KERAS IMPLEMENTATION OF NEURAL STYLE TRANSFER

[1]ABHILASHA SHARMA, [2]RITIK HANDA, [3]RAKSHIT, [4]RAGHAV PUNJANI

[1]Assistant Professor, Department of Computer Science, Delhi Technological University, New Delhi, India
[2,3,4]Student, Department of Computer Science, Delhi Technological University, New Delhi, India

Email: [1]abhi16.sharma@gmail.com, [2]ritikhanda_2k17co271@dtu.ac.in, [3]rakshit_2k17co259@dtu.ac.in, [4]raghavpunjani_2k17co249@dtu.ac.in

Abstract:. This paper uses machine learning programs to transfer everyday images to art style images by using a style image to get the transferred style and a content image we want to transfer the style onto. This project can show how machine learning techniques do well in the use of images and how artificial intelligence can achieve great success in art and creativity. Current style transfer techniques such as Prisma emphasized more on the color and layouts of style image. Hence, the synchronized image might lose color, and the content image's outline gets distorted. This project explores a process that can preserve content images' elements and apply artistic styles from style images. The neural style algorithm is very successful in discovering the style of famous pieces of art. There have been attempts to cover this style in images on various levels of success.

Index terms: Neural Style Transfer, CNN, Deep Neural Network, Keras, Content Loss, Style Loss, feature maps, Gram Matrix, Non-photorealistic rendering, Convolutional Layers, Deep Learning.

I. INTRODUCTION

Painting is a popular art form. For more than Thousand years, people are fascinated by the art of painting with the appearance of many attractive paintings, e.g. Van Gogh's "Starry Night" Previously, re-painting in a different style required a highly trained artist and a lot of time[1].

Since the '90s, the art theories behind the appealing artworks have been attracting the attention of computer science researchers along with the artists. There are a plethora of studies and techniques analyzing how to convert the images into synthetic artworks on their own[1]. Among these studies, the advances in the non-photorealistic rendering are influencing, but now it has become established in the community of computer graphics.

However, most of these NPR style algorithms are designed for different art styles and cannot be easily transferred to other styles. Style transfer is often studied as a common problem of the integration of textures in the computer vision community, which is to extract and transfer text from source to target[1]. Hertzmann et al. re-suggest the framework for making a universal style transfer by reading the same changes from the example provided by a pair of static images and styles called image analogies[1]. However, the common shortcoming of these methods is that they often fail to capture the image structures effectively and use only low-level image features.

Neural-style transfer is a fine-tuning technique that takes a content image which is like a base image that will act as the basic structure of the new generated image and a style reference image which is basically the structure that we will be applying on the base image and finally we will merge the two images together. Output image will look like a content image that is painted in style reference.

Optimized features of the input images are extracted using Convolutional Neural Network.
Here we have to present an installation program based on the Deep Neural Network that creates high-quality art images for comprehension. The system has to use neural presentations for the separation and re-integration of style and content of arbitrary images, giving us an algorithm based upon a neural network for the creation of artistic images. Moreover, due to the striking similarities between the improved performance of neural networks and biological perspectives, our work should provide a way to convey the perception of art by humans and their way of thinking about it.

When learning about object recognition, we have to make sure that the network becomes invariant to all image variation that preserves object identity. Representations that factorize the variation in the content of an image and the variation in its appearance would be efficient for this task. Thus, our ability to abstract content from style and, therefore, our ability to create and enjoy art might be primarily a preeminent signature of the powerful inference capabilities of our visual system.
The rest of the paper is classified as follows: Section II presents related work and section III represents details of list of abbreviations. Section IV presents details of experimental design and section V describes the system architecture while research methodology is explained in section VI. In section VII, we show the experimentation and the final results followed by the conclusion of the work in section VIII.

II. RELATED WORK

MYanghao Li, J. Liu and their co-authors proposed a novel understanding of neural style by regarding it as an area variation issue[2]. They hypothetically showed that coordinating with the Gram matrices of a feature map is identical to limit the Maximum Mean Discrepancy with the second order polynomial bit. They explored different avenues regarding a few other dispersion arrangement techniques, and accomplished engaging outcomes and showed that the essence of neural style transfer is to coordinate with the feature distributions between the style pictures and the created pictures[2].

