# Neural Style Transfer
# KUNAL JAIN 22113079

Neural Style Transfer (NST) is a fascinating application of deep learning where the style of one image is applied to the content of another image, creating a hybrid image that combines the content of the former with the artistic style of the latter. This technique uses Convolutional Neural Networks (CNNs) to achieve impressive results.

In this project, you will learn how to implement neural style transfer using a pre-trained CNN, such as VGG19, to extract and manipulate image features.

**Goals**

1. **Understand the Basics of Neural Style Transfer**: Learn how NST uses CNNs to separate and recombine the content and style of images.
2. **Data Preparation**: Load and preprocess the content and style images.
3. **Model Construction**: Use a pre-trained CNN to extract content and style features.
4. **Optimization**: Optimize an input image to combine the content of one image with the style of another.
5. **Visualization**: Display the resulting stylized image.

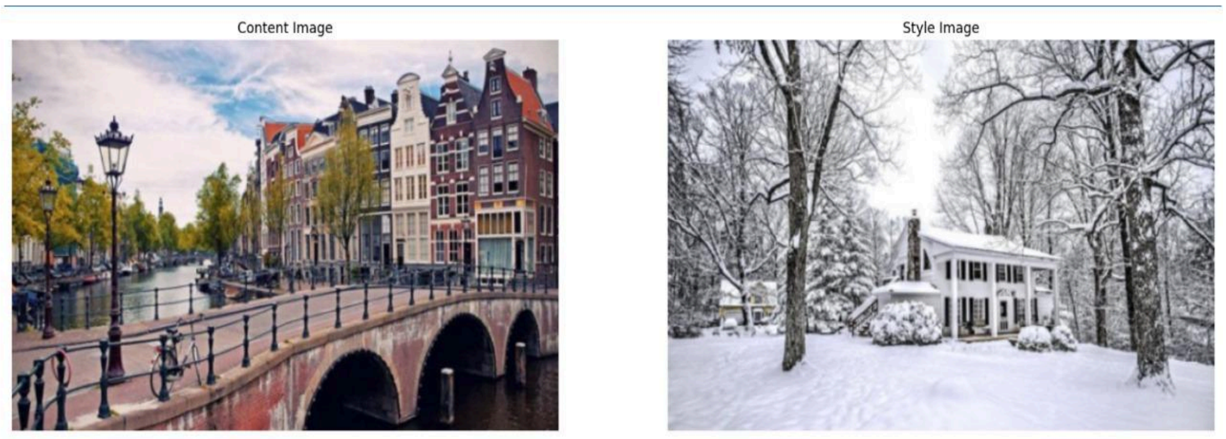
## Key Concepts

1. **Content Representation**: Extracted from intermediate layers of a CNN that capture the high-level structure and objects in an image.
2. **Style Representation**: Extracted from multiple layers of a CNN that capture textures, colors, and patterns.
3. **Loss Functions**:
   - **Content Loss**: Measures the difference between the content of the generated image and the content image.
   - **Style Loss**: Measures the difference between the style of the generated image and the style image using Gram matrices.
   - **Total Variation Loss**: Encourages spatial smoothness in the generated image.

