A Comprehensive Report on Reformations Done on the Work of -

A Neural Algorithm of Artistic Style

Leon A. Gatys, Alexander S. Ecker, Matthias Bethge

Submitted in Completion of -

Final Project of Course EE 526

On Fall 2020

Citation - Gatys LA, Ecker AS, Bethge M. **A Neural Algorithm of Artistic Style**. arXiv preprint arXiv:1508.06576. 2015 Aug 26. from link: Go to paper

Submitted By

Md Sakib Ferdous

Chemical and Biological Engineering Department

Iowa State University

Submitted To

Julie Dickerson

Chenyu Xu

Motivation

Generative adversarial networks came up with a lot of potential. In last few classes we had seen how this GAN networks can create fake images or super impose style of one image on another to create a new content image with the style of the prior. This can be used to observe how one image would look like if some famous painters would have drawn. The immense potential of Generative networks in the coming decades made me interested to work about Style GAN in the final project.

Personally apart from being a chemical engineer, I was a painter before starting grad school. I always wanted to start painting again but could not manage the time amidst grad studies. When I came to know about StyleGAN I started wondering if I could use my previous painting style on some new images to see how my paintings might look like. Technology gives us the scope to ponder unexplored life choices, undoubtedly very interesting time to be alive. (in the end I have attached a sample of one of my painting and one GAN generated painting copying my style)

Here I got the chance of **exploring built in image processing module of Keras called VGG19.**The different abstraction level of layered convolution networks offers greater **insight in how neural networks perceive and process information.** I came to know about **various built in loss computation modules in Tensorflow and Keras.** I also attempted different changes to the algorithm, adding or deducting blocks and layers, changing different parameters and see the **effect on corresponding errors and produced images.** Moreover, it was a good scope to know more about **data processing units like Numpy, Pandas, Image, os and matplotlib, processing image data and loading the dataset in google colab.**

With the knowledge and expertise obtained from this project I hope to excel more in my Deep learning research.

Abstract

Paintings by famous artists have two main criteria- one is content of the painting, what is there in the painting objectively. Another is style or how the painting is drawn, what was the combination of color, depth and brush strokes. Humans have cognitive ability to percept an image. At this age of artificial intelligence, neural networks are now able to ‘learn’ to percept abstract features of images with the help of convolutional neural network which works by mimicking the working structure of human brain. Thus it is capable of extracting styles of a certain painting by different layers of abstraction. In the pioneering Paper on this subject by Gatys, Ecker and Bethge titled **‘A Neural Algorithm of Artistic Styles’** they attempted to extract features out of famous paintings to impose it on a photograph to create first A.I. generated artwork. The implementation of the code is obtained as a google colab notebook from online. A pre-trained model called VGG19 is loaded which contained learned parameters to extract feature maps. The project goal is to develop the visual appeal of the result image by altering the parameters, specifically to get the actual content clear enough retaining colors from content but coloring style from style image. In this report result of several modification attempt like- **adding 3 fully connected layer on top, changing weights of the style and content, adding max pooling layer, changing the block features, addition of convolution steps and iteration number** at different levels is summarized. The loss of content and style was also calculated, normalized, plotted and corresponding images were also generated to observe how different modifications affect the changes. From the results it is observed that, **addition of max pooling and three top FC layers do not bring any visual change in the image. But addition of second layer of convolution helps in extracting the features retaining content to great extent and the visually best image is obtained thus the project goal is met. Changing the weights and iteration number causes the final image to be dotted but the overall score improves. Overall it took around 5 minutes for each model to train where time required by each epoch varied from 0.07 second to 0.08 second**. In the end further modification were also proposed which might have interesting results.

Introduction

a) Basics of Computer vision

The CNN network consists of building blocks each working as a neuron or processing unit. They perform the task of recognition of images by developing features at each steps. As the layers go deep they are able to extract more and more abstract features of the image. Thus it is possible to infer actual content of the image by a CNN as the pixel values, their arrangement becomes less important. For example, an image of a car can be taken at day and night causing very different pixel values and arrangement. But higher level features are same for both the car by which CNN recognizes it as a car.

b) Context of work on Paper

As NN are capable of extracting abstract features, they can also be used to catch the texture information. By stacking feature correlation of different layer, it can extract the texture information only excluding the content. In summary the work presents a way to separate content and style of an image[1]. As they are separable, they can be manipulated in different creative ways to superimpose on another image so that its content is retained but it copies the style of the former. Here two different loss functions are used for gradient descent for smooth reconstruction retaining the style and content.

c) Modification of the present work

The implementation of the network is obtained from in form of a google colab notebook. Here modification was done in total of five steps –

1. In the VGG19 architecture parameters – include top, weight, pooling and classes
2. Altering the convolutional blocks
3. Altering convolutional types
4. Adding max pooling
5. Increasing number of iterations