A. Gupta, Alexandre Alahi, J. Johnson, and Li Fei-Fei described the instability of these techniques by looking at the solution set of the style transfer objective[3]. They showed that the trace of the Gram matrix addressing style is inversely identified related to the stability of the method. they introduced an intermittent convolutional network for ongoing video style transfer which joins a temporal consistency loss and beats the instability of earlier techniques[3].

Gantugs Atarsaikhan, Brian Kenji Iwana and Atsushi Narusawa produced text styles by utilizing Neural Style Transfer[4]. By changing NST, they accomplished neural text style transfer. They likewise exhibited the impacts of utilizing distinctive weighted elements, character positions, and orientations[4].

Leon.Gatys, Aaron Hertzman, M. Bethge, and E. Shechtman bring up an expected problem of the first method: that is the algorithm transfers the colors of the original composition, which can modify the appearance of the scene undesirably. They implemented basic linear methods for transferring style while safeguarding colors[5].

Prateek Verma and Julius O. Smith introduced a strategy for making new sounds regarding it as a style-transfer issue, beginning from an random noise input signal and iteratively utilizing back-propagation to enhance the sound to adjust to filler-outputs from a pre-prepared neural architecture of interest[6].

Automatic motion-transfer transmission can help save animators' time by allowing them to move their own motion settings, which will automatically adjust the use of a variety of characters[7]. D. Holden along with his other co-authors proposed that structure can change the style of movement many times faster than the previous methods used for efficiency [7].

Xun Huang and Serge Belongie introduced a basic yet powerful methodology that interestingly empowers arbitrary style transfer in real-time[8]. Their technique accomplished speed comparable to the quickest existing methodology, without the limitation to a predefined set of styles[8].

Roman Novak and Yaroslav Nikulin examined various ways of improving the Neural algorithm of Artistic Style[9]. They adjusted the style information with the goal for it to catch more data and force a more tight limitation on the style transfer result[9].

Rujie Yin introduced a mindful style transfer algorithm for artistic creations and photographs of the same content utilizing a pre-trained neural network, getting good results over the past work[10]. Moreover, the mathematical analyses show that the style design and the data isn't totally isolated by the neural network[10].

Karen Simonyan and Andrew Zisserman examined the impact of the convolutional network depth on its exactness in the large scale image recognition setting[11]. They assessed the network to expand depths utilizing little (3x3) convolution filters, which showed an enhancement from the earlier art configurations and can be accomplished by pushing the depth to 16-19 weight layers. Their two best-performing ConvNet models are openly accessible to all for research on deep visual representation in computer vision[11].

III. ABBREVIATIONS USED

All the abbreviations that have been used in the paper are mentioned below:

TABLE I. Abbreviation table

NPR	Non-Photorealistic Rendering
NST	Neural Style Transfer
CNN	Convolutional Neural Network
VGG	Visual Geometry Group
GM	Gram Matrix
ML	Machine Learning
SGD	Stochastic Gradient Descent
Itn	Iteration

IV. EXPERIMENTAL DESIGN AND SETUP

For this project, we utilized the features that are used in the 19 layers of VGG Network, which comprises convolutional and pooling layers, and fully connected-layers[12]. You can load a pre-trained form of the network trained on in excess of 1,000,000 pictures from the ImageNet[13] database. The Pretrained network can differentiate pictures into 1000 item classes, like console, mouse pencil, and numerous creatures. Along these lines, the network has learned rich feature representation over a wide scope of images.[14]

V. SYSTEM ARCHITECTURE

.

As expressed before, neural style transfer utilizes a pre trained convolutional neural network[15]. At that point to describe a loss function which mixes two pictures flawlessly to make outwardly engaging art, Neural STyle Transfer describes inputs in the following ways:

- content image (c) — the base picture in which we will transfer the style[15].
- style image (s) — the image from which we will obtain the desired style[15].
- generated image (g) — the picture that will contain the final image generated after we have transferred the style from style image into the content image[15].

