**Report**

**Neural Style Transfer**

**Name – Adarsh Shukla En No. - 22123001**

**Overview**

Neural Style Transfer is a technique in deep learning that merges the content of one image with the style of another to create a new, aesthetically appealing image. This project implements Neural Style Transfer using TensorFlow and Keras, leveraging a pre-trained VGG19 model for feature extraction.

**Demo Video**

Click on this demo YouTube video for complete procedure of the Model

**https://www.youtube.com/watch?v=7sgF6nAb32E**

**Repository Contents**

The repository includes:

* Code for performing Neural Style Transfer.
* Pre-trained VGG19 model for feature extraction.
* Implementation of content loss, style loss, and total variation loss functions.
* Example images demonstrating the results of style transfer.

**VGG19 Network**

The VGG19 network consists of 19 layers, including 16 convolutional layers. It uses small convolution filters (3x3) to maintain spatial resolution. For Neural Style Transfer, we utilize specific layers from VGG19 to extract content and style features.

**Content and Style Extraction**

* **Content Layer:** The content of the image is represented by high-level features from a deeper layer in the network. We use the block5\_conv2 layer, which captures complex features like shapes and structures, essential for preserving the content during style transfer.
* **Style Layers:** The style of the image is captured by the correlations between different features. Multiple layers are used to capture different levels of style information:
  + block1\_conv1: Captures low-level features like edges and textures.
  + block2\_conv1: Captures more complex patterns.
  + block3\_conv1, block4\_conv1, block5\_conv1: Capture progressively higher-level patterns and textures.

By combining features from these layers, the model effectively blends the style of the style image with the content of the content image.

**Implementation Details**

* **Feature Extraction:** The VGG19 model extracts feature maps from the specified content and style layers for both the content and style images.
* **Content Loss:** Measures the difference between the feature maps of the content image and the generated image at the block5\_conv2 layer, ensuring the generated image retains the content of the original content image.
* **Style Loss:** Computed by comparing the Gram matrices of the feature maps from the specified style layers between the style image and the generated image, ensuring the generated image adopts the style characteristics of the original style image.
* **Total Loss:** A weighted sum of the content loss and style loss, minimized during optimization to generate the final stylized image.

By carefully selecting and combining these layers, the model produces images that effectively blend the content of one image with the style of another, creating a visually appealing result.

**Loss Functions**

**Content Loss**

The content loss measures the difference in content between the generated image and the content image. It is computed using the mean squared error between the feature representations of the content image and the generated image from the content layer.

python

Copy code

def get\_content\_loss(noise, target):

loss = tf.reduce\_mean(tf.square(noise - target))

return loss

**Style Loss**

The style loss measures the difference in style using Gram matrices. The Gram matrix captures the correlations between different filter responses, representing the style of the image. The style loss is computed as the mean squared error between the Gram matrices of the style image and the generated image.

python

Copy code

def get\_style\_loss(noise, target):

gram\_noise = gram\_matrix(noise)

loss = tf.reduce\_mean(tf.square(target - gram\_noise))

return loss

**Total Loss**

The total loss is a weighted sum of the content loss and style loss. This combined loss is used as the optimization objective during the training process.

python

Copy code

def compute\_loss(model, loss\_weights, image, gram\_style\_features, content\_features):

...

total\_loss = content\_loss \* content\_weight + style\_loss \* style\_weight

return total\_loss, style\_loss, content\_loss

**Usage**

**Running the Code**

1. **Upload Images:** The script prompts you to upload a content image and a style image.
2. **Run Style Transfer:** To run the style transfer, call the run\_style\_transfer function with the appropriate parameters:

python

Copy code

best, best\_loss, image = run\_style\_transfer(content\_path, style\_path, epochs=2000)

1. **Display Results:** The script includes functionality to display the content, style, and generated images using Matplotlib.

