ARIES SUMMER PROJECT

Neural Style Transfer

#Reference: https://arxiv.org/pdf/1508.06576

-INTRODUCTION

In fine art, especially painting, humans have mastered the skill to create unique visual experiences through composing a complex interplay between the content and style of an image. Thus far the algorithmic basis of this process is unknown and there exists no artificial system with similar capabilities. However, in other key areas of visual perception such as object and face recognition near-human performance were recently demonstrated by a class of biologically inspired vision models called Deep Neural Networks. Here, I tried to implement an artificial system based on a Deep Neural Network that creates artistic images of high perceptual quality. The system uses neural representations to separate and recombine content and style of arbitrary images, providing a neural algorithm for the creation of artistic images. Moreover, in light of the striking similarities between performance-optimised artificial neural networks and biological vision, our work offers a path forward to an algorithmic understanding of how humans create and perceive artistic imagery. When Convolutional Neural Networks are trained on object recognition, they develop a representation of the image that makes object information increasingly explicit along the processing hierarchy.8 Therefore, along the processing hierarchy of the network, the input image is transformed into representations that increasingly care about the actual content of the image compared to its detailed pixel values.

-MODEL SPECIFICATIONS

For this task, I have used the popular pretrained model of VGG Net (#Reference: https://arxiv.org/abs/1409.1556v6).

ARCHITECTURE:

Layers: The network consists of 19 layers, including 16 convolutional layers, 3 fully connected layers, 5 Max Pooling layers, and the SoftMax layer.

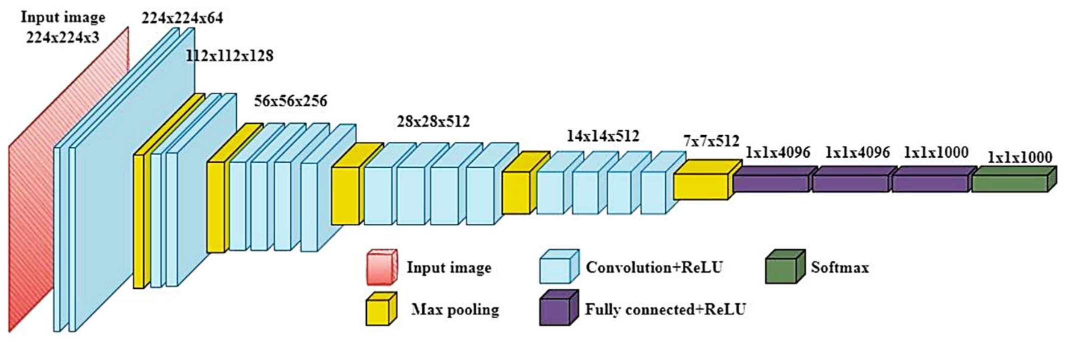
Convolutional Layers: The convolutional layers use filters with a very small receptive field: 3x3 (smallest size to capture the notion of left/right, up/down, centre).
Pooling Layers: Max pooling is performed over a 2x2 pixel window, with a stride of 2.
Fully Connected Layers: There are three fully connected layers. The first two have 4096 channels each, and the third has 1000 channels, one for each class in the ImageNet dataset.

PRETRAINED-MODEL:

Training Data: The pretrained model is trained on the ImageNet dataset, which consists of over 14 million images and 1000 classes.
Transfer Learning: VGG-19 is often used for transfer learning. The pretrained weights can be fine-tuned on a new dataset, which allows the model to leverage the features learned on ImageNet.
Feature Extraction: The lower layers of VGG-19 are good at detecting edges, textures, patterns, and simple shapes, while the deeper layers detect more complex structures and objects. This makes it a robust feature extractor for various image recognition tasks.