The architecture of our model is shown in figure 1. We will explain the different components in the next section. The thought is to give a general comprehension of the work process occurring during style transfer[15].

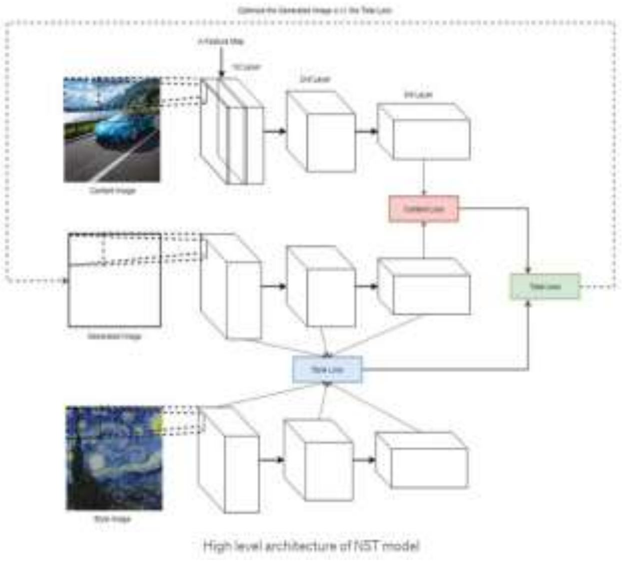


Figure 1: Architecture of NST model
Source: Adapted from [15]

VI. PROPOSED METHODOLOGY

We will use pre trained networks of convolutional networks. CNN is a category of deep neural networks, often used to analyze visual images. It is a part of artificial neural networks that has a commanding method in computer vision tasks since the astounding results that were shared on the object recognition competition known as the ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2012. Expert-level performances in various fields have been achieved by CNN.

This is the typical structure of a Convolutional Neural Network:

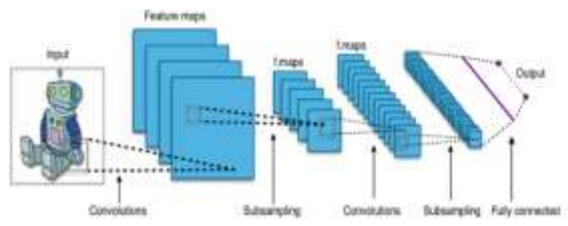
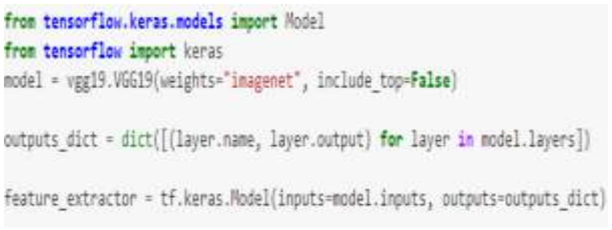


Figure 2: Structure of CNN
Source: Adapted from [16]

To minimize the work, we must use libraries like Keras and imply it on our problems. It is used for transfer learning and we can load the model as shown below.

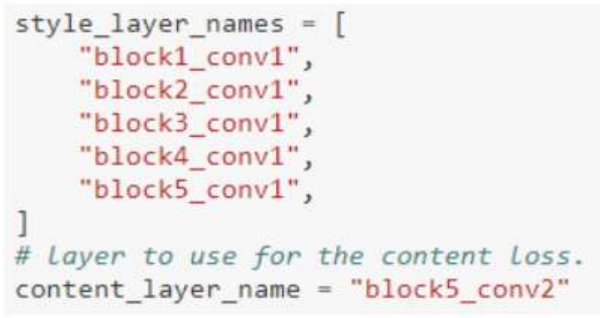


```
from tensorflow.keras.models import Model
from tensorflow import keras
model = vgg19.VGG19(weights="imagenet", include_top=False)

outputs_dict = dict([(layer.name, layer.output) for layer in model.layers])

feature_extractor = tf.keras.Model(inputs=model.inputs, outputs=outputs_dict)
```

Figure 3: Code for loading VGG19
Source: Adapted from [17]

Initially we import the libraries. Then the vgg19 model is downloaded where include_top=False shows that we do not want the last layer of softmax, which is the output layer used to divide 1000 classes in this competition.

