International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 01 | Jan 2021 www.irjet.net p-ISSN: 2395-0072

Style Transfer for Artistic Image Reconstruction using Convolutional Neural Networks

Sravya Vattem[1], Varun Kumar Bejugam[2], Janani Pokkuluri[3]

[1-3]*Student, Dept. of CSE, Keshav Memorial Institute of Technology, Hyderabad, Telangana, India*

Abstract – *Style transfer is an example of image stylization, an image processing and manipulation technique that's been studied for numerous decades within the broader field of non-photorealistic rendering. Style transfer is a popular computer vision technique that aims to transfer visual styles to specific content images. This is implemented by generating the output image such that it preserves some notions of the content image while adapting to certain characteristics of the style image. These characteristics are extracted from the images using a convolutional neural network. In this paper, we aim to implement a loss function that will minimize the distance between the generated image and extracted content and style representations.*

***Key Words*: CNN, style transfer, feature extraction, loss, VGG19, Gram matrix**

1. INTRODUCTION

Style transfer is a computer vision technique that accepts two images- a content image and a style reference image and combines them such that the generated output image retains the core notions of the content image, but appears to be "painted" in the style of the style reference image.

A photograph, for example, would act as the image content, and a Van Gogh painting would be the style reference image. The output result would be a portrait that appears to be painted by Van Gogh himself.

If we think about solving this task via a traditional supervised learning approach, learned style transfer requires a pair of input images- both an original content image and a stylistic representation of that original image. From there, a machine learning model learns the reconstruction and can apply it to new original images.

Unfortunately, this approach is mostly impractical, as these kinds of image pairs rarely exist. In recent years, however, a new approach, Neural Style Transfer (NST), has changed what's possible.

NST employs deep neural networks to power these transformations- neural networks are used to extract feature representations of images related to both content and style so that we can measure how well the style transfer is working without the explicit image pairs. With this new approach, only a single style reference image is needed for the neural network to learn and apply it to original content images.

Neural Style Transfer exploits the learning property of neural networks- in this case, we use convolutional neural networks. The initial layers of the convolutional neural network, or CNN, identify the low-level features such as edges, and the final layers identify the more detailed nuances or the high-level features in an image.

Neural Style Transfer is done by "merging" two images- the "content" image (C) and a "style" image (S), to create a "generated" image (G). This generated image is the final output stylized image.

You can use transfer learning as follows:

1. A straightforward implementation of existing pre-trained models
2. Feature extraction of pre-trained models
3. Modifying the last layer's weights of a pre-trained model

This paper discusses the implementation of the second approach. When we initially start out with big differences in the loss, it is observed that the style transfer is not satisfactory. We can see that styles are transferred, but they seem rough and unintuitive. With each cost minimization iteration, we proceed in the direction of a better merging of the style and content, ultimately resulting in a better-quality image.

The central element for this process is calculating the loss. There is a total of 3 costs that need to be calculated:

1. Content cost
2. Style cost
3. Final cost

Neural Style Transfer is a new, exciting deep learning application. The representations extracted by the series of convolutional layers in a convolutional neural network model can be deconstructed such that content and style can be separated.

This separation forms the Gram matrix, which can be used as a loss function for the CNN. This model can be used to do things such as transfer the artistic style of Vincent Van Gogh into an ordinary photograph.

2. DATA DESCRIPTION

Two different types of data are used to act as the content and style images respectively. Both the content and style images have been sourced from Wikipedia. The content images are

photographs of a landscape and a well-known personality, whereas the style images are famous paintings The Starry Night and the Girl with a Pearl Earring. The objective is to transform the photographs to resemble artworks created in the style of the paintings, seemingly by the same artist.

3. MODULES

I. Data Preprocessing:

a) Reading image:

The raw image is read and converted into an image tensor. Subsequently, the images are transformed to be compatible with the input format specifications of the ImageNet dataset on which the VGG19 network is pre-trained.

b) Resize image:

The convolution neural networks have become very common in recent years in the field of computer vision. But these networks are restricted to the size of the input images. The convolutional neural networks cannot infer on the wide range of the image resolutions, these networks accept only fixed shape of the input images.