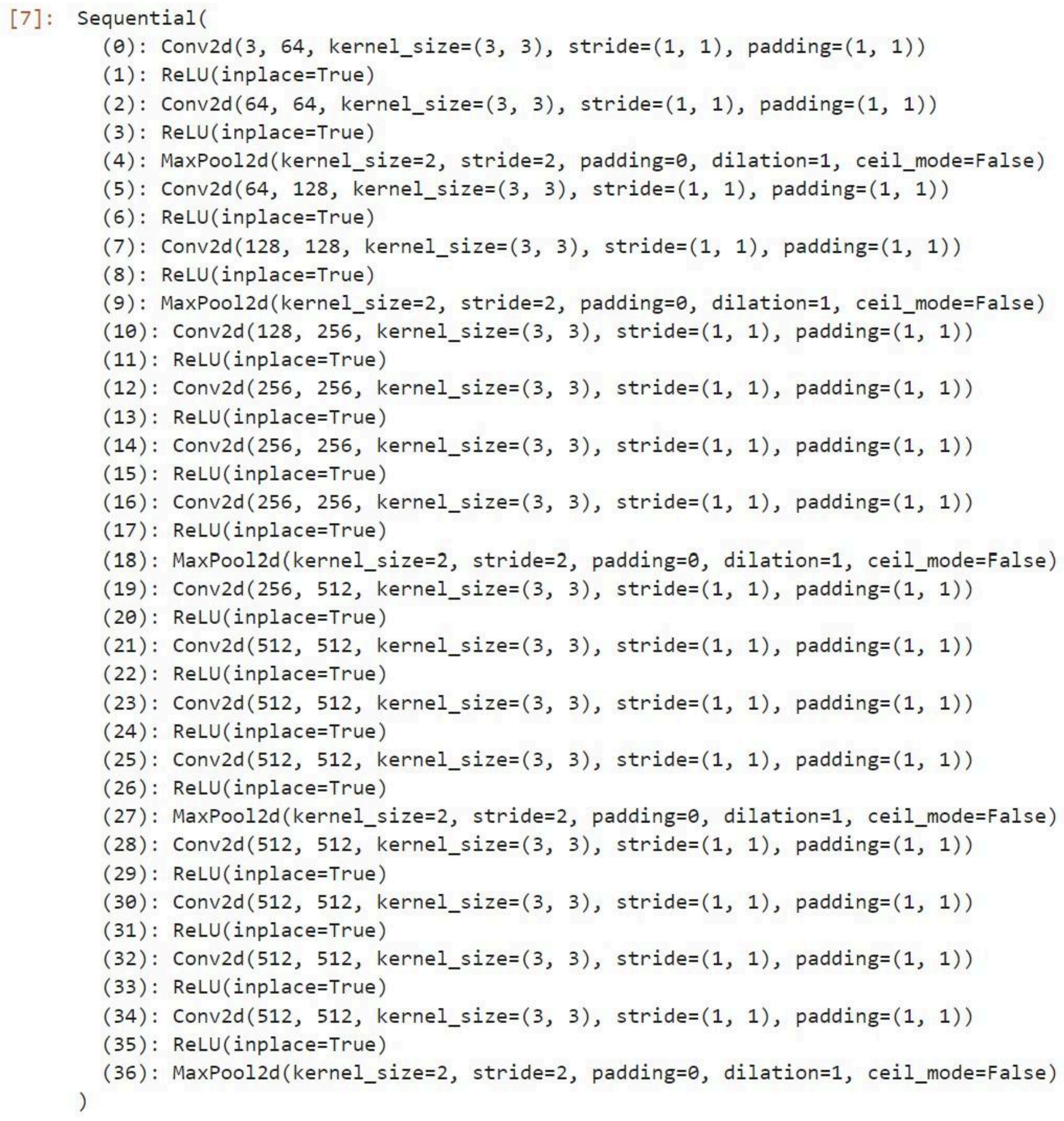
## Steps

1. **Load and Preprocess Images**:
   - Load the content and style images.
   - Preprocess the images to match the input requirements of the pre-trained CNN.
2. **Build the Model**:
   - Use a pre-trained CNN (e.g., VGG19) to extract features from the content and style images.
   - Define a model that outputs the content and style representations.

3. **Define the Loss Functions**:
    - Compute the content loss and style loss.
    - Combine these losses to define the total loss function.
4. **Optimization**:
    - Initialize the generated image (typically as a copy of the content image).
    - Use an optimizer (e.g., Adam) to minimize the total loss with respect to the generated image.
5. **Visualize the Results**:
    - Display the generated image after optimization.



These are my input content and style images
Number of steps used are 1550 with 4 steps