Finally a dictionary is created which stores key as the name of layer and value as the layer outputs. We then defined our model by including it as an indication of input and VGG results as a dictionary for each layer. Next, we will describe the layers from which we will extract content elements and styles.



```
style_layer_names = [
    "block1_conv1",
    "block2_conv1",
    "block3_conv1",
    "block4_conv1",
    "block5_conv1",
]
# Layer to use for the content loss.
content_layer_name = "block5_conv2"
```

Figure 4: Content and Style Layers
Source: Adapted from [17]

We already have a dictionary where we can lay out these layers and extract the results.

A. Loss Functions

Finding the image you want will define the loss function that will help us to get the desired result. per pixel losses concept will be applied here.

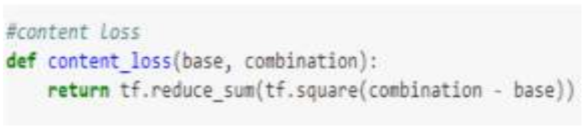
Per Pixel Loss is a measuring unit used to understand the difference between images at pixel level. It compares pixel output values with input values. Sometimes the loss of pixels has its problems in terms of representing all logical features. This is where the permanent loss comes into play[17]. following are the two main loss terms-

1.Content Loss

2.Style Loss

B. Content Loss

It ensures that the content we are looking for in our result is well received. It has been shown that each pixel value is being focused at lower levels while CNN captures information about content at high levels of the network.



```
#content loss
def content_loss(base, combination):
    return tf.reduce_sum(tf.square(combination - base))
```

Figure 5: Content Loss Function
Source: Adapted from [17]

Here the features of the content are basic while the features of the output image that are produced are composite. Here reduce_sum includes the sum of objects in all the specified parameters, in this case, the corresponding pixel difference between the input and the generated image[17].

C. Style Loss

Many layers are involved in calculating the loss function for style which makes it difficult to define it as compared to content. Style details are measured as the number of intersections available between feature maps for each layer. Here we apply the GM to compute the loss of style.

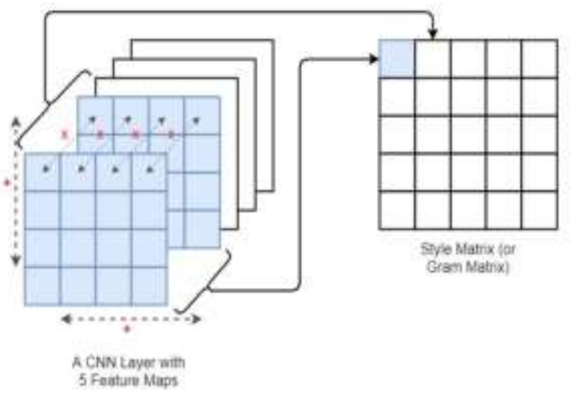
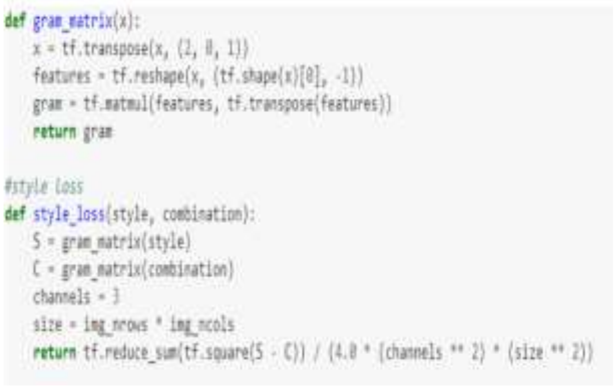

Style Matrix (or Gram Matrix)
A CNN Layer with 5 Feature Maps

Figure 6: Computing Style Matrix for a CNN Layer with 5 feature maps
Source: Adapted from [15]