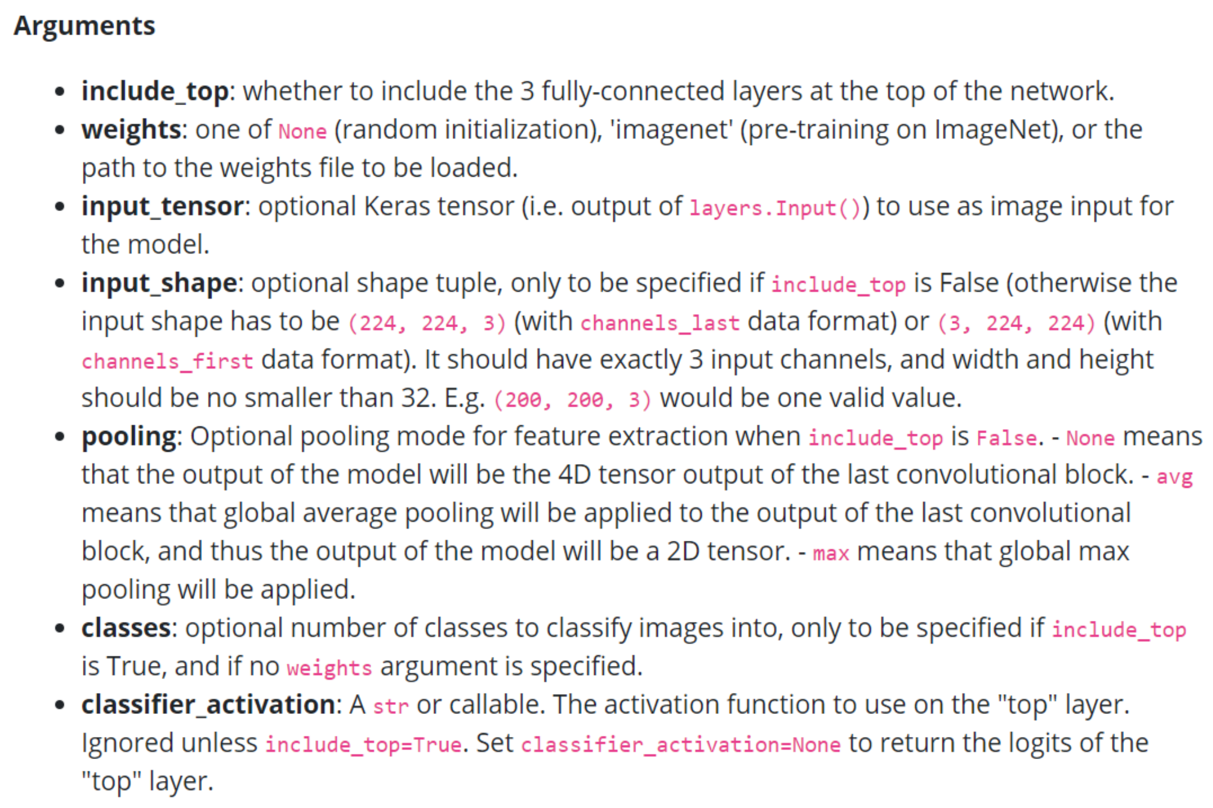
They are summarized in next page.

Change at Level	Changes
1. In the VGG19 architecture parameters	include_top=**True**, pooling= **max**, classes=**1000**)
2. Altering the convolutional blocks in contents layer from 2 to 1	content_layers = [**'block1_conv2'**]
3. Adding second convolution block after each main block	style_layers = [**'block1_conv2'** **'block2_conv2'**, **'block3_conv2',** **'block4_conv2'**]
4. Adding 4 additional Convolutional blocks in Layer 4 and 5	[**'block4_conv1,2,3,4'**, **'block5_conv1,2,3,4**]
5. Changing $\frac{\alpha}{\beta} = \frac{Content\ weight}{Style\ weight}$	Increased **100** times
6. Increasing number of iteration	**10000** (ten times increase)

Methodology

a) The VGG19 image processing framework

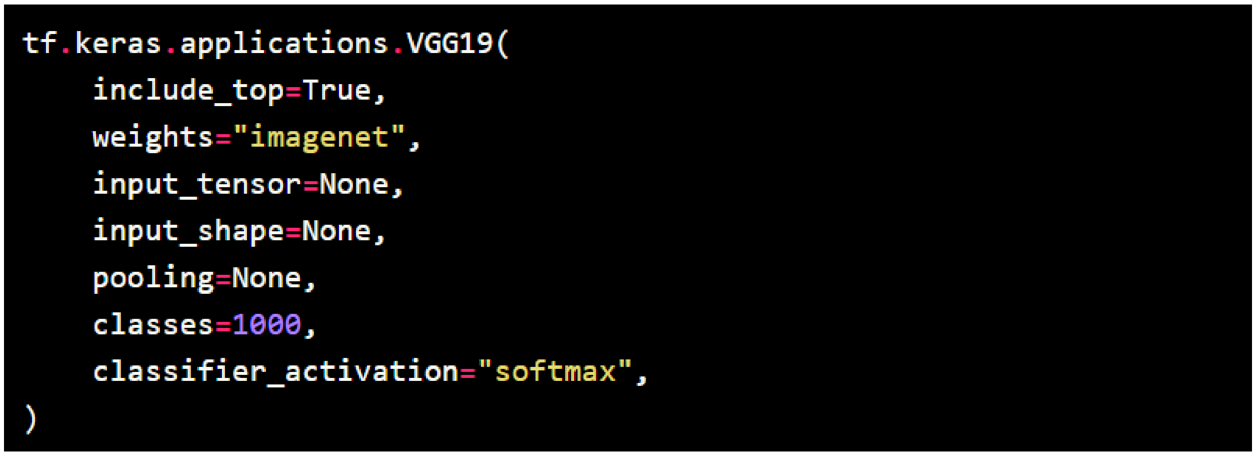
The VGG19 is the successor of AlexNet, which is a Convolution Neural Network used for image classification pre trained on 14 million datasets of ImageNet. [2] The arguments of the input of the network [3] are –