**Code Execution**

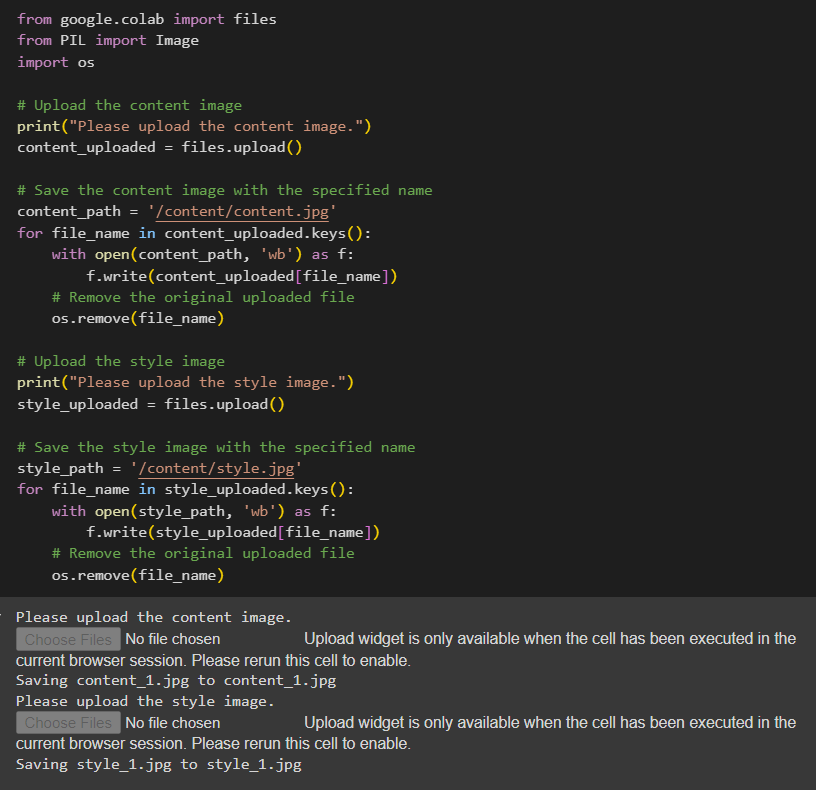
Run each cell in the provided Colab notebook. <https://colab.research.google.com/github/adarshukla3005/neural_style_transfer/blob/main/Neural_Style_Transfer.ipynb>

**Take Input images:**

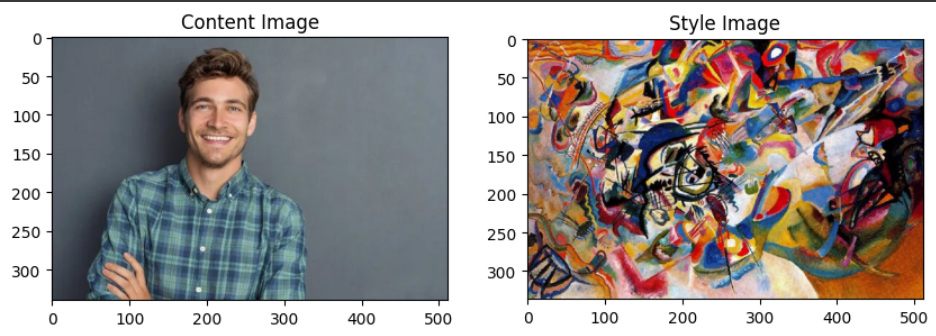
Upload the required content and style image

files. You can download the files from the repository. Make sure to take input content and

Style images.



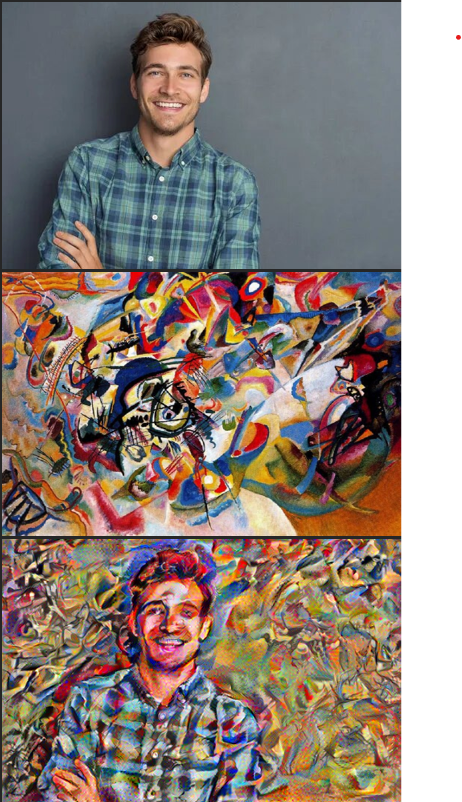
**Preview images**:



**Some Stylized images**



**Transfer Results**



**References**

[A Neural Algorithm of Artistic Style](https://arxiv.org/abs/1508.06576) by Leon A. Gatys, Alexander S. Ecker, Matthias Bethge

[TensorFlow Neural Style Transfer Tutorial](https://www.tensorflow.org/tutorials/generative/style_transfer)

https://hackernoon.com/how-do-neural-style-transfers-work-7bedaee0559a

**Conclusion**

The Neural Style Transfer project demonstrates the power of deep learning in creating visually appealing images by combining the content of one image with the style of another. By leveraging a pre-trained VGG19 model and carefully crafting loss functions, the project achieves impressive results in artistic image generation. This implementation serves as a foundational approach that can be further optimized and extended for various artistic and practical applications.