So now we have to do the style matrices and the result matrices using GM and calculate the difference between them. Gij is the product of the ith and jth layer map and is summarized in height and width as shown above[17].



```
def gram_matrix(x):
    x = tf.transpose(x, (2, 0, 1))
    features = tf.reshape(x, (tf.shape(x)[0], -1))
    gram = tf.matmul(features, tf.transpose(features))
    return gram

#style loss
def style_loss(style, combination):
    S = gram_matrix(style)
    C = gram_matrix(combination)
    channels = 3
    size = img_nrows * img_ncols
    return tf.reduce_sum(tf.square(S - C)) / (4.0 * (channels ** 2) * (size ** 2))
```

Figure 7: Style Loss Function
Source: Adapted from [17]

D. Final Loss

Now both the loss functions are computed. We will have to compute a weighted summation of both the style losses and computed content to calculate the final loss.The final loss is defined as,

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

Figure 8: Total Loss Function
Source: Adapted From [15]

where α and β are user-defined hyperparameters[15]. We can control the percentage of content and style image being incorporated into the desired image by carrying these hyperparameters[15].

The code above is the final combination of losses by

crossing the layers and taking a weighted summary from the last second line to calculate the final loss. Finally, we will have to define an optimizer (Adam or SGD) that will optimize network losses.
Finally we will have a working Neural Style Transfer System and we will be able to get the desired artistic image using content image and the style image.

VII. EXPERIMENTAL RESULTS

Here, we will show the first 10 iterations of neural style transfer for different inputs to demonstrate how the style image gets blended into the content image more smoothly with each iteration.

INPUT 1

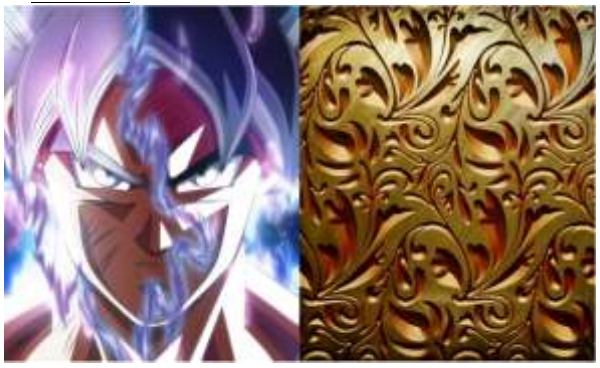


Figure 9: 1st Input Content and Style images

GENERATED IMAGES FOR EACH ITERATION

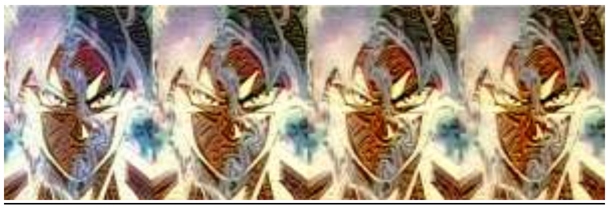


Figure 10: Generated images of 1st to 4th iterations

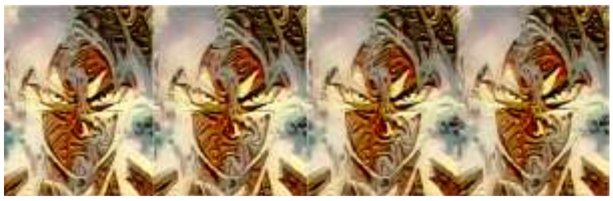


Figure 11: Generated images of 5th to 8th iterations

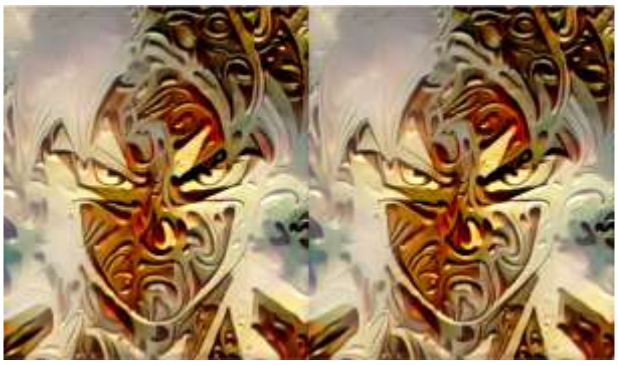


Figure 12: Generated images of 9th to 10th iterations

From the above results, we can clearly see that with each iteration, the image is getting better and smoother. From this we can say that the loss value is decreasing with each iteration. We can also see that, for the first few iterations, the change in the generated image seems to be drastic and after that the change becomes very gradual.