Arguments

- **include_top**: whether to include the 3 fully-connected layers at the top of the network.
- **weights**: one of `None` (random initialization), 'imagenet' (pre-training on ImageNet), or the path to the weights file to be loaded.
- **input_tensor**: optional Keras tensor (i.e. output of `layers.Input()`) to use as image input for the model.
- **input_shape**: optional shape tuple, only to be specified if `include_top` is False (otherwise the input shape has to be `(224, 224, 3)` (with `channels_last` data format) or `(3, 224, 224)` (with `channels_first` data format). It should have exactly 3 input channels, and width and height should be no smaller than 32. E.g. `(200, 200, 3)` would be one valid value.
- **pooling**: Optional pooling mode for feature extraction when `include_top` is `False`. - `None` means that the output of the model will be the 4D tensor output of the last convolutional block. - `avg` means that global average pooling will be applied to the output of the last convolutional block, and thus the output of the model will be a 2D tensor. - `max` means that global max pooling will be applied.
- **classes**: optional number of classes to classify images into, only to be specified if `include_top` is True, and if no `weights` argument is specified.
- **classifier_activation**: A `str` or callable. The activation function to use on the "top" layer. Ignored unless `include_top=True`. Set `classifier_activation=None` to return the logits of the "top" layer.

Figure: Hyperparameters of VGG19



```
tf.keras.applications.VGG19(
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax",
)
```

Figure: VGG19 input arguments

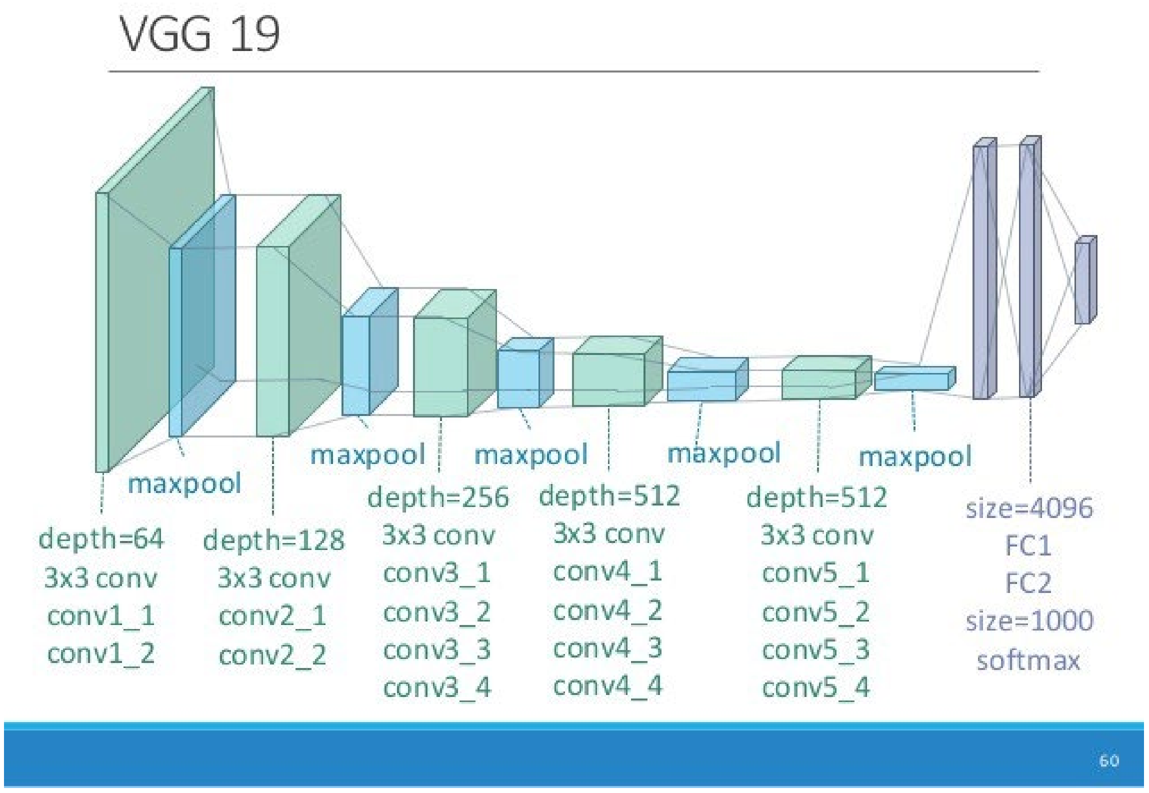


Figure: Structure of the blocks.

Source - https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Ftse1.mm.bing.net%2Fth%3Fid%3DOIP.w4aEwqrfXAl3vB2ZdEaU4gHaFj%26pid%3DApi&f=1

b) Loss function style and content scores

Computing loss is essential to get an image with desired content and style feature by smooth gradient descent. Here two losses are computed and through iteration the algorithm tries to minimize both.

1. **Content loss**

Calculating content loss is straightforward. A base content image and target image is taken as an input and the Euclidean distance between them is computed.

Let, C_{nn} be a pre-trained deep convolutional neural network.

Let, ‘X’ be any image, then $C_{nn}(X)$ is the network fed by X.

Let, $F^l_{ij}(x) \in C_{nn}(x)$ & $P^l_{ij}(p) \in C_{nn}(p)$

Above equation denotes the respective intermediate feature representation of the network with inputs ‘x’ and ‘p’ at layer ‘l’.

