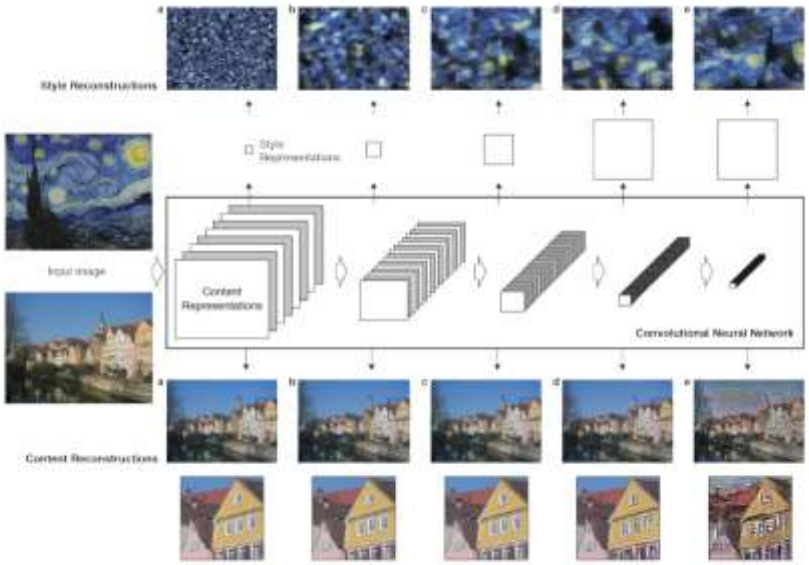
NEURAL STYLE TRANSFER

An advanced system using Deep Neural Networks (DNNs) has been developed to generate artistic images of exceptional perceptual quality. This system employs neural representations to distinctly separate and then recombine the content and style of any given images, facilitating a neural algorithm for artistic creation. At the core of this system are Convolutional Neural Networks (CNNs), a type of DNN highly effective in image processing tasks. CNNs consist of multiple layers of small computational units that process visual information in a hierarchical, feed-forward manner. Each layer functions as a collection of image filters that extract specific features from the input image. As a result, the output of each layer is a series of feature maps, which are different filtered versions of the input image.

In the higher layers of the network, high-level content such as objects and their spatial arrangements are captured, without focusing on the exact pixel values. Conversely, lower layers produce reconstructions that closely match the original pixel values of the image. Hence, the feature responses in the higher layers are known as the content representation.

To capture the style of an input image, the system uses a feature space designed to encapsulate texture information. This feature space is derived from the filter responses in each layer of the network, emphasizing the correlations between these responses across the spatial dimensions of the feature maps. By integrating feature correlations from multiple layers, the system generates a stationary, multi-scale representation of the input image. This representation effectively captures the texture details without retaining the global arrangement of the image.

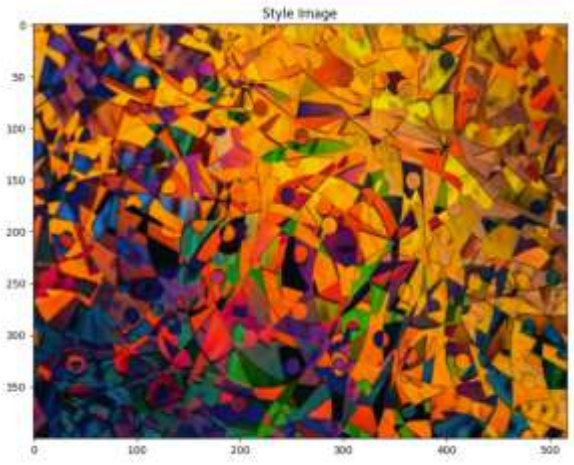
For example:



We reconstruct the input image from layers ‘conv1 1’ (a), ‘conv2 1’ (b), ‘conv3 1’ (c), ‘conv4 1’ (d) and ‘conv5 1’ (e) of the original VGG-Network. The style representation computes correlations between the different features in different layers of the CNN. We reconstruct the style of the input image from style representations built on different subsets of CNN layers (‘conv1 1’ (a), ‘conv1 1’ and ‘conv2 1’ (b), ‘conv1 1’, ‘conv2 1’ and ‘conv3 1’ (c), ‘conv1 1’, ‘conv2 1’, ‘conv3 1’ and ‘conv4 1’ (d), ‘conv1 1’, ‘conv2 1’, ‘conv3 1’, ‘conv4 1’ and ‘conv5 1’ (e)).

The key finding is that the representations of content and style in the Convolutional Neural Network are separable. That is, we can manipulate both representations independently to produce new, perceptually meaningful images. We generate images that mix the content and style representation from two different source images. While the global arrangement of the original photograph is preserved, the colours and local structures that compose the global scenery are provided by the artwork. This process effectively transforms the photograph into the style of the artwork, making the synthesized image resemble the artistic work while retaining the original content of the photograph.

The style representation is a multi-scale representation that includes multiple layers of the neural network. The style representation included layers from the whole network hierarchy.