INPUT 2

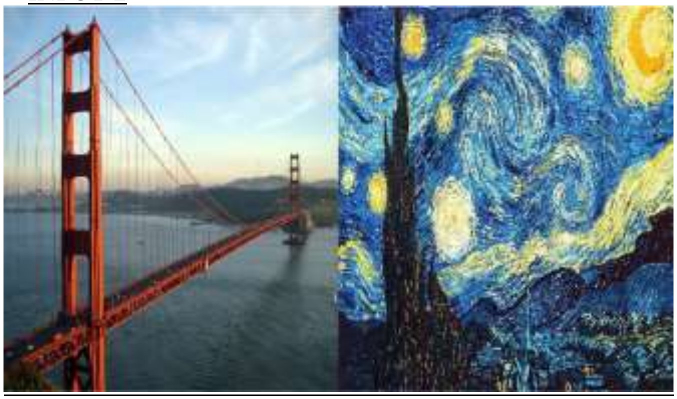


Figure 13: 2nd Input Content and Style images

GENERATED IMAGES FOR EACH ITERATION

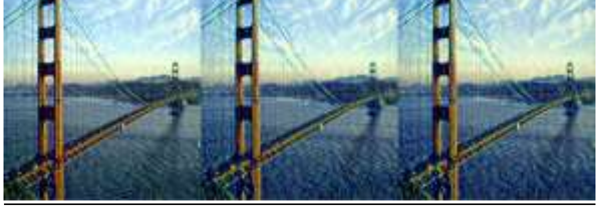
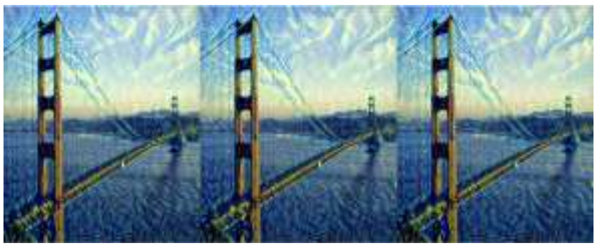
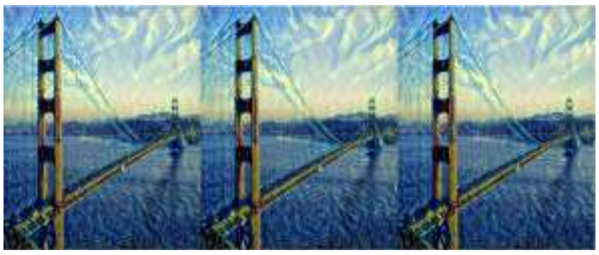


Figure 14: Generated images of 1st to 3rd iterations



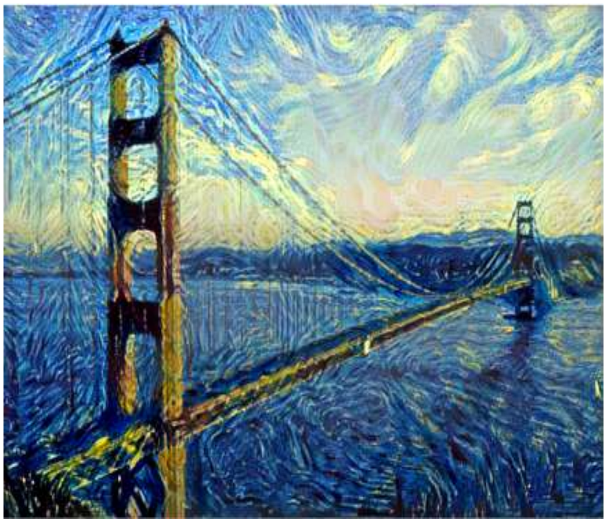
Itn-4 Itn-5 Itn-6

Figure 15: Generated images of 4th to 6th iterations



Itn-7 Itn-8 Itn-9

Figure 16: Generated images of 7th to 9th iterations



Itn-10

Figure 17: Generated image of 10th iteration

For the ease of the reader, we have displayed the total current loss value and the improvement percentage of optimizing the desired result image for some particular input images in the form of tables and bar graphs as shown below. In each table we will show the respected values corresponding to the first 10 iterations.

