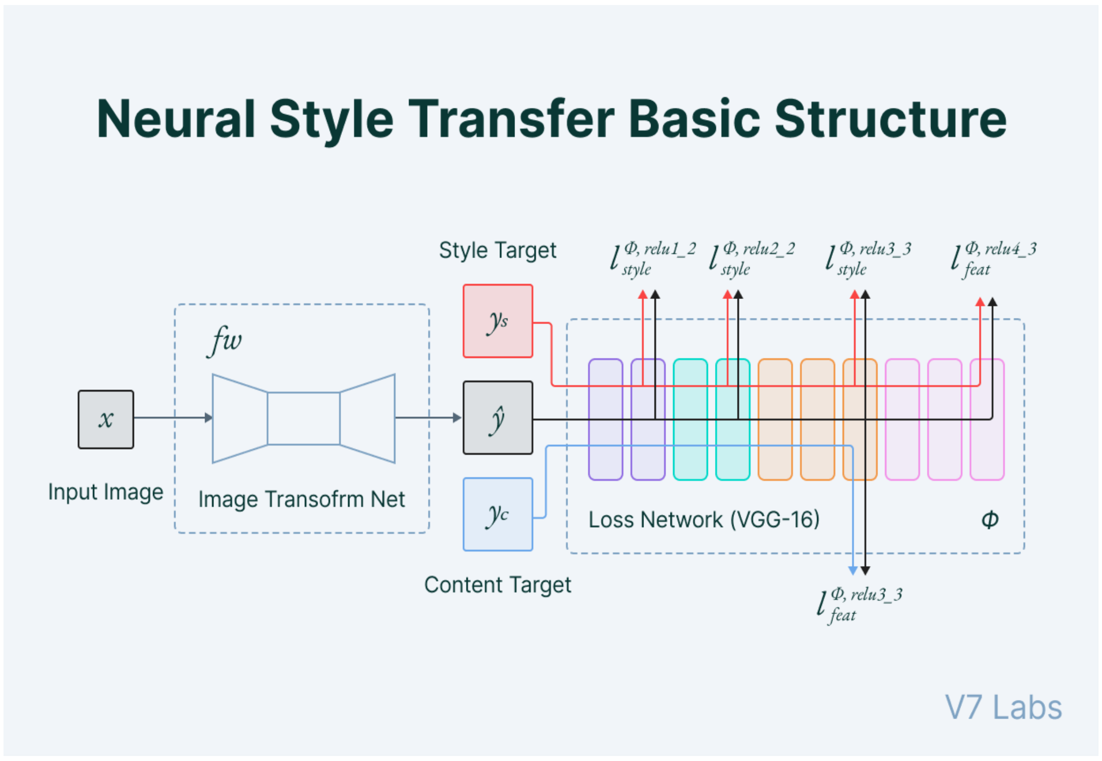
Experiment No. 3

Problem Statement- Create and Art with Neural style transfer on given image using deep learning.

Theory:

Neural style transfer is a fascinating technique in deep learning that combines the content of one image with the style of another, creating captivating artistic results. This process involves leveraging convolutional neural networks to extract features from both the content image and the style reference image. By optimizing a target image to match the content features of the content image and the style features of the style reference image simultaneously, neural style transfer generates a new image that embodies the content of the original image in the artistic style of the reference image. The key to this technique lies in understanding and manipulating the feature representations extracted by the neural network at different layers, balancing the preservation of content and the infusion of style. The result is a visually appealing fusion of content and style, showcasing the power of deep learning algorithms to create unique and expressive art forms.


