**CREATING ARTISTIC IMAGE FROM VGG19**

**NEURAL NETWORK**

NAME-YASH DIXIT

BATCH-F2

ENROLMENT NO-9919103055

**Under the guidance of Dr. Neeraj Jain**



# 

**BIG DATA ANALYTICS**

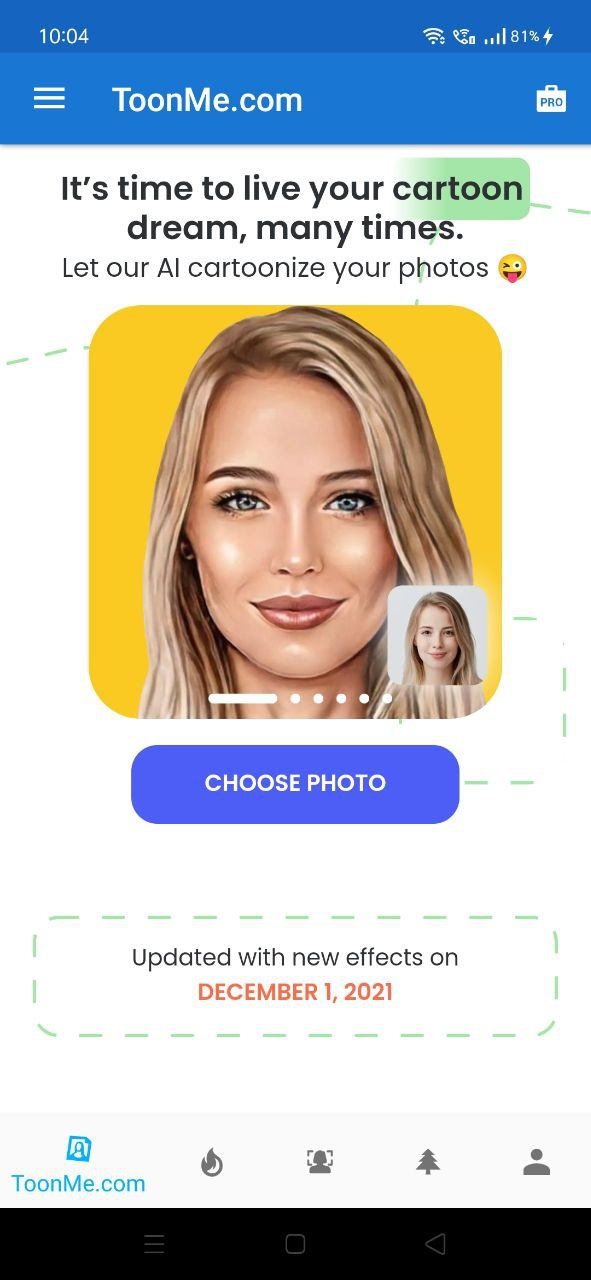
**Jaypee Institute of Information Technology University, Noida**

**ODD SEM 2021**

**PROBLEM STATEMENT**

We have seen plenty of apps nowadays which take the original image and convert it into some other form using some predefined filters , less people know about the actual procedure of how it happens and what makes images look in different forms.

Recently I was using this App form the photo lab and i got fascinated about how it actually happens



Therefore after using the prior knowledge of machine learning and deep learning neural networks I tried to adopt the same mechanism and perform a similar styling of any image using

My own procedure.

**INTRODUCTION**

This is a basic representation of how a neural network can be used to give a style to an image and thereby apply filters to your image.In this Kernel I have explained part by part how a basic Neural style transfer is actually constructed. I have gone through various works on Neural Style Transfer and how they could be constructed.

Style transfer is the technique of reconstructing images in the style of another image.

There are various research papers on this topic that how neural networks could be used to generate artistic style images, each author stating their own innovative way of optimization and loss function to create a artistic image generating neural network.

**FOR THIS PURPOSE I HAVE USED THE VGG19 NEURAL NETWORK**

VGG-19 is a convolutional neural network that is 19 layers deep. You can load a pre-trained version of the network trained on more than a million images from the ImageNet database .

The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.

It is a model, with weights pre-trained on ImageNet.ImageNet, is a dataset of over 15 millions labeled high-resolution images with around 22,000 categories. ILSVRC uses a subset of ImageNet of around 1000 images in each of 1000 categories. In all, there are roughly 1.3 million training images, 50,000 validation images and 100,000 testing images.