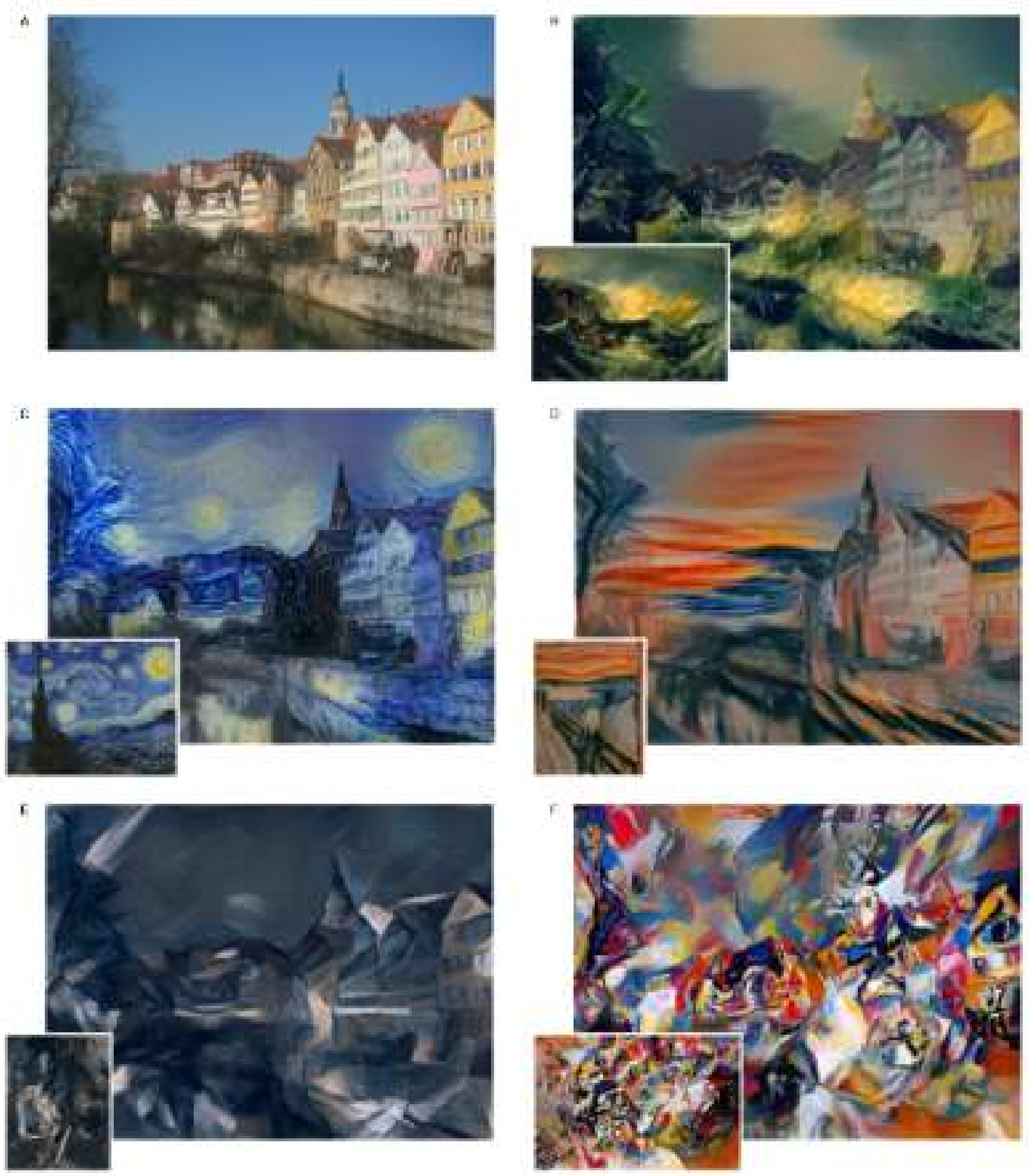
USAGE IN APPLICATIONS:

Object Recognition: The model can classify images into one of the 1000 classes in the ImageNet dataset.
Image Segmentation: By using VGG-19 as a backbone in segmentation models, one can leverage its strong feature extraction capabilities.
Style Transfer: VGG-19 is commonly used in style transfer applications due to its ability to extract detailed features and represent different styles.



VGG-19 is a powerful and versatile pretrained model widely used in computer vision tasks due to its simple yet effective architecture. Its ability to generalize well to various image recognition and processing tasks makes it a valuable tool for both research and practical applications in deep learning.

-NST EXAMPLES



The picture sets above are some examples of this technique, which can be achieved by the model mentioned earlier. I'll be using images of size 256x256 for this project to stylize them to a new unique art. Below are some of the training implementation parts, code snippet explanation, and the results fetched.

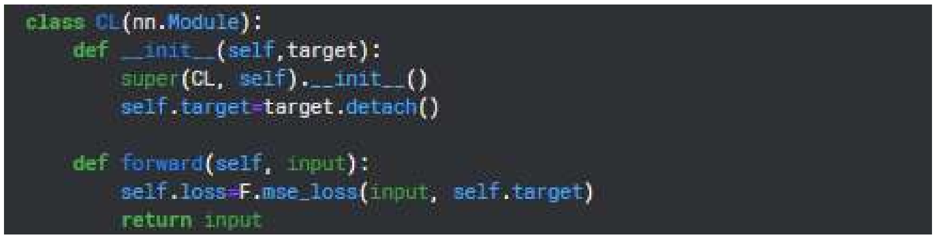
ARIES SUMMER PROJECT

-TRAINING

For NST, first of all, I needed to define all the elements used (style losses, content losses, gram matrix, the parsing functions, etc.) [All mentioned in the notebook].