Neural Style Transfer Basic Structure
Style Target
$l_{style}^{\Phi, relu1\_2}$
$l_{style}^{\Phi, relu2\_2}$
$l_{style}^{\Phi, relu3\_3}$
$l_{feat}^{\Phi, relu4\_3}$
y_s
f_W
x
$\hat{y}$
Input Image
Image Transofrm Net
y_c
Loss Network (VGG-16)
Φ
Content Target
$l_{feat}^{\Phi, relu3\_3}$
V7 Labs

Content loss

It helps to establish similarities between the content image and the generated image. It is intuitive that higher layers of the model focus more on the features present in the image i.e. overall content of the image. Content loss is calculated by Euclidean distance between the respective intermediate higher-level feature representation of input image (x) and content image (p) at layer l.

$$L^l_{content}(p, x) = \sum_{i,j} (F^l_{ij}(x) - P^l_{ij}(p))^2$$

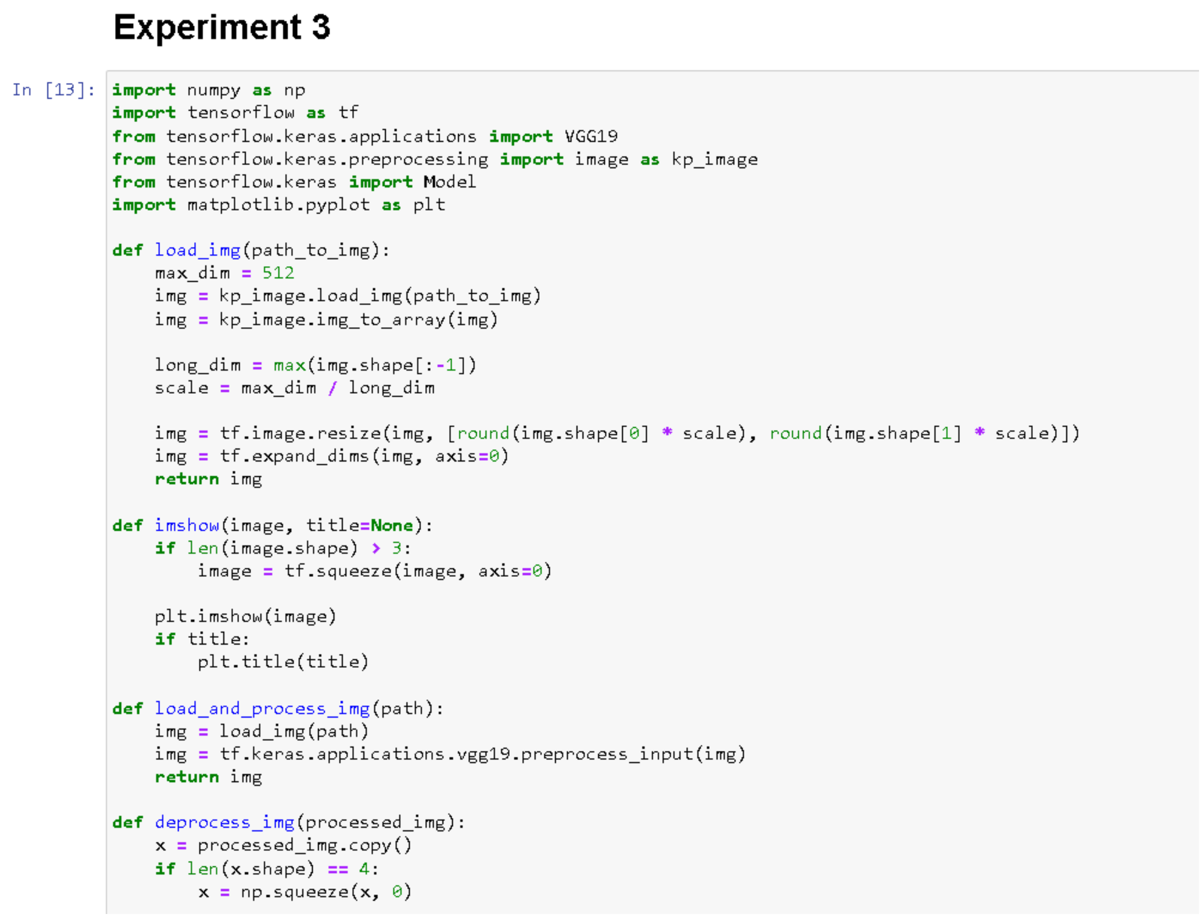
It is natural for a model to produce different feature maps in higher layers being activated in the presence of different objects. This helps us to deduce that images having the same content should also have similar activations in the higher layers

Style loss

Style loss is conceptually different from Content loss. We cannot just compare the intermediate features of the two images and get the style loss. That's why we introduce a new term called Gram matrices. Gram matrix is a way to interpret style information in an image as it shows the overall distribution of features in a given layer. It is measured as the amount of correlation present between features maps in a given layer.