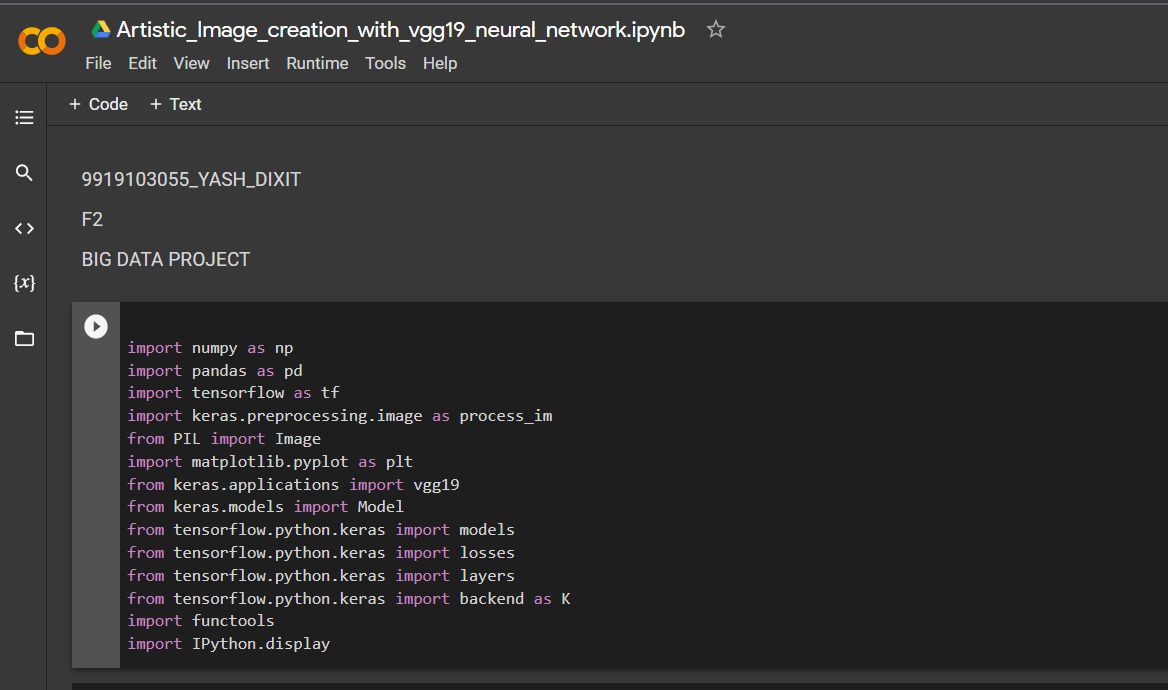
**Tools Used**

1. **GOOGLE’S VGG19**

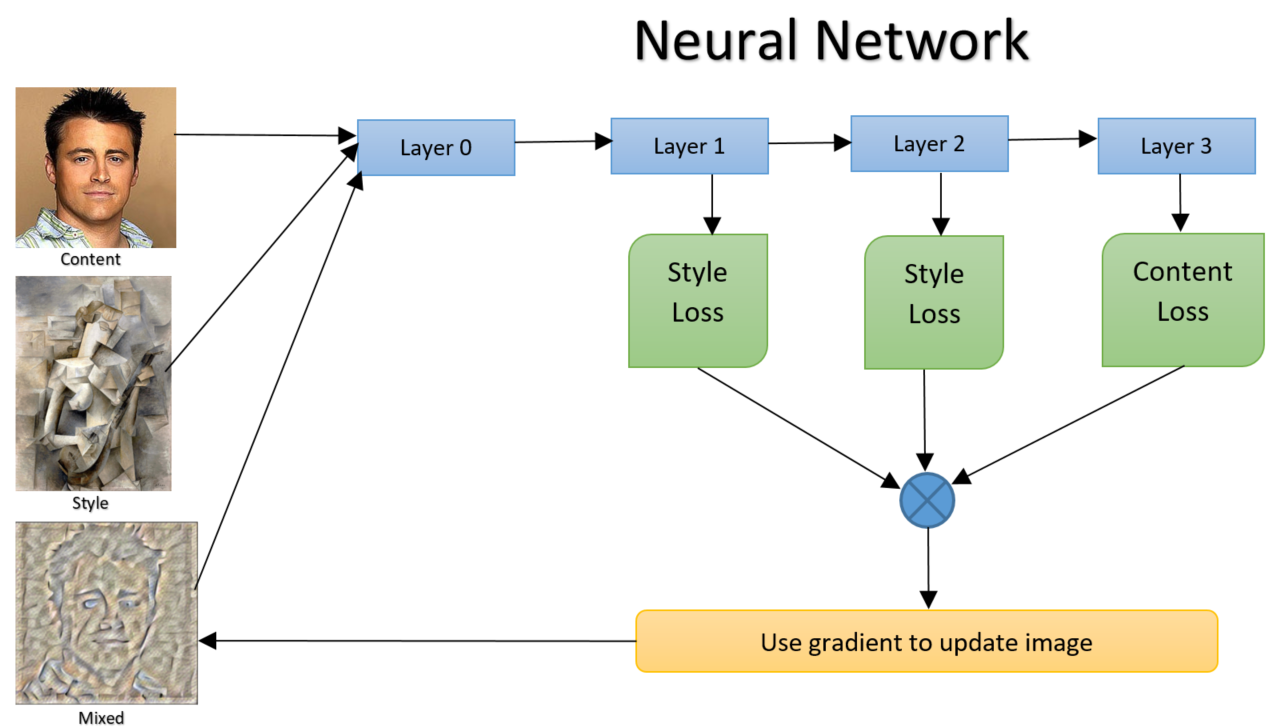
* VGG is a convolutional neural network that has a depth of 19 layers. It was built and trained by K. Simonyan and A. Zisserman at the University of Oxford in 2014. You can access all the information from their paper, Very Deep Convolutional Networks for Large-Scale Image Recognition, published in 2015. The VGG-19 network is trained using more than 1 million images from the ImageNet database. It was trained on 224x224 pixels colored images. Naturally, you can import the model with the ImageNet trained weights. This pre-trained network can classify up to 1000 objects. In this tutorial, we will get rid of the top part used for classification and add our own additional layers so that it can be repurposed for neural style transfer

1. **TENSORFLOW**
2. **PANDAS**
3. **NUMPY**
4. **IMAGE TOOL FORM PIL LIBRARY**
5. **KERAS**

* **LOSSED**
* **LAYERS**
* **PROCESSING IMAGE**
* **MODELS**

****

**PROPOSED WORK AND WORKFLOW**

****

**Preprocessing the image**

This involves first storing the image first as the pil object changing the size of the image and resizing the style and the content image in the same style and then processing the image to array using the keras process\_img so that it gets converted into a keras object.

****

**CONTENT LOSS**

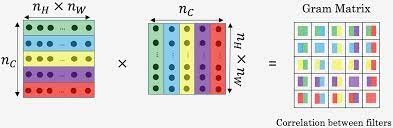
Our content loss definition is actually quite simple. We’ll pass the network both the desired content image and our base input image. This will return the intermediate layer outputs (from the layers defined above) from our model. Then we simply take the euclidean distance between the two intermediate representations of those images.

**STYLE LOSS**

Computing style loss is a bit more involved, but follows the same principle, this time feeding our network the base input image and the style image. However, instead of comparing the raw intermediate outputs of the base input image and the style image, we instead compare the Gram matrices of the two outputs.

Mathematically, we describe the style loss of the base input image, x, and the style image, a, as the distance between the style representation (the gram matrices) of these images. We describe the style representation of an image as the correlation between different filter responses given by the Gram matrix Gˡ, where Gˡᵢⱼ is the inner product between the vectorized feature map i and j in layer l. We can see that Gˡᵢⱼ generated over the feature map for a given image represents the correlation between feature maps i and j.

