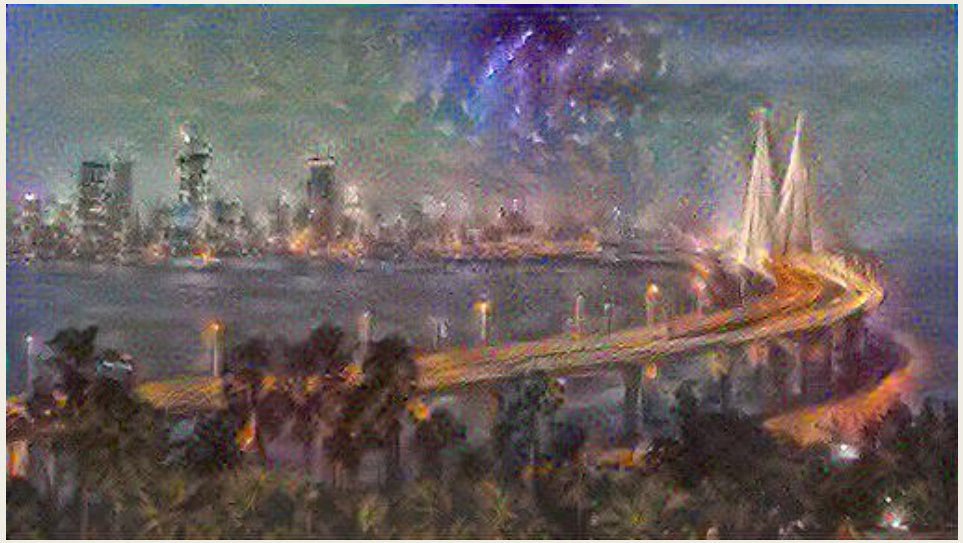
Results

Content Image



Style Image

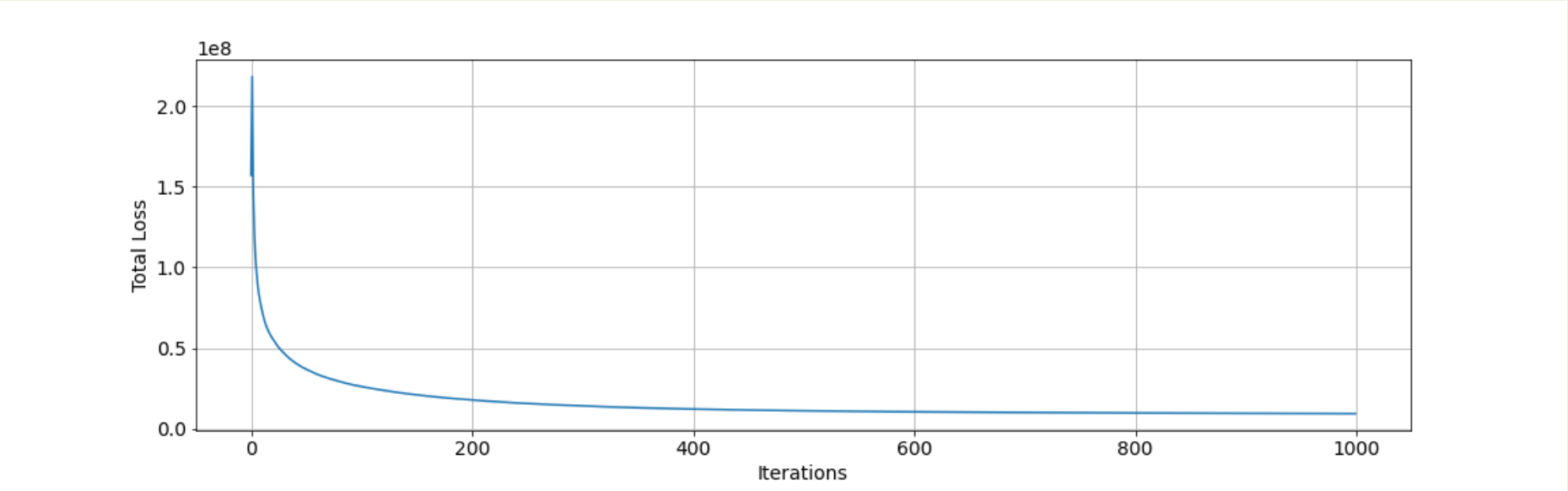
Stylized output Image (VGG19)