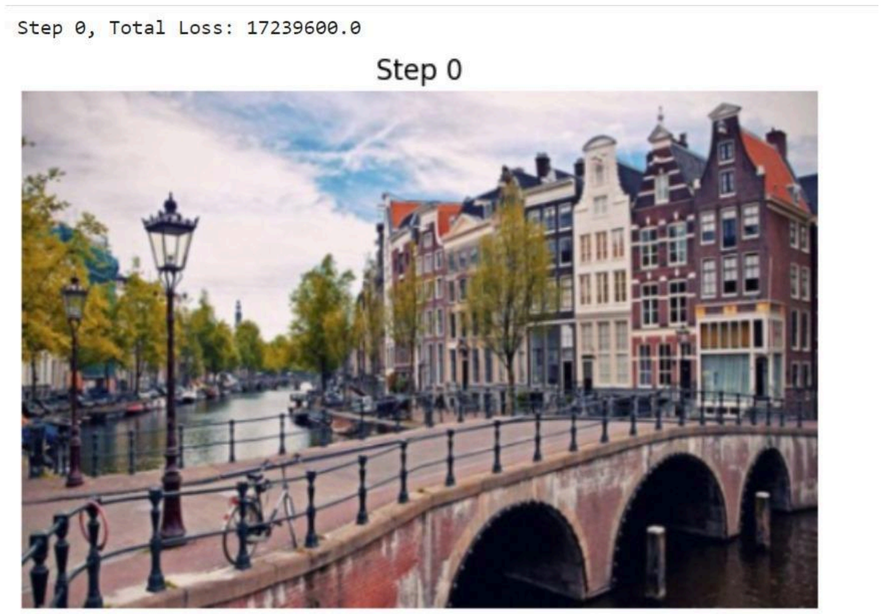


```
[7]:  Sequential(
        (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): ReLU(inplace=True)
        (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (3): ReLU(inplace=True)
        (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
        (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (6): ReLU(inplace=True)
        (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (8): ReLU(inplace=True)
        (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
        (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (11): ReLU(inplace=True)
        (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (13): ReLU(inplace=True)
        (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (15): ReLU(inplace=True)
        (16): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (17): ReLU(inplace=True)
        (18): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
        (19): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (20): ReLU(inplace=True)
        (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (22): ReLU(inplace=True)
        (23): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (24): ReLU(inplace=True)
        (25): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (26): ReLU(inplace=True)
        (27): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
        (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (29): ReLU(inplace=True)
        (30): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (31): ReLU(inplace=True)
        (32): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (33): ReLU(inplace=True)
        (34): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (35): ReLU(inplace=True)
        (36): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      )
```

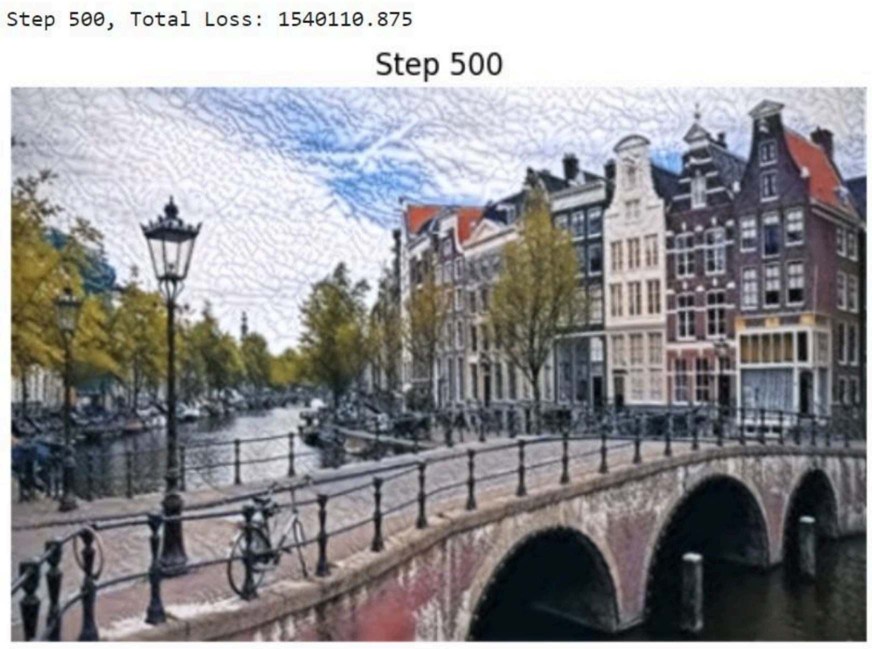
**Input Layer**: Accepts the input image.
**Convolutional Layer**: Applies multiple convolutional filters to the input to extract features.
**Activation Layer (ReLU)**: Adds non-linearity to the network.
**Pooling Layer**: Reduces the spatial dimensions of the feature maps.

**Fully Connected Layer**: Combines the features extracted by the convolutional layers for classification.
**Output Layer**: Provides the final classification output.