This is a big challenge during the data gathering and cleaning phase of every task. The common practice to overcome this issue is to reshape all the images in the dataset to a fixed shape for both training and deployment phases of the convolutional neural network. The raw content images are resized to a maximum of 512 pixels on each axis.

c) Normalization:

Data normalization is an important step that ensures that each input parameter (pixel, in this case) has a similar data distribution. This makes convergence faster while training the network. Data normalization is done by subtracting the mean from each pixel and then dividing the result by the standard deviation.

The distribution of such data would resemble a Gaussian curve centered at zero.

The image pixels are rescaled between -1 and 1 to achieve an even distribution.

II. Style Reconstruction:

Style representations are extracted from the responses of the CNN layers of the VGG19 network that computes the correlations between various filters of the CNN layers. These feature correlations are given by the Gram matrix $G^l \in R^{Nl \times Nl}$, where G^l_{ij} is the inner product between the vectorized feature map i and j in layer l.

The style of an image can be depicted by the correlations across different feature maps. Gram matrix includes this information by taking the outer product of the feature vector with itself at each location, and averaging that outer product over all locations.

After computing the style representations of both the style image and the generated image, the Gram matrices of both images are computed. The sum of the square difference of the Gram matrices or the l^2 norm of the element subtraction of the matrices is minimized. This will eventually lead to the minimization of the difference between the style image and the corresponding generated image.

$$G^l_{ij}(I) = \sum_k A^l_{ik}(I) A^l_{jk}(I).$$

Fig -1: Gram matrix for style reconstruction

4. PROPOSED ALGORITHM

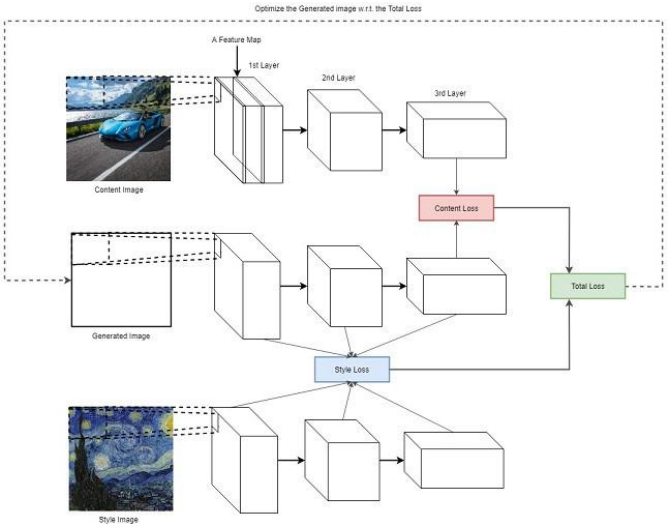

Fig -2: Proposed architecture for Neural Style Transfer

I. Convolutional Neural Network (CNN):

A convolutional neural network is an artificial neural network that is popularly used for analyzing images and although image analysis has been the most widespread use of CNN, it can also be used for other data analysis or classification problems.

Most substantially, we can think about a CNN as something which has a sort of specialization for having the option to choose or identify patterns and figure out them. This pattern recognition is the thing that makes CNN so helpful for image analysis. This is what separates a CNN structure from a

standard multi-layer perceptron or MLP, a CNN has shrouded layers called convolutional layers and these layers are unequivocally what makes a CNN novel.

CNNs also have other non-convolutional layers, however, the premise of a CNN is the convolutional layers. Like some other layers, a convolutional layer gets an input image. This received input image is transformed and then the output yielded from the transformation is contributed to the next following layer.

With each convolutional layer, we have to determine the number of filters the layer ought to have (these filters are really what recognize the pattern in the input image- multiple edges, shapes, textures, objects, etc). So, one type of pattern a filter could detect could be edges, corners, or circles. These simple and kind of geometric filters are what we would see at the start of our network.