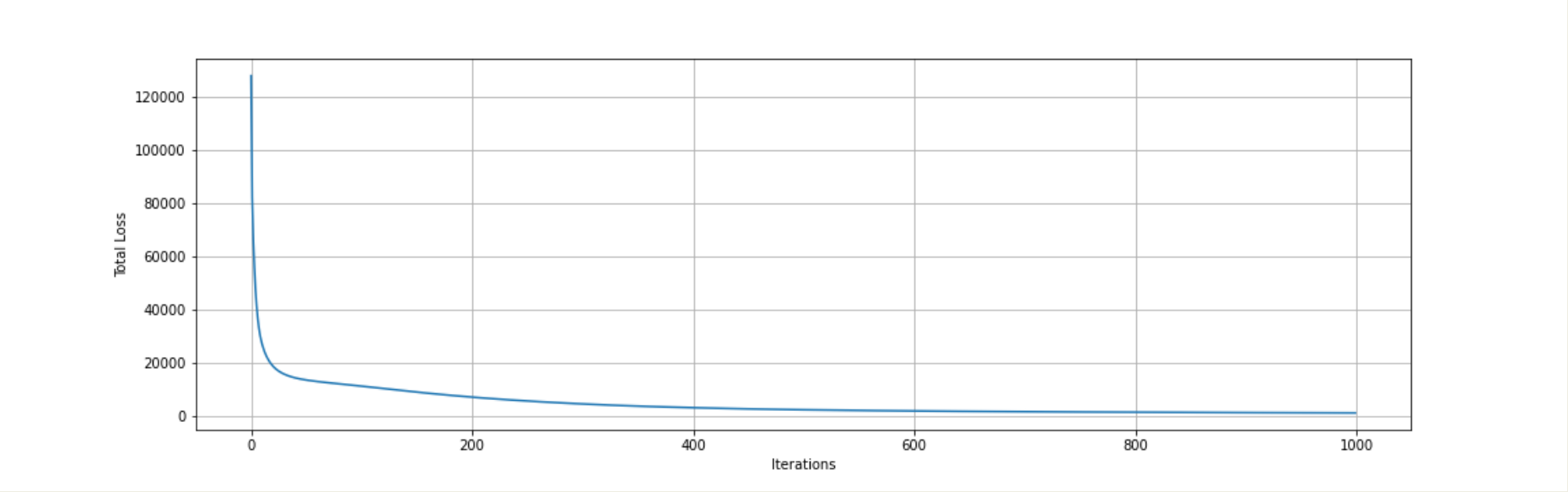
Stylized output Image (ResNet50)

Results (Plot of Total Loss vs Iterations for Starry Night 1)VGG19, 2)ResNet50)

1



2



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.

Conclusion

- 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.

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/

Demo

- Demo Video
  https://drive.google.com/file/d/16zuMYQM2ELqmMMhJP0vvoNz6kWJPjuVX/view?usp=sharing
- ResNet50
  https://colab.research.google.com/drive/1Lx-K4d0adIntZozcZSivXrtMhNawjA74?usp=sharing
- VGG19
  https://colab.research.google.com/drive/1CdCj_SE2vFkacIlNIcgIkg1np9YZWPye?usp=sharing

Contributions

- 213070008 Bodayyagari Govardhan Reddy: Understanding different implementations of the paper [1] in [3] and explaining others, making the VGG19 model work with custom images and fine tuning it for our set of content and style images. Total Loss plots against iterations.
- 213079001 Kadam Yash Sachin: Modification of the VGG19 based model to ResNet50 model based on [4] and modifying it to be used on our set of content and style images. Hyperparameter testing for both the models. Trying out different set of content and style layers.

Sources of code

1. Neural Style transfer with eager execution, Nikhil Agrawal

   https://github.com/nikhilagrawal2000/Neural-Style-Transfer-with-Eager-Execution

2. Neural Style Transfer using ResNet50 with tf.keras, Leo Barbosa,

   https://github.com/Leo8216/Neural-Style-Transfer-using-ResNet50-with-tf.keras-

- Modifications: Use of different content and style layers for ResNet50 model. Total loss vs number of iterations plotted.