NEURAL STYLE TRANSFER

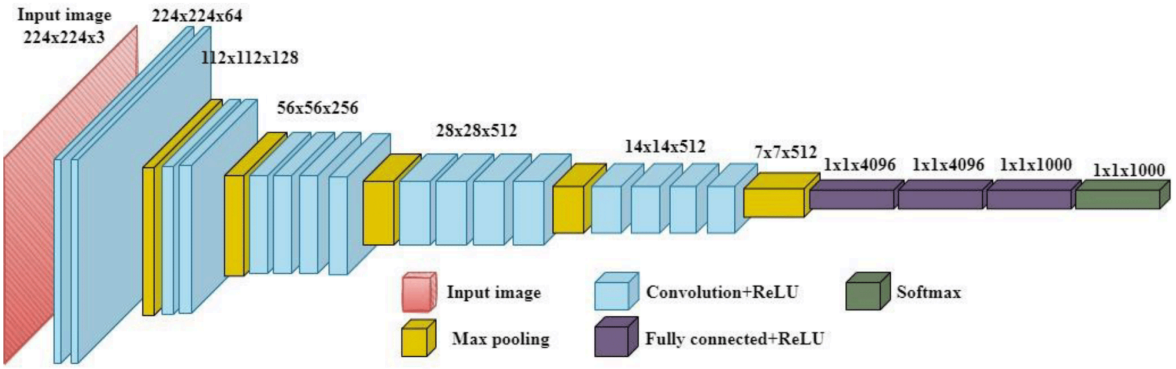
AKSHIT MANOCHA

Enrollment No. - 23112011

The text describes a method for generating images using a VGG-Network, a type of Convolutional Neural Network (CNN) known for its high performance in visual object recognition tasks. Here's a detailed explanation of the key points:

VGG-Network Overview

1. **Architecture**: The VGG-Network used has 19 convolutional layers and 5 pooling layers, but the fully connected layers are not used in this process. The network is available for exploration in the Caffe framework.



We used a pretrained VGG 19 model and removed the softmax and the fully connected layers

Encoding and Content Reconstruction

1. **Layer Function**: Each layer of the network acts as a non-linear filter bank. The complexity of these filters increases with the depth of the layer.

2. **Feature Maps**: An input image is encoded in each layer by the filter responses. For a layer ll with N_l filters, the feature responses are stored in a

matrix F^l of size $N_l \times M_l$, where M_l is the product of the feature map's height and width.

3. **Loss Function for Content Reconstruction**: To recreate the content of an image, a loss function $L_{content}$ defined as the squared error between the feature representations of the original image $\vec{p}$ and the generated image $\vec{x}$ at a specific layer l. This is mathematically represented as:

$$L_{content}(\vec{p},\vec{x},l) = ½ \sum i,j(F_{ij}^l - P_{ij}^l)^2$$

where Fl and Pl are the feature representations of $\vec{x}$ and $\vec{p}$ in layer l, respectively.

4. **Gradient Descent**: The gradient of this loss with respect to the image $\vec{x}$ is computed via backpropagation, allowing the initially random image $\vec{x}$ to be adjusted to match the feature responses of the original image $\vec{p}$.

Style Reconstruction

1. **Gram Matrix**: To capture the style of an image, the correlations between filter responses in a layer are computed using a Gram matrix GlGl. The elements of GlGl are the inner products of vectorized feature maps. It is calculated by first transforming the image into channels,height*width and then performing a matrix multiplication with its transpose

2. **Loss Function for Style Reconstruction**: The style reconstruction loss measures the mean-squared error between the Gram matrices of the original and generated images.

3. **Total Style Loss**: The total style loss L_{style} is a weighted sum of the losses from multiple layers. The weights used for each layer in this model are 2 for the first layer and it decreases all the way to 0.25 for the last convolutional layer.

Combining Content and Style

1. **Total Loss Function**:

   To blend the content of a photograph with the style of a painting, the total loss LtotalLtotal is a combination of the content and style losses:

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

   where α and β control the relative importance of content and style.

2. **Optimization**: By minimizing this total loss using adam optimiser of learning rate 0.003 , the generated image $\vec{x}$ is adjusted to simultaneously match the content of the photograph and the style of the painting.

Specific Results

- **Content Reconstructions**: Different layers (‘conv1_1’,, etc.) were used to reconstruct content from various depths of the network.

- **Style Reconstructions**: Style representations were matched at increasing layers (‘conv1_1’, ‘conv2_1’, etc.) to progressively capture more complex styles.

- **Content and Style Blending**: The content from layer ‘conv4_2’ and styles from multiple layers (‘conv1_1’ to ‘conv5_1’) were combined with specific weightings to create images that mix both attributes.

This method leverages the hierarchical feature extraction capability of CNNs to synthesise images that maintain high-level content while adopting intricate styles from different artworks.

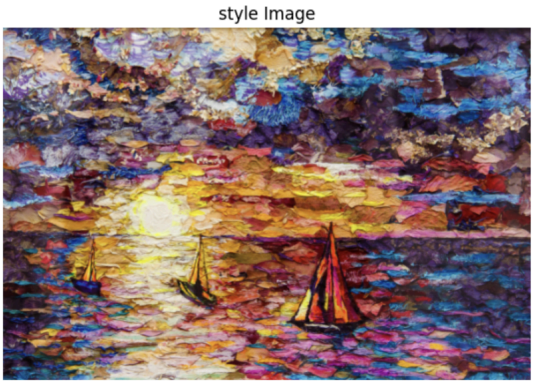
Problem Faced:

While training the model the generated image often came out like this:

However this problem was solved by introducing normalisation while converting the png to tensor.

Training:

In order to get the desired generated image with had to run around 2000 epochs after which the results didn't give noticeable any differences



```
Epoch [1/5000]
```

Stylized Image

```
Epoch [1001/5000]
```

Stylized Image

```
Epoch [2001/5000]
```

Stylized Image

```
Epoch [3001/5000]
```

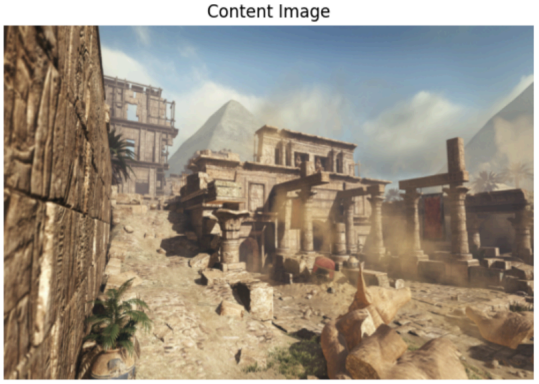
Stylized Image

```
Epoch [4001/5000]
```

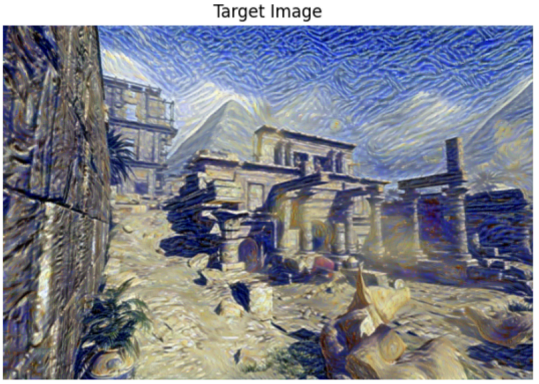
Stylized Image

Other Generated Images:

Content Image



Target Image



Content Image

Target Image