The deeper our network goes the more sophisticated these filters become so in later layers rather than edges and simple shapes our filters may be able to detect specific objects like eyes, ears, hair, nose, etc. The training was carried out on the CNN, which is a type of neural network that has been shown to be effective in image processing.

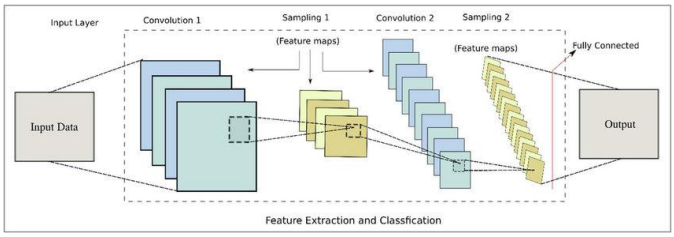

Fig -3: General Architecture of the convolutional neural network

Convolution: The center structure block of CNN is the convolutional layer. Convolution is a numerical activity to join two arrangements of data. For our situation, the convolution is applied to enter information utilizing a convolution filter to produce a feature map.

We perform the convolution cycle by sliding the channel over the given input image. At each position, we do the multiplication of the corresponding element-wise matrix and sum the generated result. This absolute value is applied to the feature map. The zone where the convolution cycle occurs is known as the receptive field.

Further, at this point we move the filter to the right side and perform a similar activity, adding the outcome to the feature map subsequently. We proceed with this throughout the elements of the kernel and finally total the convolution which brings about the feature map.

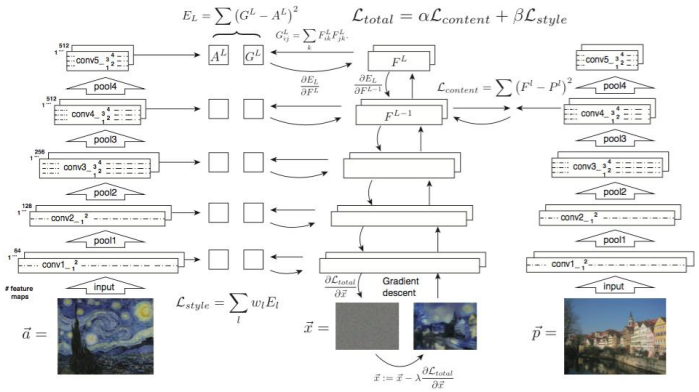

Fig -4: Process flowchart for style transfer

II. VGG19:

VGG19 is a deep convolutional neural network designed to tackle the difficulties faced by the shallow convolutional neural networks in extracting the extremely detailed features in an image. It is a variant of the original VGG network model which, in short, consists of 19 layers (16 convolutional, 3 fully connected layers, 5 max-pooling layers, and 1 SoftMax layer).

VGG is considered to be the successor of the AlexNet model designed by the Visual Geometry Group at Oxford. This deep convolutional neural network is trained on ImageNet which is a generalized dataset consisting of the 14,197,122 images organized according to the WordNet hierarchy.

The intermediate layers of the VGG network are used to extract the style and content representations of the image. The network's input layer is followed by a series of convolutional blocks which extract the hierarchical visual representations of the given input image.

The initial convolutional layers of the VGG19 model extract the basic features or the low-level representation of the images like edges and textures. The final few layers represent higher-level features like wheels or eyes. The intermediate layers of the VGG19 network are leveraged to extract the style of an image and the last convolution blocks are used to extract the content of the image.