After the intermediate steps the result are attached below
Step 0

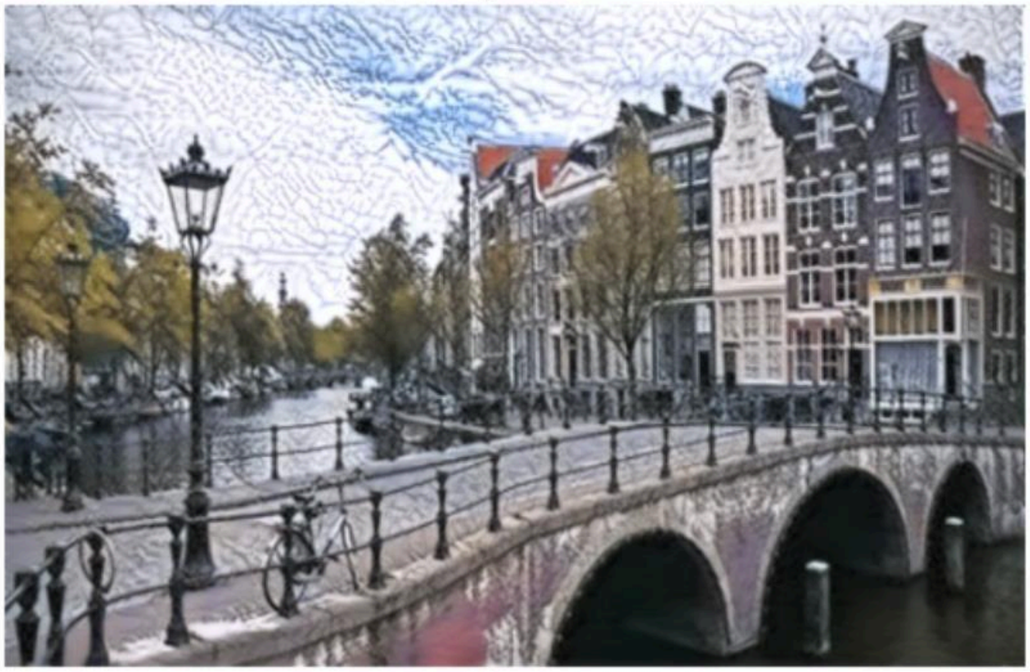


```
Step 0, Total Loss: 17239600.0
```



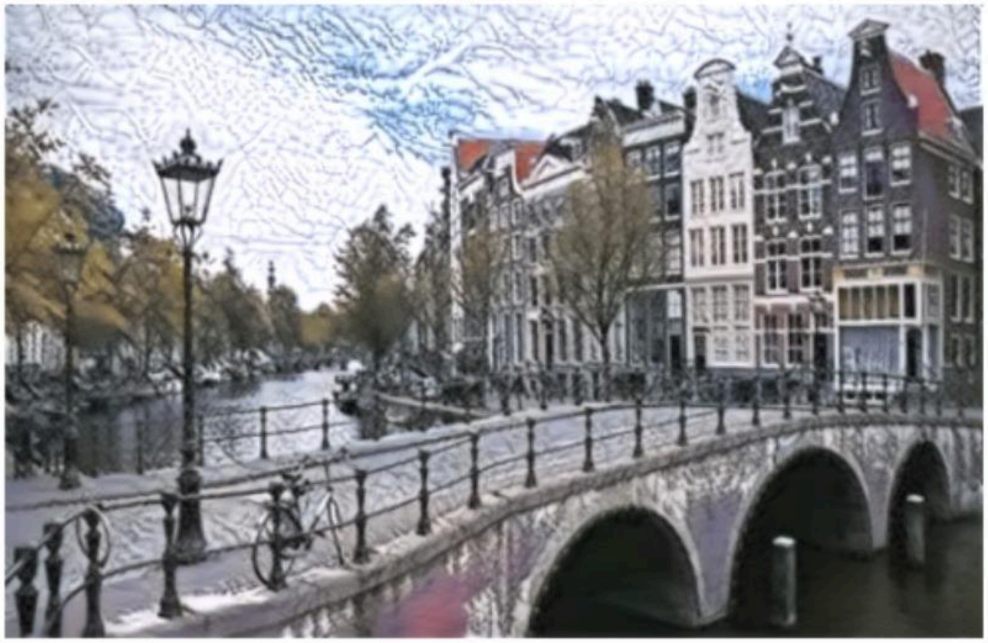
```
Step 500, Total Loss: 1540110.875
Step 1000, Total Loss: 897090.3125
```