Then we describe the content distance (loss) formally as:

$$L^l_{ij}(p,x)=\sum (F^l_{ij}(x)-P^l_{ij}(p))^2$$

2. **Style loss**

Let, style loss of the base input image is x, the style image is ‘a’ as the distance between the style representation (the gram matrices) of these images.

$$E_l= \frac{1}{4N^2M^2} l\sum i,j(G^l_{ij} - A^l_{ij})2$$

where G^l_{ij} and A^l_{ij} are the respective style representation in layer l of x and a. N describes the number of feature maps, each of size M = height x width. Thus, the total style loss across each layer is -

$$L_{style}(a,x)=\sum_{l\in L} w_l E_l$$

c) Implementation of the Code

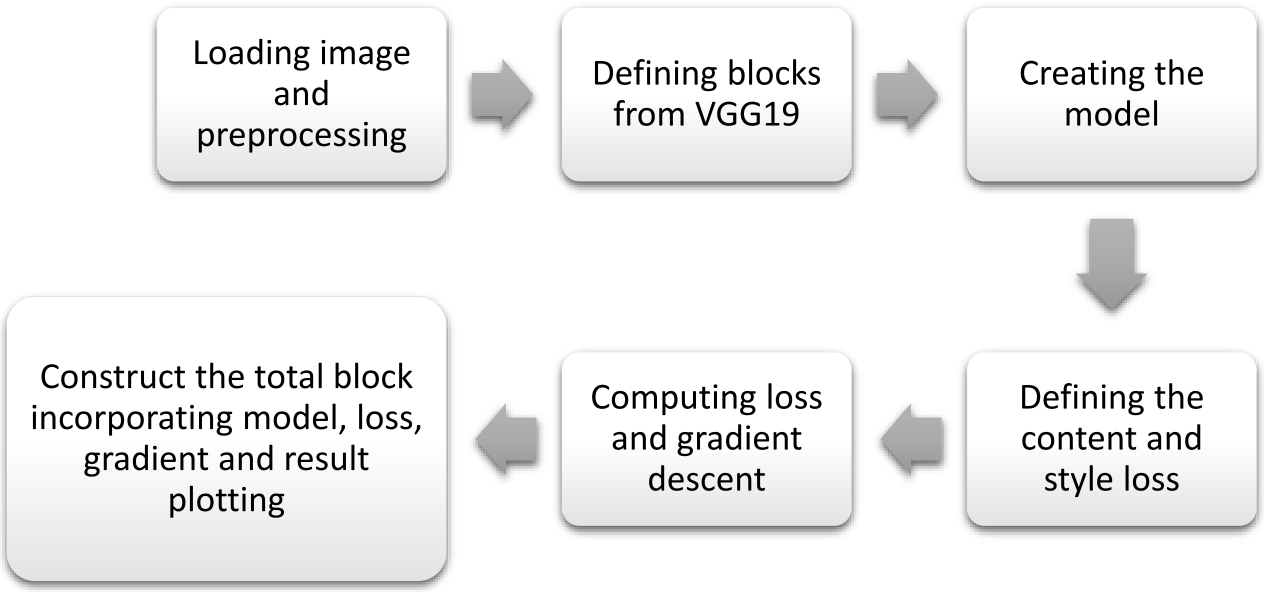
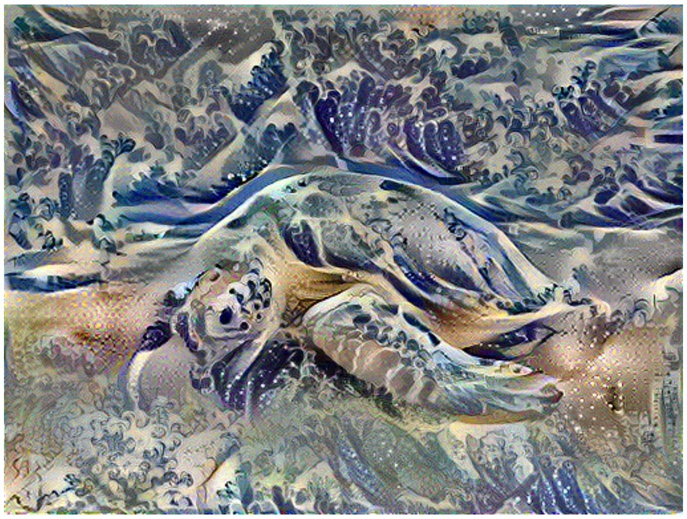

Loading image and preprocessing
Defining blocks from VGG19
Creating the model
Defining the content and style loss
Computing loss and gradient descent
Construct the total block incorporating model, loss, gradient and result plotting

Chart: The structure of Algorithm for Neural Style Transfer Learning

Simulation Results

Original (Trial 1)