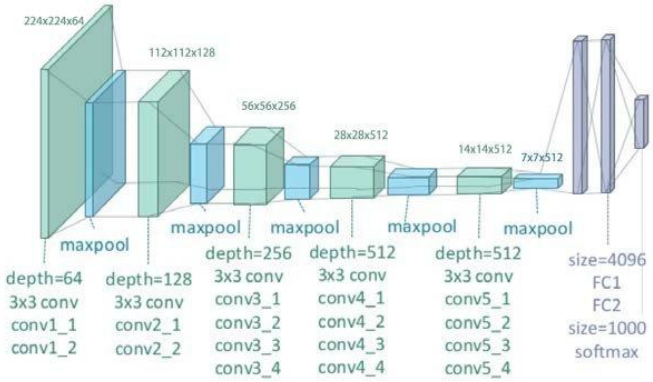

Fig -5: VGG19 network architecture

```
content_layers = ['block5_conv2']

style_layers = ['block1_conv1',
                'block2_conv1',
                'block3_conv1',
                'block4_conv1',
                'block5_conv1']
```

Fig -6: Segregation of content and style layers

III. Loss functions:

In this section, we discuss two loss functions- the content loss function and the style loss function. The content loss function makes sure that the higher layers' activations are similar between the content image and the generated output image.

The style loss function ensures that the correlation of activations in all the layers are similar between the style image and the generated output image. The details are discussed further below.

a) Content cost function:

The content cost function essentially ensures that the content present in the content image is captured in the generated image. It has been observed that CNNs tend to capture details and information about the content in the higher levels, whereas the lower levels are more focused on individual pixel values.

Therefore, we use the top-most CNN layer to define the content loss function. Let $A^l_{ij}(I)$ be the activation of the l^{th} layer, i^{th} feature map and j^{th} position obtained using the image I.

Then the content loss is defined as,

$$L_{content} = \frac{1}{2} \sum_{i,j} (A^l_{ij}(g) - A^l_{ij}(c))^2$$

Essentially $L_{content}$ captures the root mean squared error between the activations produced by the output stylized image and the content image. Further, we discuss why minimizing the difference between the activations of higher layers ensures that the content of the content image is preserved.

Intuition behind content loss:

If we visualize what is learnt by a neural network, we can find evidence that suggests that different feature maps in higher layers of the CNN are activated in the presence of different objects. Hence, if two images are taken such that they have the same content, they should have similar activations in the higher layers.

b) Style loss function:

Delineating the style loss function requires more work. To extract the style information from the VGG19 network, we use all the layers of the CNN. Furthermore, style information is quantified as the amount of correlation between features maps in a given layer.

Next, a loss is calculated as the difference of correlation present between the feature maps computed by the generated output image and the style reference image.

Mathematically, the style loss is defined as,

$$L_{style} = \sum_l w^l L^l_{style} \text{ where,}$$

$$L^l_{style} = \frac{1}{M^l} \sum_{ij} (G^l_{ij}(s) - G^l_{ij}g)^2 \text{ where,}$$

$$G^l_{ij}(I) = \sum_k A^l_{ik}(I) A^l_{jk}(I).$$

w^l is a weight given to each layer while computing the loss and M^l is a hyperparameter that depends on the size of the l^{th} layer.

Intuition behind the style loss

Though the above equation system appears complex, the idea is relatively simple. The goal is to compute a style or Gram matrix for the generated output image and the style reference image. The style loss is then defined as the root mean square difference between the two Gram matrices.

An illustration is provided further that depicts how the style matrix is computed. The style matrix is essentially a Gram matrix, where the $(i,j)^{th}$ element of the style matrix is computed by calculating the element wise multiplication of the i^{th} and j^{th} feature maps and summing across both the width and the height.

In the figure, the red crosses denote element wise multiplication and the red plus signs denote summing across both the width and the height of the feature maps.

c) Final loss:

The final loss is defined as,

$$L = \alpha L_{content} + \beta L_{style}$$

Here, α and β are user-defined hyperparameters. β has absorbed the M^l normalization factor that has been defined earlier. By controlling α and β you can manipulate the amount of content and style injected to the generated output image.