Step 1000

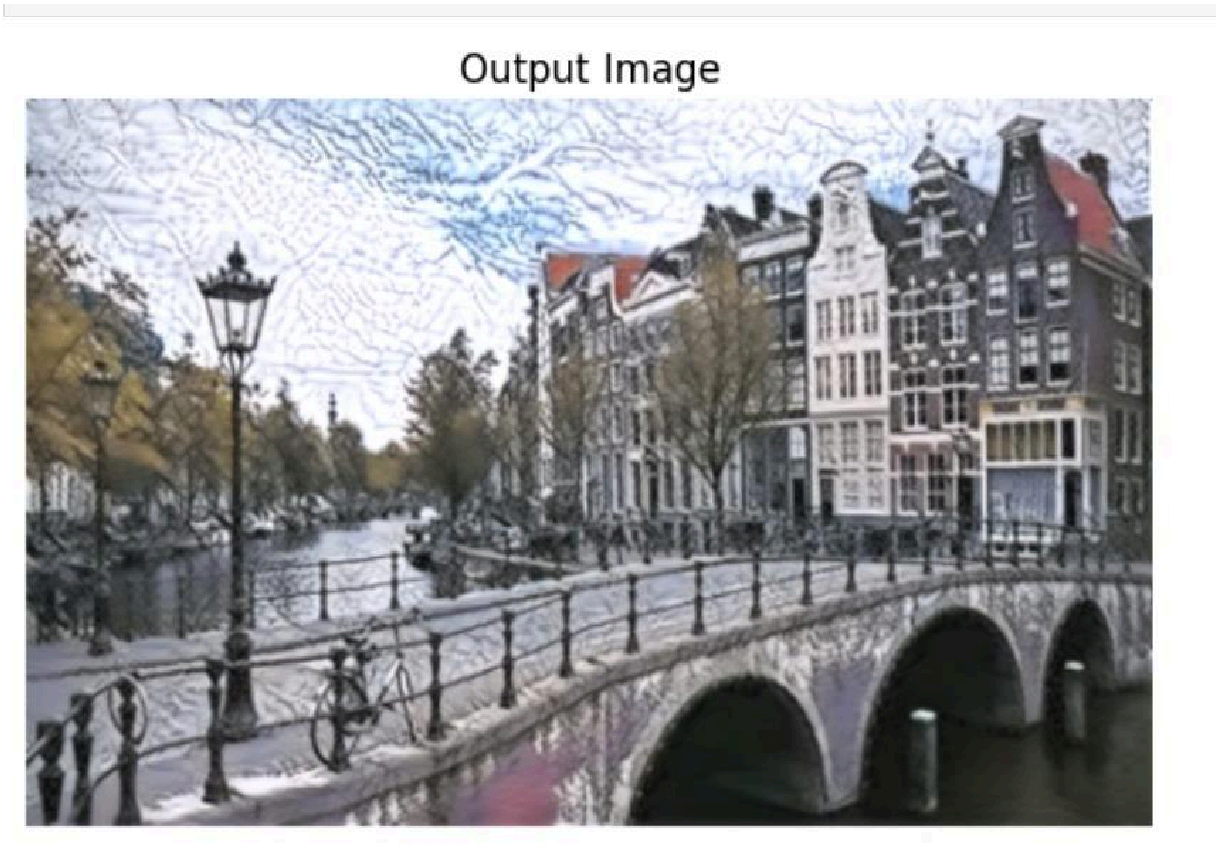


```
Step 1500, Total Loss: 660499.875
```

Step 1500



Final output image



The neural style transfer project effectively integrated advanced deep learning techniques with artistic image processing, utilizing pretrained models and optimization algorithms to synthesize visually captivating images. This endeavor not only demonstrated the technical implementation of neural style transfer but also underscored its potential to enhance creativity and visual expression through machine learning methods.

By combining theoretical insights with practical application, this project contributes significantly to the fields of computer vision and deep learning. It offers valuable insights into how artificial intelligence can enrich human creativity in visual arts and design.

Successfully completing this project establishes a solid foundation for future research and applications in image synthesis, computational creativity, and interdisciplinary collaborations at the convergence of AI and visual arts.