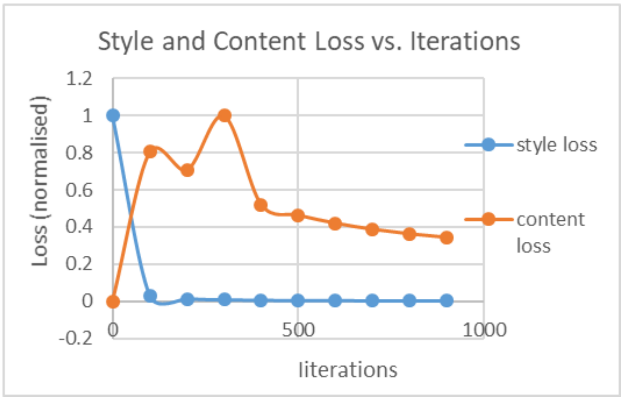
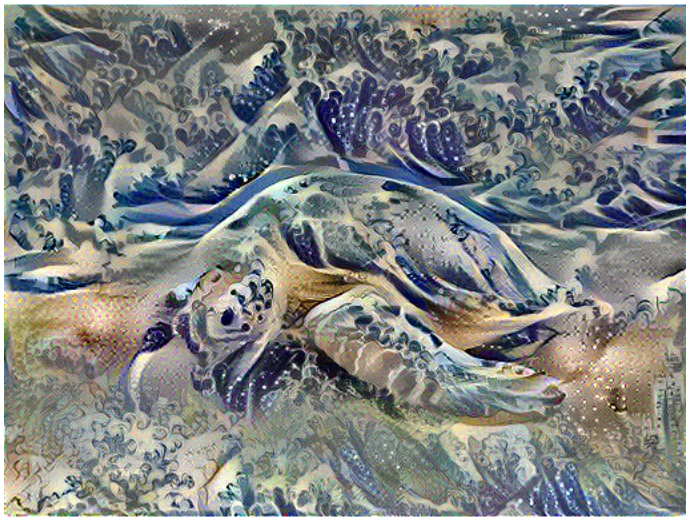


Figure: As a result of superimpose almost all of the color features of the content image is replaced by the style image.

Trial 2



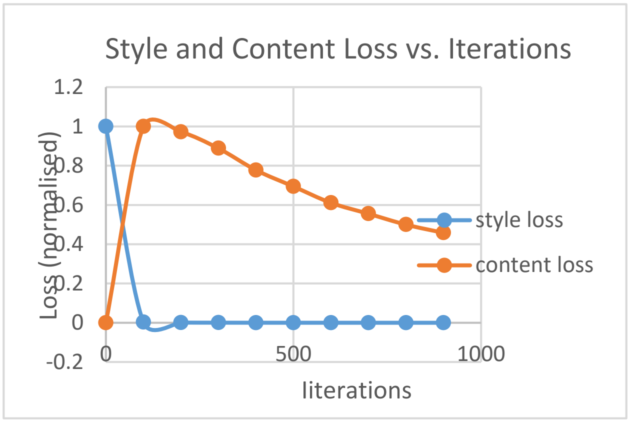
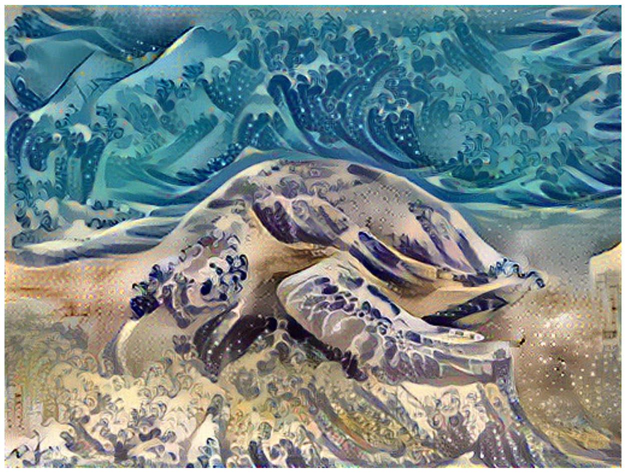


Figure: It is observed that not much change is done on the main work on addition of top fully connected layer, max pooling and adding class. At some part the color feature is retained.

Trial 3



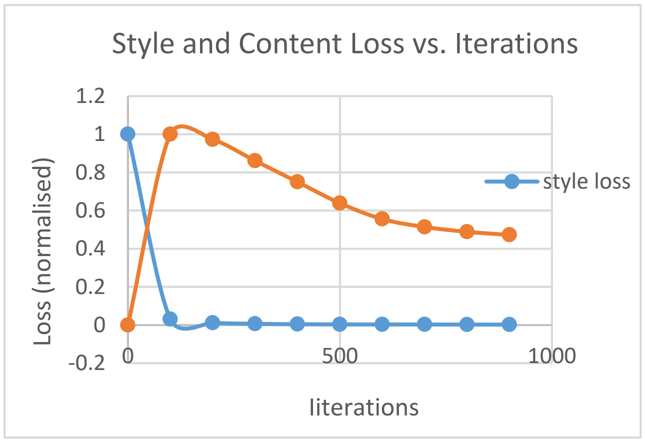
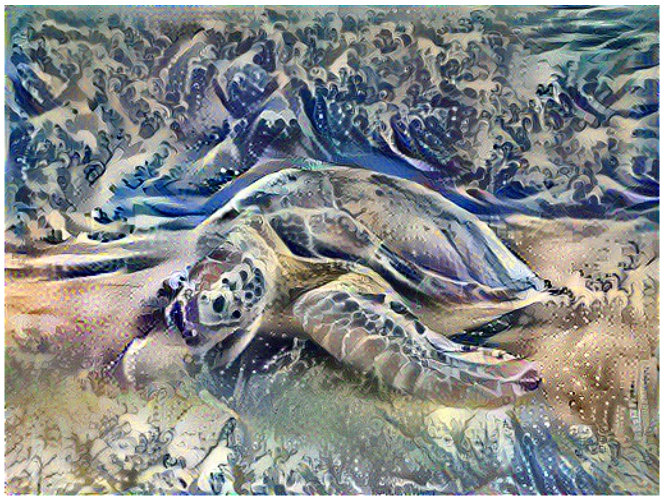


Figure: By far the most visually appealing trial. The content layer block was changed to first block instead of fifth. So Macro features were retained, giving the image the cartooning style of painting avoiding too much detail.