TABLE II. Iteration - Current Loss Value table

ITERATION	CURRENT LOSS VALUE
Itn-1	298951650.0
Itn-2	113869330.0
Itn-3	66625684.0
Itn-4	45801948.0
Itn-5	35806490.0
Itn-6	30146176.0
Itn-7	25975218.0
Itn-8	23655852.0
Itn-9	20967332.0
Itn-10	19168550.0

From the above Table-1, we can see that the loss value is decreasing with each iteration and we have also shown this decrease in a bar graph(GRAPH-1) shown below.

GRAPH-1

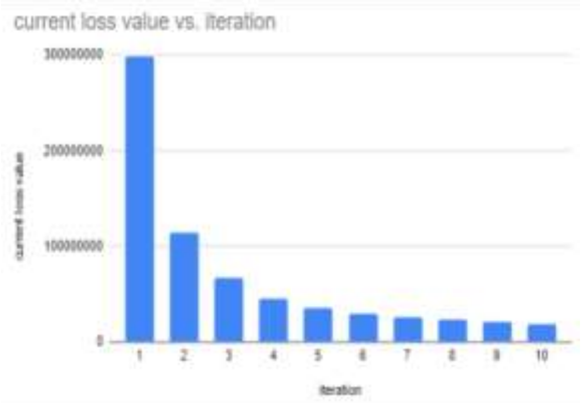


Figure 18: Current Loss Value Vs Iteration Graph

We can see that initially the drop in loss value is very huge but it becomes gradual after a few iterations matching with the trend of generated image results that we have shown above.

TABLE III. Iteration - Improvement percentage

ITERATION	IMPROVEMENT %AGE
Itn-1	0.000%
Itn-2	61.910%
Itn-3	41.489%
Itn-4	31.255%
Itn-5	21.823%
Itn-6	15.808%
Itn-7	13.835%
Itn-8	8.929%
Itn-9	11.365%
Itn-10	8.579%

From the above Table-2, similar to Table-1, we can see that the improvement percentage is decreasing with each iteration and we have also shown this decrease in a bar graph(GRAPH-2) shown below.

GRAPH-2

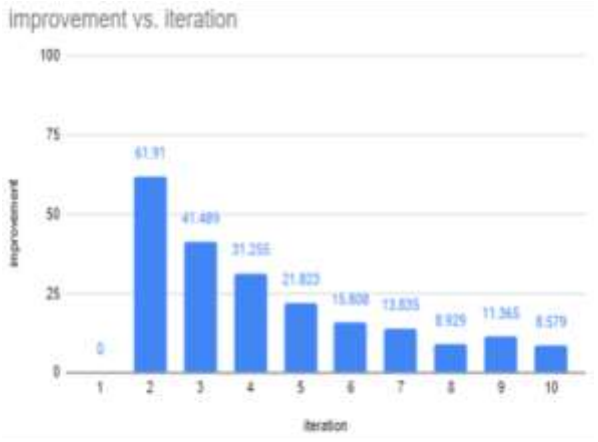


Figure 19: Improvement Percentage Vs Iteration Graph

We can see that initially the drop in improvement percentage is very huge similar to loss value but it becomes gradual after a few iterations matching with the trend of generated image results that we have shown above.
After testing different images and analyzing from what we have seen, we recommend choosing style images with simple outlines and large blocks of colorful blushes to convey artistic style and avoid using paintings with multiple small color blocks. Our implementation of the CNN model can effectively detect the transfer of art style.