There are two types of Losses to be minimized during the training phase:

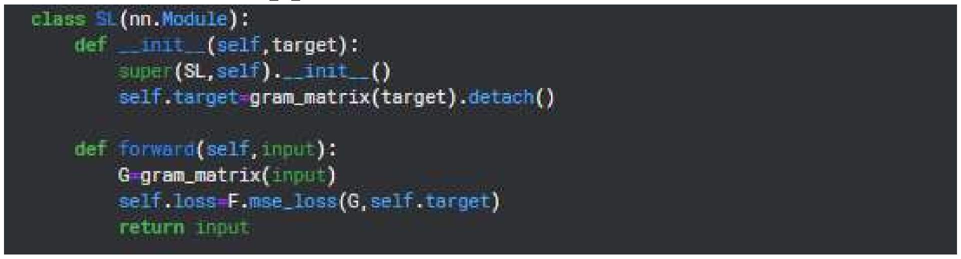
I. **Content Loss**
The content loss in Neural Style Transfer (NST) measures how different the content of the generated image is from the content of the original content image. It helps ensure that the generated image retains the essential elements of the content image. The content loss is typically implemented using the Mean Squared Error (MSE) loss between the feature representations of the content image and the generated image at certain layers of the neural network. Code Snippet:



```
class CL(nn.Module):
    def __init__(self,target):
        super(CL, self).__init__()
        self.target=target.detach()

    def forward(self, input):
        self.loss=F.mse_loss(input, self.target)
        return input
```

II. **Style Loss**
Style loss in Neural Style Transfer measures how different the style of the generated image is from the style of the original style image. This loss ensures that the generated image captures the textures, patterns, and colours of the style image. Style loss is typically computed using the Gram matrix of the feature maps from a convolutional neural network. The Gram matrix represents the correlations between different feature maps, capturing the texture information. Code Snippet:



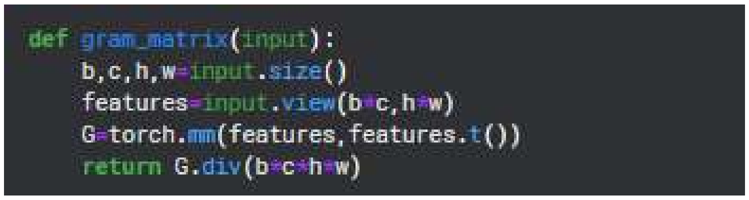
```
class SL(nn.Module):
    def __init__(self,target):
        super(SL,self).__init__()
        self.target=gram_matrix(target).detach()

    def forward(self,input):
        G=gram_matrix(input)
        self.loss=F.mse_loss(G,self.target)
        return input
```

Gram Matrix: The Gram matrix is a matrix of dot products of feature vectors. For a given set of feature maps extracted from an image, the Gram matrix measures the correlations between these feature maps, which effectively captures the texture and style of the image. The Gram matrix captures the style of an image because:

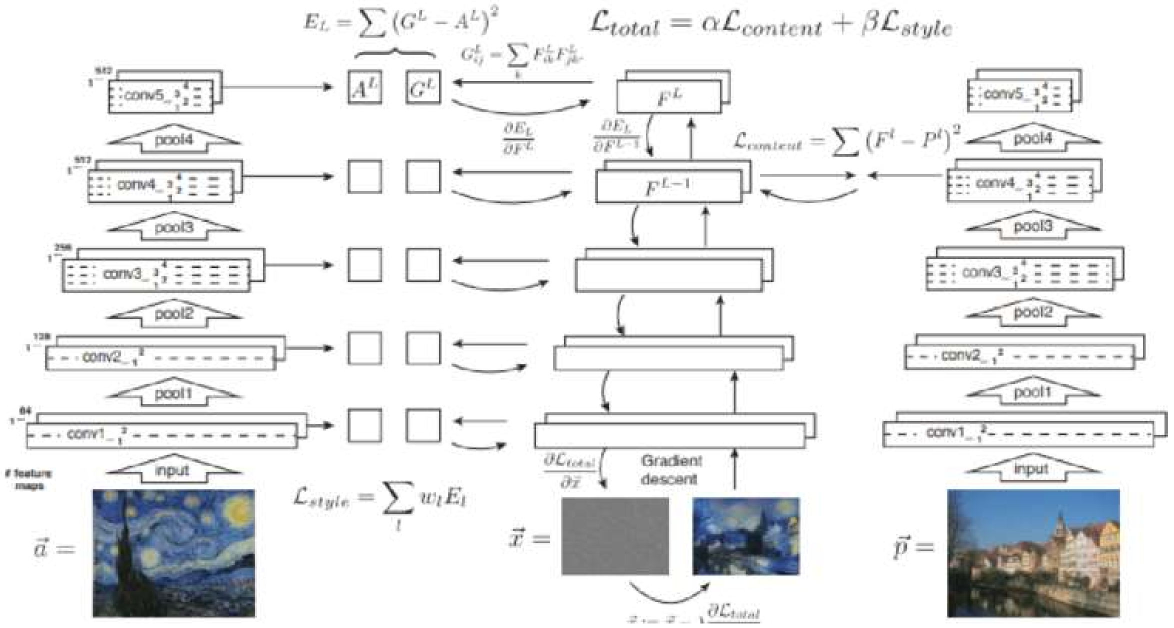
- **Correlation of Features**: It encodes how feature maps (which can be thought of as filters detecting specific patterns) correlate with each other. High values in the Gram matrix indicate strong correlations between specific patterns detected by different filters, which correspond to the textures in the image.
- **Spatial Invariance**: The Gram matrix is invariant to spatial rearrangements of the features, meaning it focuses on the texture and