The above are the content and style images which are used in my neural style transfer project .

Project Explanation

Import necessary libraries. PyTorch library for tensor computation. PyTorch's vision library, which includes pre-trained models and image transformations. Optimizer module for gradient descent optimization. Python Imaging Library which is used for image manipulation. Library for plotting images and graphs , creating deep copies of objects , fundamental package for numerical computing and for interacting with the operating system.

Load and preprocess the images. Load the image and convert it to RGB format . Resize the image while maintaining aspect ratio and apply transformations to convert image to a tensor and normalize it. Displays the content and style images using 'im_convert' function.

```
[3]: device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
     device

[3]: device(type='cpu')
```

This function checks if CUDA-enabled GPU devices are available and properly configured on your system. CUDA is a parallel computing platform and application programming interface (API) model created by NVIDIA, which allows developers to use NVIDIA GPUs for general-purpose processing (like deep learning computations). It returns CPU means CUDA-capable GPUs are not available or properly configured and it specifies that tensors and operations should be executed on the CPU.

Below is the set of content and style images used in project

VGG-19 Model:It is a convolutional neural network (CNN) architecture that was proposed by the Visual Geometry Group (VGG) at the University of Oxford. It is well-known for its simplicity and effectiveness in image classification tasks. It is used to extract features that capture the content of the content image. These features are extracted from deeper layers of the network where the information about objects and their structures in the image is encoded. It extracts features from the style image that capture the textures, colors, and patterns characteristic of the style image. These features are typically extracted from

multiple layers spanning the network, capturing both low-level and high-level style elements.

In the project, specific layers are chosen from the VGG-19 model for both content and style feature extraction. Typically, a middle layer (e.g., 'conv4_2') is chosen to extract content features. This layer strikes a balance between capturing high-level content information and being computationally feasible. Multiple layers (e.g., 'conv1_1', 'conv2_1', 'conv3_1', 'conv4_1', 'conv5_1') are chosen to capture style features at different scales and levels of abstraction.

The VGG-19 model used in the project is pretrained on the ImageNet dataset. This pretrained model serves as a feature extractor because it has already learned to recognize a wide range of visual patterns and features, making it suitable for extracting meaningful content and style representations from images.Load the model and freeze all model parameters to prevent updating during optimization process. Use the functions and extract features from specified layers of VGG model, content image and style image.

```
[6]: 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)
     )
```

After extracting features from the chosen layers, the style features are used to compute Gram matrices. These matrices capture the correlations between different feature maps at each layer, representing the style of the image. The Gram matrix operation involves reshaping the feature maps into matrices and computing their dot product. Calculate the gram matrix from the feature tensor and store gram matrices for style image features across specified layers.

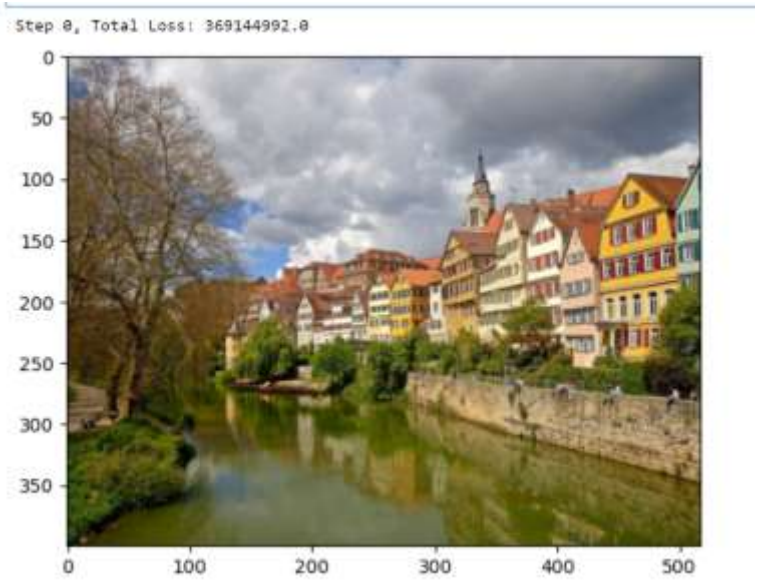
Compute content loss which is the difference in content between the target and content images. Style loss is the difference in style between the target and style images using Gram matrices. Total variation loss which encourages spatial smoothness in the generated image.

~a and ~x be the original image and the image that is generated. Let ~p be the photograph and ~a be the artwork. The loss function we minimise is

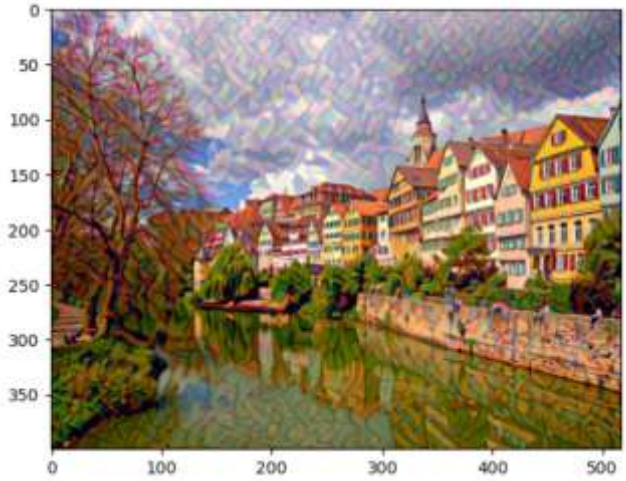
$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

where α and β are the weighting factors for content and style reconstruction respectively.