**GRAM MATRIX**

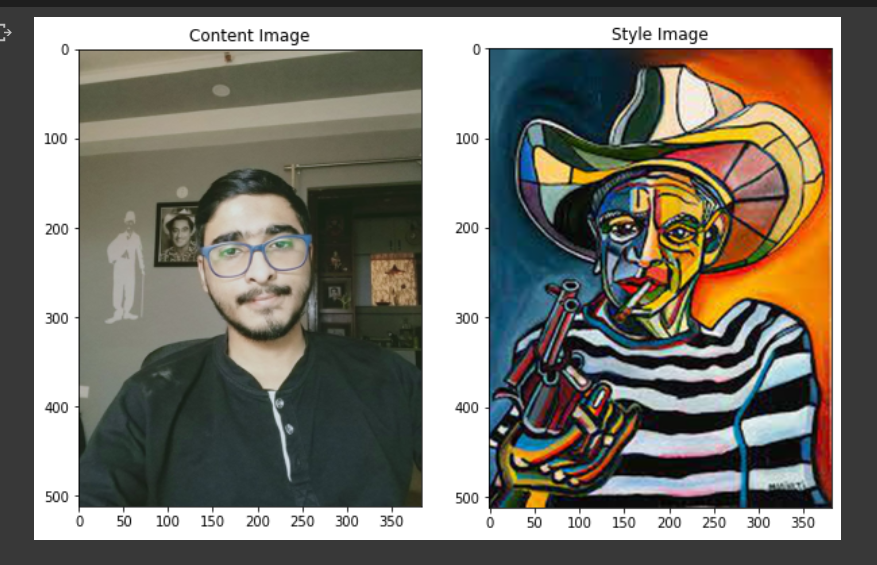


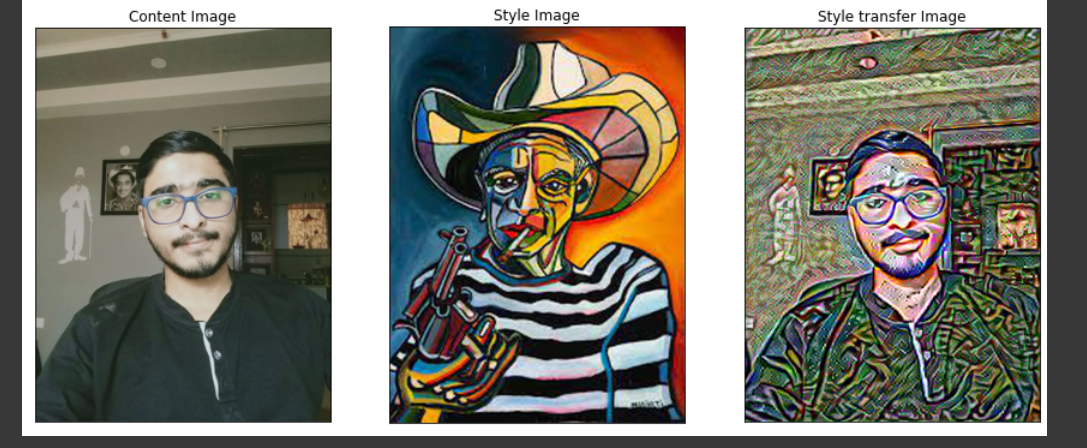
**RUNNING THE GRADIENT DESCENT**

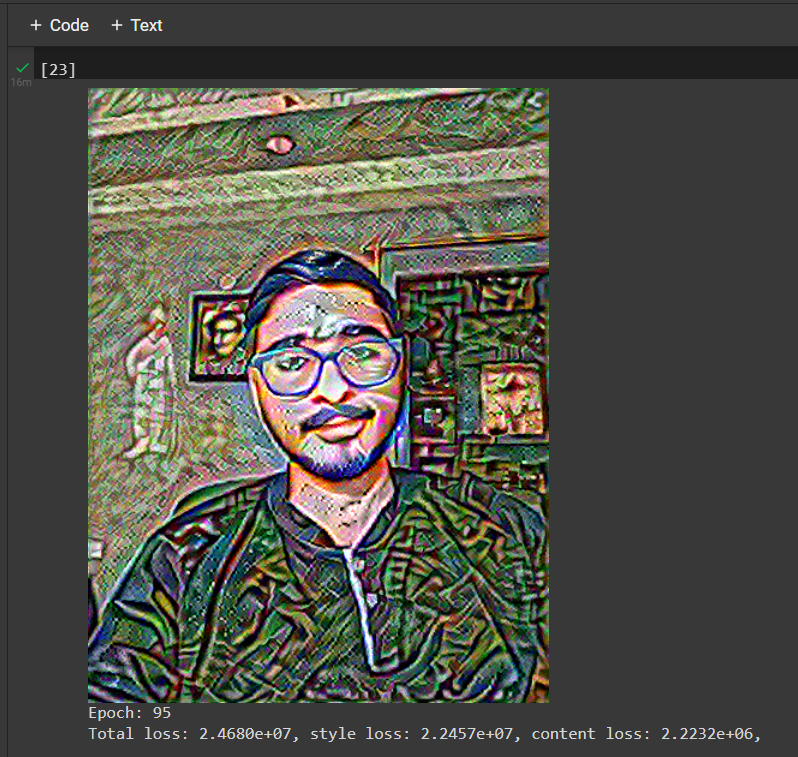
Now that we can output predictions for style and content information, it is time to configure our model’s optimizer with Adam and create a custom loss function:

Now we will define a custom train\_step function where we take advantage of GradientTape, which allows us to do automatic differentiation for loss calculation. GradientTape records the operations during the forward pass and then can compute the gradient of our loss function for our input image for the backward pass. Note that we use tf.function() decorator so that TensorFlow knows that we are passing a train\_step function.

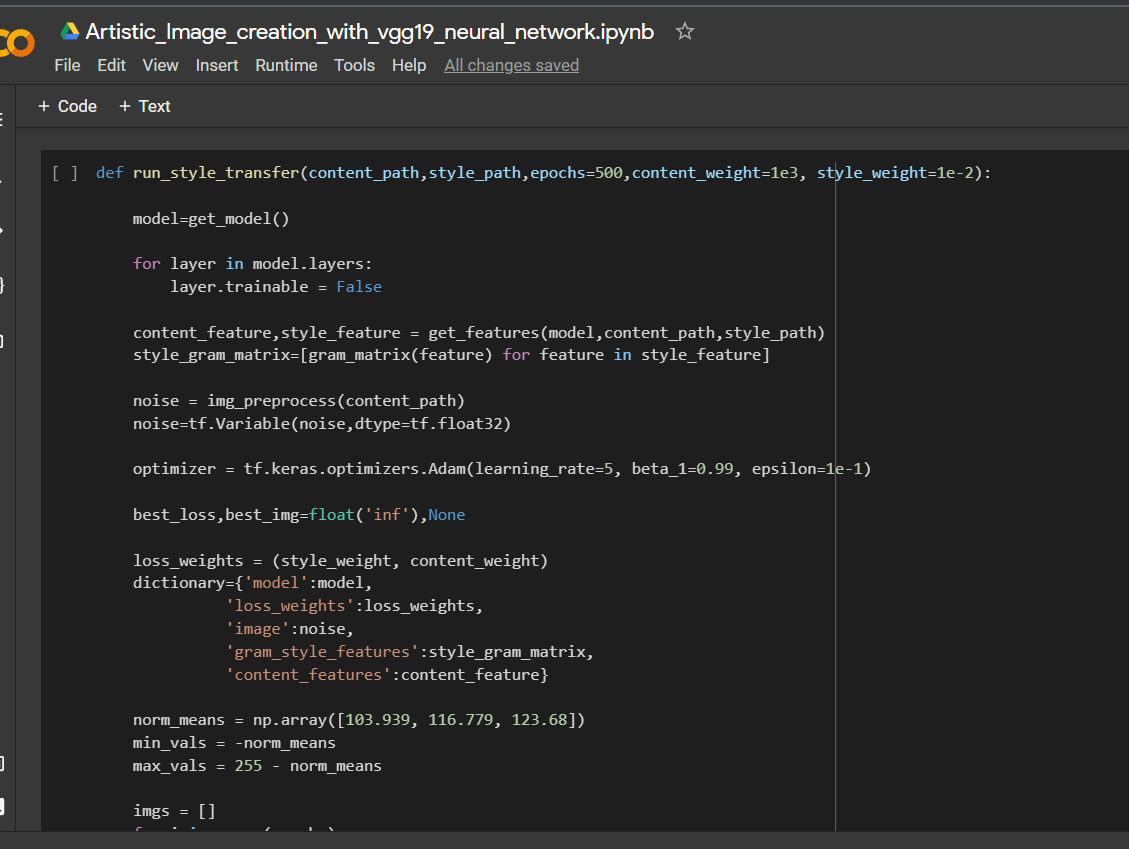
**RESULT**

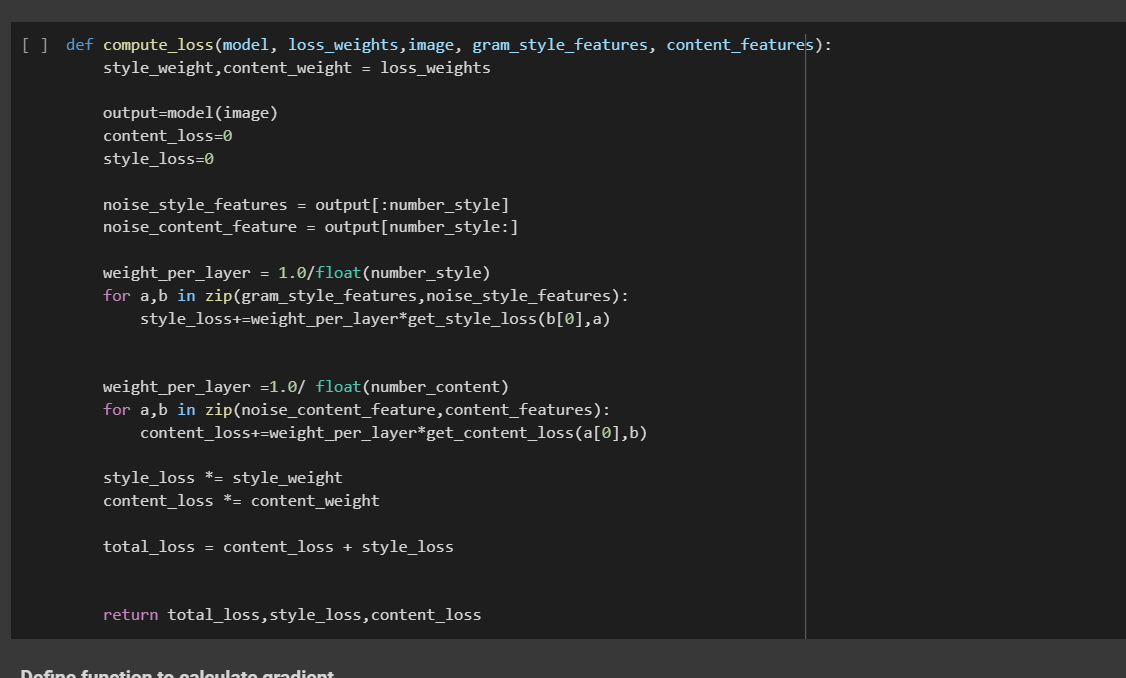






**CODES SNAPSHOT**





**CONCLUSIONS**

This is not a perfect algorithm and code as you see that the image is not completely blended to the other but we can surely say that the result can be improved by running on more epochs.