IV. Gram Matrix:

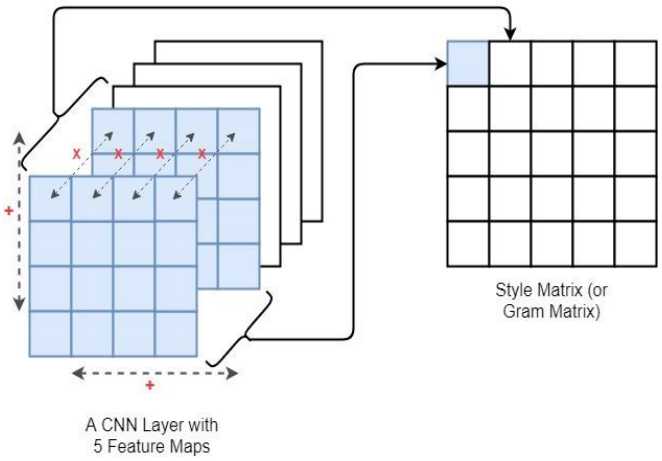

Fig -7: Computation of Gram matrix

Why is it that style is captured in the Gram matrix?

The Gram matrix essentially captures the "distribution of features" of a set of feature maps in a given layer in a CNN. By minimizing the style loss between two images, we are essentially trying to match the distribution of features between both the images.

5. TRAINING

The training and implementation of the model were done using the Deep Learning framework TensorFlow in Python. A helper function is used, which loads both the images in arrays of numbers and reshapes them as required for making them compatible with the model.

Once the images are reshaped, a pre-trained VGG19 model is loaded for extracting the feature representations. The model is only used for extracting features; hence the classifier part of the model is not utilized in this implementation.

A custom VGG19 model is created, composed of content and style layers to help run forward passes on the images and extract the necessary features along the way. The Gram matrix is defined, as well as a custom function used to create a model consisting of content and style layers. This will be used for returning the content and style features from the respective images.

The overall content and style weights and also the weights for each of the style representations are determined, and another function is used, that calculates the gradients of the final loss function defined earlier. These gradients are used to update the generated output image.

Backpropogation: Backpropagation is a process where the neural network learns the parameters using gradient descent algorithms. The gradient descent algorithms compute the gradient of the cost function by using partial differentiation, with respect to the corresponding weights in each layer of the convolutional neural network. The computed gradient is backpropagated using the chain rule of differentiation.

The generated image is initialized with the content image and then backpropagated using the Adam optimizer with the final loss computed in the previous step to generate a blend of both style and content image.

GradientTape provides automatic differentiation, which can calculate the gradients of a function based on its composition. The tf.function decorator is also used to speed up the operations.

6. RESULTS

We implemented the custom VGG19 model on two sets of images to demonstrate style transfer. In the first set of images, the artistic style of the painting The Starry Night is transferred to a landscape photograph of Manhattan, New York.