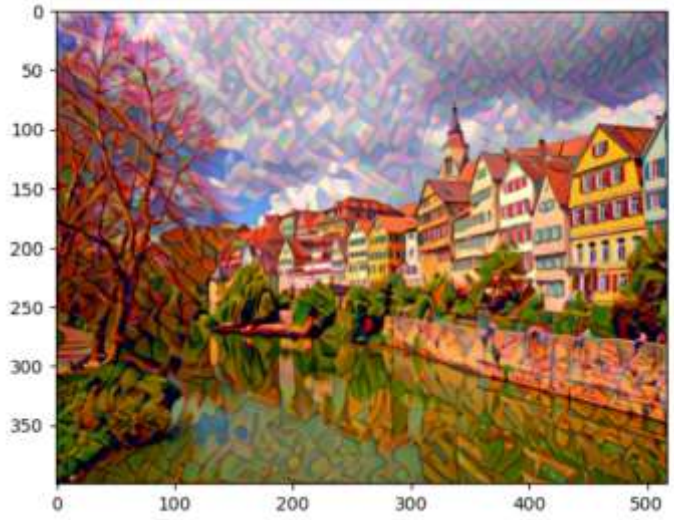
Initialise the target image as the copy of the content image. Use Adam optimiser to minimise the total loss. Adjust the contributions of content and style losses as per need. The optimisation loop iterates through a number of steps to minimize the total loss by adjusting the target image. We used 2000 steps , weights for content and style losses used were 1e3 and 1e-2 respectively .



Step 500, Total Loss: 25724714.0



Step 1000, Total Loss: 11044033.0

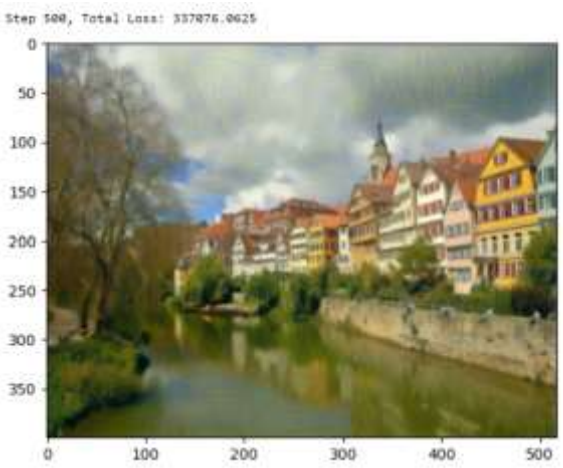


Step 1500, Total Loss: 5219308.0

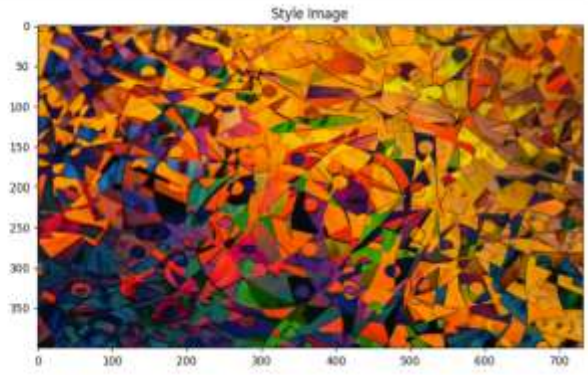
Converts the optimized tensor back to an image format , display the final stylized image and save the final stylized image to a file.

The VGG-19 model is employed in the neural style transfer project primarily for its ability to extract rich hierarchical features from images. These features are essential for computing both the content loss (to ensure the generated image retains the content of the original content image) and the style loss (to imbue the generated image with the style of the style image). By leveraging the pretrained VGG-19 model and its feature extraction capabilities, the project achieves effective separation and synthesis of content and style in the generated images.

Other examples performed –



Output Image



Output Image

It can be further improved by using network with 16 convolutional and 5 pooling layers of the 19 layer VGG Network. Not using any of the fully connected layer. Replace the max-pooling operation by average pooling which improves the gradient flow and a slightly more appealing results.

Conclusion

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

By combining theoretical understanding with practical implementation, this project contributes to the broader field of computer vision and deep learning, offering insights into how artificial intelligence can augment human creativity in visual arts and design.

The successful completion of this project sets a foundation for future research and applications in image synthesis, computational creativity, and interdisciplinary collaborations at the intersection of AI and visual arts.