In order to do this, we loaded in a pretrained model and used its learned feature maps to describe the content and style representation of our images.

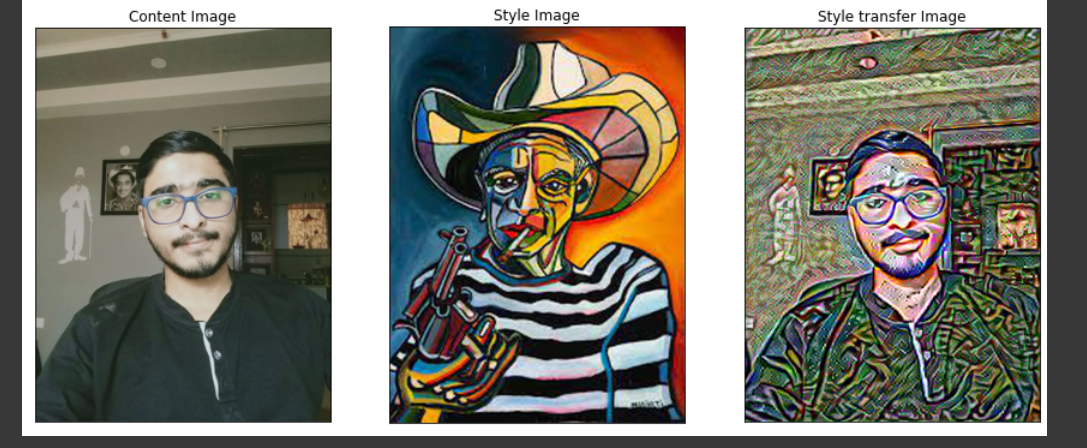


Image is not perfect but will improve in more epocs.

**References**

[https://towardsdatascience.com/neural-style-transfer-tutorial-part-1-f5cd3315fa7f](https://www.tensorflow.org/tutorials/generative/style_transfer)

<https://www.tensorflow.org/tutorials/generative/style_transfer>

<https://stackoverflow.com/questions/61292890/transfer-learning-only-works-with-trainable-set-to-false>