Trial 4



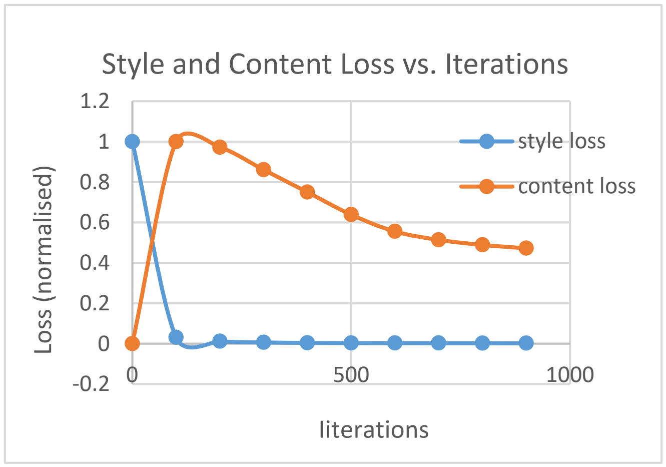
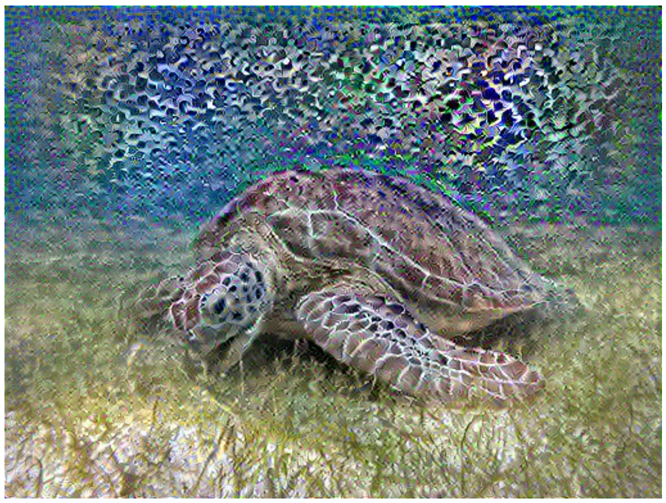


Figure: All the convolutional blocks of layer 4 and 5 used. As a result, the error only increased. But the grass beneath the turtle is much more prominent and not looking like waves. It can be inferred that adding more layers might cause in deviating from actual features.

Trial 5



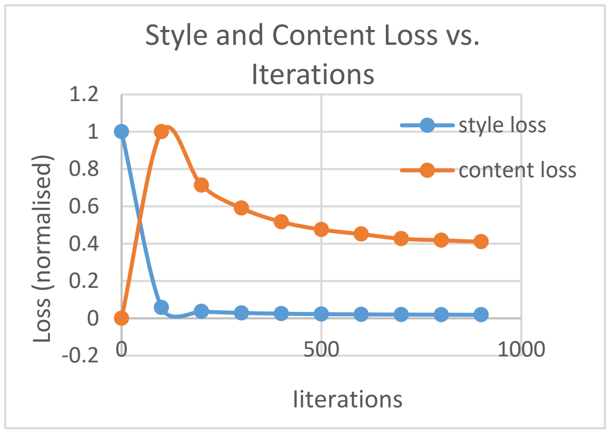
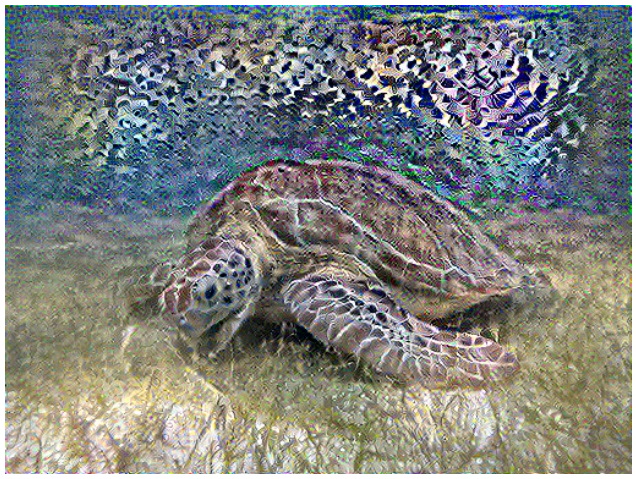


Figure: Changed the feature weights $\frac{\alpha}{\beta}$ by 100 times, resulted in less error. Increasing content weight resulted in retaining most of the content features. The last blocks extract smaller features, cause their kernel size is smaller effect of which seems prominent. Although the error reduced but the image doesn't look better.