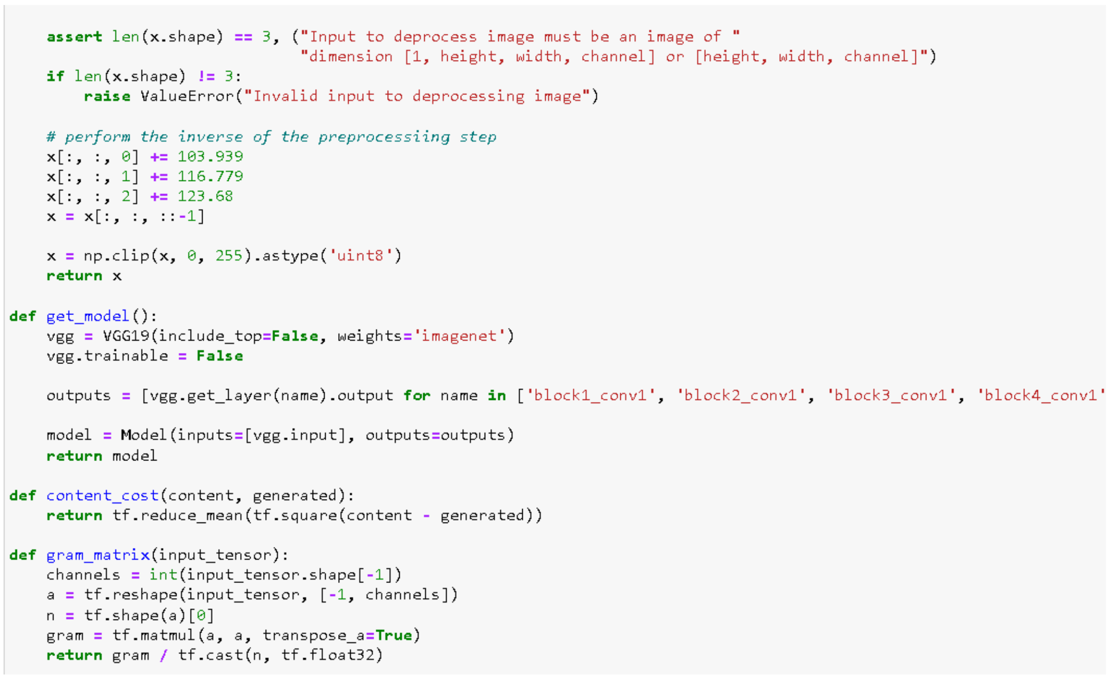
Conclusion: Thus, we have successfully implemented ART network.

Code and Output:



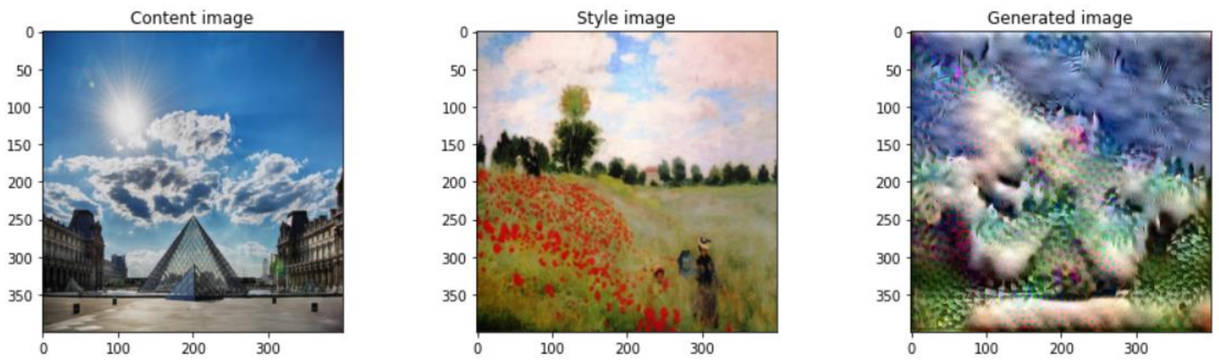
Experiment 3

```
In [13]: import numpy as np
         import tensorflow as tf
         from tensorflow.keras.applications import VGG19
         from tensorflow.keras.preprocessing import image as kp_image
         from tensorflow.keras import Model
         import matplotlib.pyplot as plt

         def load_img(path_to_img):
             max_dim = 512
             img = kp_image.load_img(path_to_img)
             img = kp_image.img_to_array(img)

             long_dim = max(img.shape[:-1])
             scale = max_dim / long_dim

             img = tf.image.resize(img, [round(img.shape[0] * scale), round(img.shape[1] * scale)])
             img = tf.expand_dims(img, axis=0)
             return img

         def imshow(image, title=None):
             if len(image.shape) > 3:
                 image = tf.squeeze(image, axis=0)

             plt.imshow(image)
             if title:
                 plt.title(title)

         def load_and_process_img(path):
             img = load_img(path)
             img = tf.keras.applications.vgg19.preprocess_input(img)
             return img

         def deprocess_img(processed_img):
             x = processed_img.copy()
             if len(x.shape) == 4:
                 x = np.squeeze(x, 0)
```



```
    assert len(x.shape) == 3, ("Input to deprocess image must be an image of "
                               "dimension [1, height, width, channel] or [height, width, channel]")
    if len(x.shape) != 3:
        raise ValueError("Invalid input to deprocessing image")

    # perform the inverse of the preprocessing step
    x[:, :, 0] += 103.939
    x[:, :, 1] += 116.779
    x[:, :, 2] += 123.68
    x = x[:, :, ::-1]

    x = np.clip(x, 0, 255).astype('uint8')
    return x

def get_model():
    vgg = VGG19(include_top=False, weights='imagenet')
    vgg.trainable = False

    outputs = [vgg.get_layer(name).output for name in ['block1_conv1', 'block2_conv1', 'block3_conv1', 'block4_conv1'

    model = Model(inputs=[vgg.input], outputs=outputs)
    return model

def content_cost(content, generated):
    return tf.reduce_mean(tf.square(content - generated))

def gram_matrix(input_tensor):
    channels = int(input_tensor.shape[-1])
    a = tf.reshape(input_tensor, [-1, channels])
    n = tf.shape(a)[0]
    gram = tf.matmul(a, a, transpose_a=True)
    return gram / tf.cast(n, tf.float32)
    for i in range(num_iterations):
        grads = compute_grads(cfg)
        opt.apply_gradients([(grads, generated)])

        J_content = content_cost(content_targets[num_content_layers], model(generated)[num_content_layers])
        J_style = 0
        for layer in range(num_content_layers):
            J_style += style_weights[layer] * style_cost(style_targets[layer], model(generated)[layer])

        J_total = total_cost(J_content, J_style, content_weight, style_weight)

        if J_total < best_loss:
            best_loss = J_total
            best_img = generated.numpy()

        if i % 100 == 0:
            print("Iteration {}, Total Loss: {:.2f}, Style Loss: {:.2f}, Content Loss: {:.2f}".format(
                i, J_total, J_style, J_content))

    return best_img

content_path = r"C:\Vridhi\BE - AIDS\sem 8\CI\content.jpg"
style_path = r"C:\Vridhi\BE - AIDS\sem 8\CI\style.jpg"
```

```
In [14]: # Show the 3 images in a row
         fig = plt.figure(figsize=(16, 4))
         ax = fig.add_subplot(1, 3, 1)
         imshow(content_image[0])
         ax.title.set_text('Content image')
         ax = fig.add_subplot(1, 3, 2)
         imshow(style_image[0])
         ax.title.set_text('Style image')
         ax = fig.add_subplot(1, 3, 3)
         imshow(generated_image[0])
         ax.title.set_text('Generated image')
         plt.show()
```


Content image
Style image
Generated image
0
50
100
150
200
250
300
350
0
100
200
300