VIII. CONCLUSION

In this project, we implemented Neural Style Transfer that allows us to merge two images to create a new artistic image. Initially we learned about NST and its architecture. After that, we described the details of the neural style transmission network and TensorFlow along with loss functions, VGG Network. We also talked about the two main losses which helped us in getting the desired results; loss of content and loss of style in detail, and they see how they come together to explain the final loss. Eventually we were able to use our NST model and saw the art made by this model.
Due to the technical challenges and the increasing demand of industry, Neural Style Transfer has become an interesting field to do research. A large amount of research has been done on NST. It is a very rapid growing technique and still there is a lot of work that can be done on it.
NST as a field of research is still immature despite the remarkable progress that has been made in these few past years. Right now, we have to make the current algorithms of NST more efficient so that we can incorporate multiple styles more accurately. For this we must improve the quality of the generated image in a large dataset of input content and style images and we must avoid bad results. We have seen that NST works with different efficiency with different artistic styles. Like it generates good results when working with arbitrary irregular styles but cannot generate good results with regular kinds of styles. Distorted results are produced because of image reconstruction by CNN. Previous papers have used natural environmental images as content images to show their results but these techniques do not usually work with abstract images because pre-trained classifiers are not able to extract proper features of these types of images. An additional practice of the NST is not only to imitate man-made art by NST techniques but rather to create a new form of art created by AI under the guidance of the basic principles of art[1].

REFERENCES

[1] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu & Mingli Song, Neural Style Transfer: A Review. In IEEE Transactions on Visualization and Computer Graphics, vol. 26, no. 11, pp. 3365-3385, 1 Nov. 2020, doi: 10.1109/TVCG.2019.2921336.
[2] Yanghao Li., Naiyan Wang, Jiaying Liu & Xiaodi Hou, Demystifying neural style transfer. arXiv preprint arXiv:1701.01036, 2017.
[3] Agrim Gupta, Justin Johnson, Alexandre Alahi & Li Fei-Fei, Characterizing and improving stability in neural style transfer. In Proceedings of the IEEE International Conference on Computer Vision (pp. 4067-4076), 2017.
[4] Gantugs Atarsaikhan, Brian Kenji Iwana, Atsushi Narusawa, Keiji Yanai & Seiichi Uchida, Neural font style transfer. In 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR) (Vol. 5, pp. 51-56). IEEE, November, 2017.
[5] Leon A. Gatys, Matthias Bethge, Aaron Hertzmann & Eli Shechtman, Preserving color in neural artistic style transfer. arXiv preprint arXiv:1606.05897, 2016.

[6] Prateek Verma & Julius O.Smith, Neural style transfer for audio spectograms. In 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA. arXiv preprint arXiv:1801.01589, 2018.

[7] Daniel Holden, Ikhsanul, Ikuo Kusajima & Taku Komur, Fast neural style transfer for motion data. IEEE computer graphics and applications, 37(4), 42-49, 2017.

[8] Xun Huang & Serge Belongie, Arbitrary style transfer in real-time with adaptive instance normalization. In Proceedings of the IEEE International Conference on Computer Vision (pp. 1501-1510). 2017.

[9] Roman Novak & Yaroslav Nikulin, Improving the neural algorithm of artistic style. arXiv preprint arXiv:1605.04603, 2016.

[10] Rujie Yin, Content aware neural style transfer. arXiv preprint arXiv:1601.04568, 2016.

[11] Karen Simonyan & Andrew Zisserman, Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[12] Elvin Mirzazada, Neural Style Transfer with Deep VGG Model. May 14, 2020. Available: https://medium.com/@mirzezadeh.elvin/neural-style-transfer-with-deep-vgg-model

[13] ImageNet, Available: http://www.image-net.org.

[14] VGG19 convolutional neural networks. Available: https://www.mathworks.com/help/deeplearning/ref/vgg19

[15] Thushan Ganegedara, Intuitive Guide to Neural Style Transfer. Jan 9, 2019. Available: https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-neural-style-transfer-ef88e46697ee3

[16] Vikas Solegaonkar, Convolutional Neural Network. Feb 18, 2019. Available: https://towardsdatascience.com/convolutional-neural-networks-e5a6745b2810

[17] Introduction And Implementation to Neural Style Transfer - Deep Learning. Oct 22, 2020. Available: https://www.analyticsvidhya.com/blog/2020/10/introduction-and-implementation-to-neural-style-transfer-deep-learning/

★★★