Trial 6



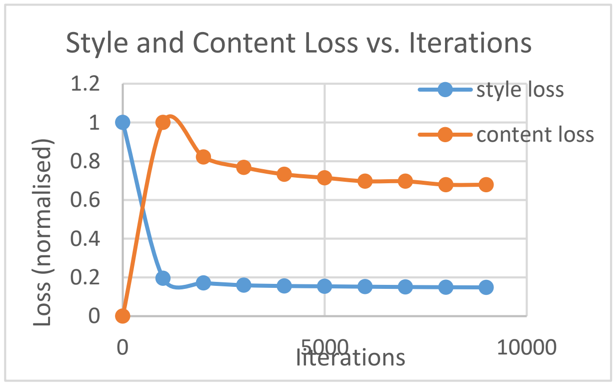
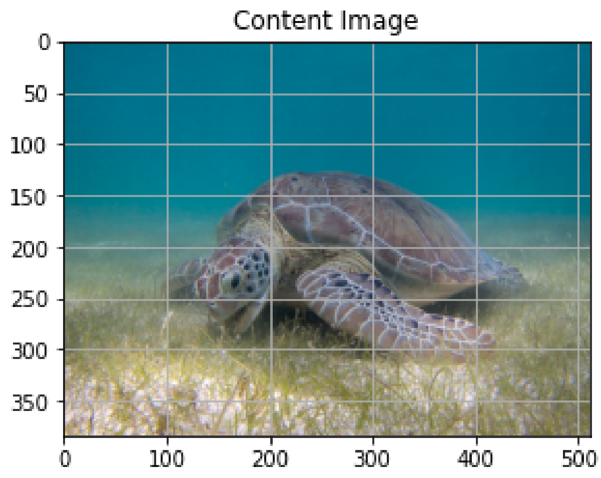
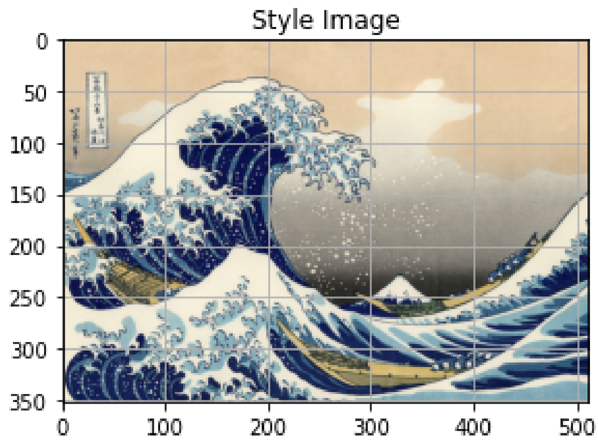


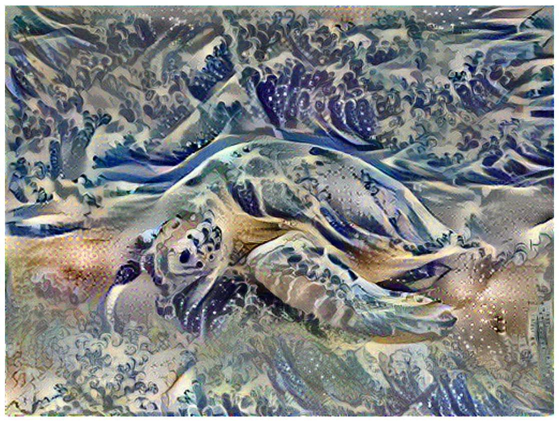
Figure: The iteration number increase extracted minute features of the style, the error increased a little but the image does not appear better.

Summary of Result





Best by the paper



Best by this project

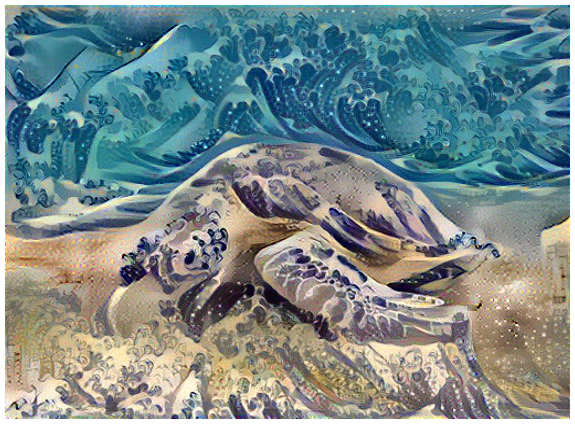


Figure: The content image – “Green Sea Turtle by By P.Lindgren and style image Great wave waves of Kannagawa by Katsushika Hokusai. In the work done in paper the style transfer was little bit hedgy with lots of dots and pixels. Different parameters were changed to obtain the style without interfering the content.

Error Analysis

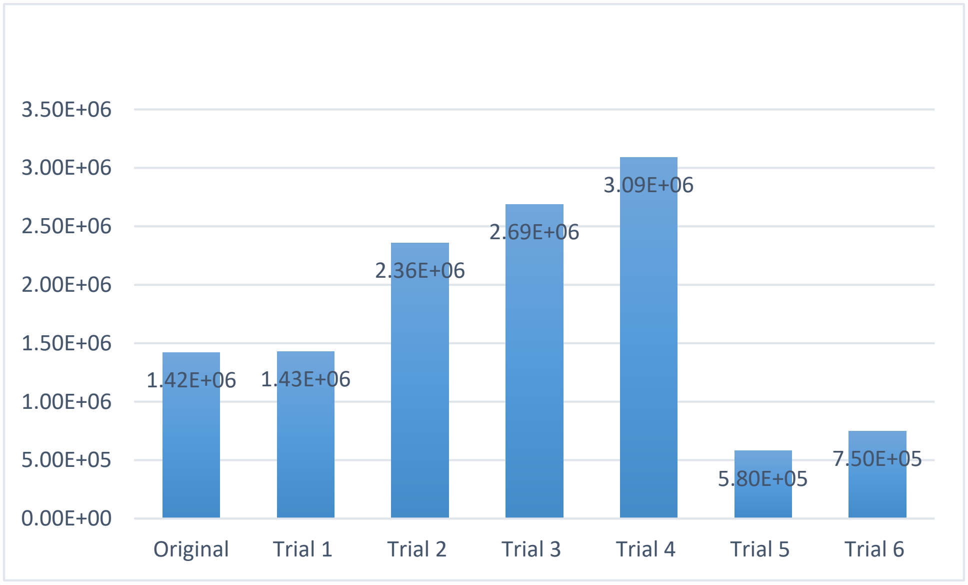


Figure: Error comparison of all the trials and original work.