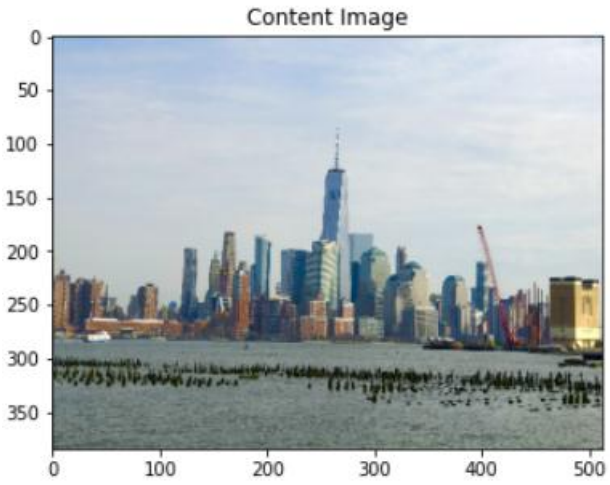

Fig -8: Content image for first example case

Fig -9: Style image for first example case

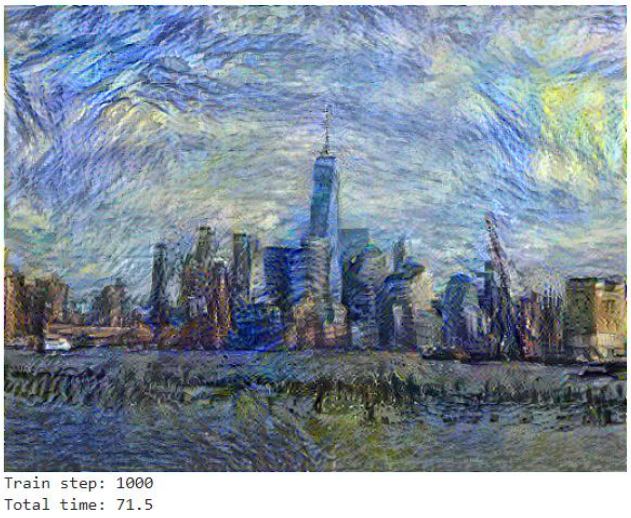

Fig -10: Output stylized image for first example case

The second implementation is done on a human photograph to demonstrate style transfer on diverse images. The artistic style of the painting Girl with a Pearl Earring is transferred to the content image to produce an output image that resembles a painting created by the same artist.

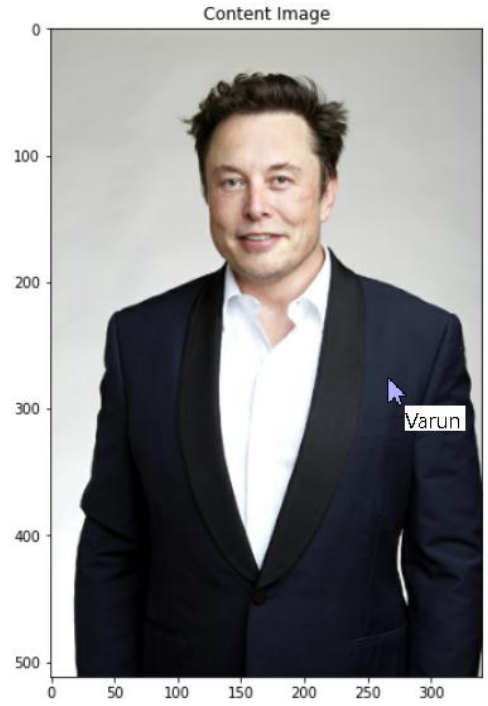

Fig -11: Content image for second example case

Fig -12: Style image for second example case

Fig -13: Output stylized image for second example case

7. CONCLUSIONS

Deep neural networks have already surpassed human level performance in object recognition and detection tasks. However, deep networks still lacked the same efficiency in the case of tasks like generating art that holds high perceptual quality until recent times.

Creating art using Neural Style Transfer allows one to achieve outputs that are comparable to human creations, as well as opens up a new spectrum of possibilities. And with the vast proliferation of deep learning, it has become easier and more accessible to do so.

In addition to empowering people all around the world to experiment with their own creativity, the importance of style transfer can be observed to be playing out in the commercial art world. In recent times, an AI artwork featured at Christie's sold at one of their auctions for more than $430,000.

With the continued advancements of AI-accelerated hardware, both in the cloud and on the edge, style transfer can now be applied to captured and live video. This new capability opens up numerous doors and prospects in design, content generation, and development of creative tools. Given this advancement, we can see how style transfer can be applied in a number of ways:

1. Photo and video editing
2. Artist-Community engagement
3. Commercial art
4. Gaming
5. Virtual Reality

REFERENCES

[1] L. Gatys, A. Ecker, and M. Bethge, "A Neural Algorithm of Artistic Style," J. Vis., 2016, doi: 10.1167/16.12.326.

[2] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song, "Neural Style Transfer: A Review," IEEE Trans. Vis. Comput. Graph., 2020, doi: 10.1109/TVCG.2019.2921336.

[3] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," 2016, doi: 10.1109/CVPR.2016.265.

[4] F. Luan, S. Paris, E. Shechtman, and K. Bala, "Deep photo style transfer," 2017, doi: 10.1109/CVPR.2017.740.

[5] Y. Li, N. Wang, J. Liu, and X. Hou, "Demystifying neural style transfer," 2017, doi: 10.24963/ijcai.2017/310.