patterns rather than the exact spatial arrangement, which is essential for capturing the style. The code snippet for the Gram Matrix:



```
def gram_matrix(input):
    b,c,h,w=input.size()
    features=input.view(b*c,h*w)
    G=torch.mm(features,features.t())
    return G.div(b*c*h*w)
```

So as per the core logic behind this task, the researchers have tailored it as mentioned below:



$E_L = \sum (G^L - A^L)^2$
$\mathcal{L}_{total} = \alpha \mathcal{L}_{content} + \beta \mathcal{L}_{style}$
$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}$
A^L
G^L
F^L
F^{L-1}
$\frac{\partial E_L}{\partial F^L}$
$\frac{\partial E_L}{\partial F^{L-1}}$
$\mathcal{L}_{content} = \sum (F^l - P^l)^2$
conv5_1
conv4_1
conv3_1
conv2_1
conv1_1
pool4
pool3
pool2
pool1
input
512
256
128
64
# feature maps
$\mathcal{L}_{style} = \sum_l w_l E_l$
$\frac{\partial \mathcal{L}_{total}}{\partial \vec{x}}$
Gradient descent
$\vec{a} =$
$\vec{x} =$
$\vec{p} =$

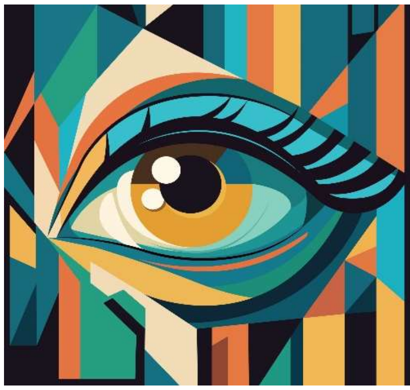
NOTE: Also, we can produce the style from the original, and stylised images via a little tweaking to the present methods. But for this project, we are limited to a single task.

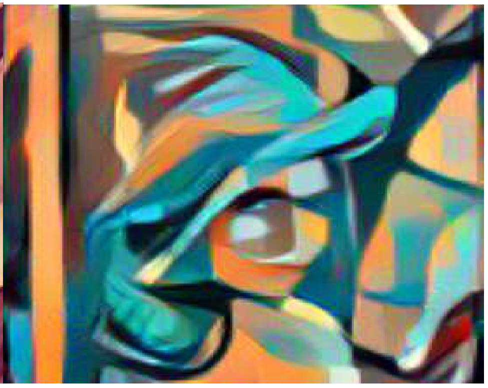
ARIES SUMMER PROJECT

The training for single image is done over 4000 epochs, giving the Content Weight as 1, and Style Weight 100000, using the optimizer LBFGS.

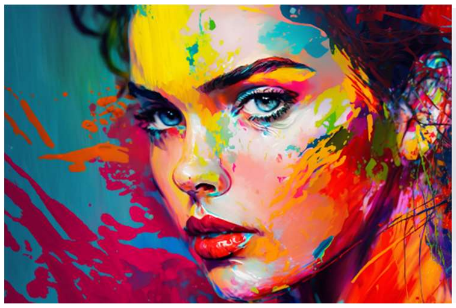
-RESULTS

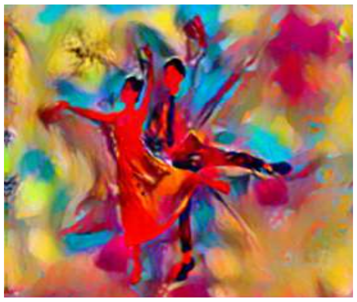
Below are some of the best results that were made by the model:





Style + Original === NST's Styled





Abhivansh Gupta

23112004

Chemical Engineering, Batch 2027