It is observed that addition of max pooling, and fully connected layer does not contribute in increase of the result. But changing the convolutional layers increased error but the image appeared better. The error function is not the right representative of visual appeal. Increasing the content weight decreased error as expected. But after a certain iteration number the score becomes even worse.

Scope for Future Work

There are lots of other motivations possible to bring in the original work by Gatys. A very detailed review is obtained from Roman Novak titled “Improving the Neural Algorithm for Artistic Style”. [4] They are –

1. Layer weight adjustment: Using different weights at different layers
2. Using more Layers: This is attempted in this paper and worked really fine
3. Activation Shift: This will allow to help with Trial 5 and 6 where sparsity ruined the image.
4. Correlation chain
5. Blurred Correlations
6. Gradient masking
7. Amplifying activations
8. Adjacent activation correlation
9. Content aware gram matrices
10. Gram cubes

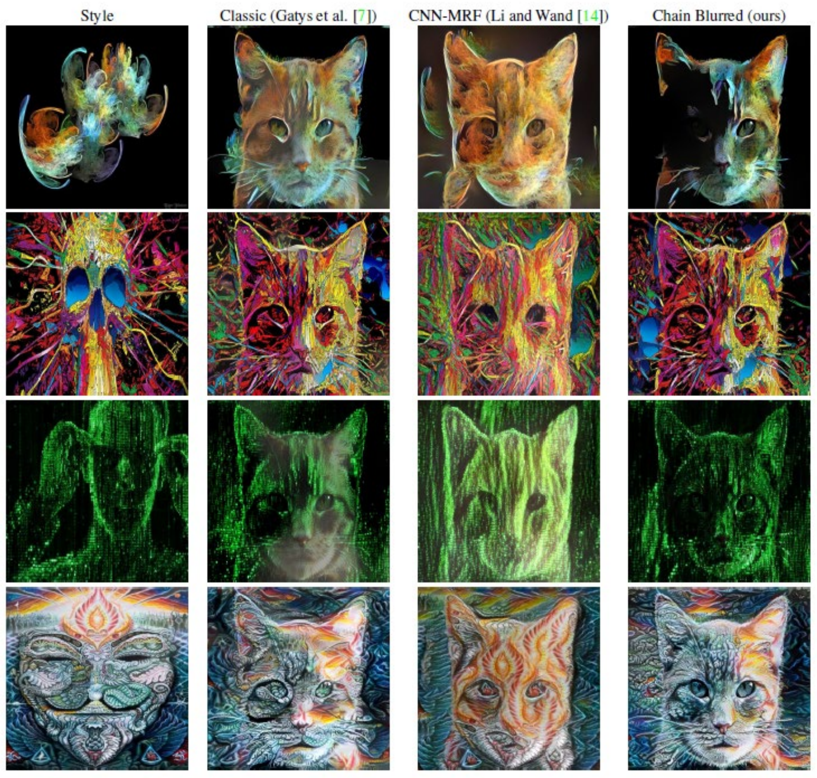


Table 3: Comparing our results to existing style transfer solutions (part 1). Style images (top to bottom): [16, 2, 15, 20].

Figure: Comparison of the work of Gatys with other developed procedure.

Apart from this changes certain image manipulation can be done prior to extracting features to separate content from background, dividing the image in half or centralizing and treat each part differently.

Conclusion

In this project different parameters were adjusted to produce more visually appealing image extending the pioneering work on Style transfer learning. Paintings are abstract art pieces having aesthetic values perceptible by humans. This work was a great scope to observe how neural networks and associated block treats image. One of the important learning outcome was to know that the score or loss function is not necessarily best representative of aesthetics of the image. It was also observed that the additional convolutional blocks take the pattern into account other than retaining colors. The improved knowledge of feature extraction can be used in various other machine learning fields.

Reference

a) Papers

[1] L. Gatys, A. Ecker, and M. Bethge, “A Neural Algorithm of Artistic Style,” *J. Vis.*, vol. 16, no. 12, p. 326, 2016, doi: 10.1167/16.12.326.

[2] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–14, 2015.

[3] R. Novak and Y. Nikulin, “Improving the Neural Algorithm of Artistic Style,” pp. 1–15, 2016, [Online]. Available: http://arxiv.org/abs/1605.04603.

b) Links –

[4] VGG19 details - https://keras.io/api/applications/vgg/

c) Codes -

“Neural Style Transfer with Eager execution”

GitHub Source -

https://github.com/tensorflow/models/blob/master/research/nst_blogpost/4_Neural_Style_Transfer_with_Eager_Execution.ipynb

Google colab link –

https://colab.research.google.com/github/tensorflow/models/blob/master/research/nst_blogpost/4_Neural_Style_Transfer_with_Eager_Execution.ipynb

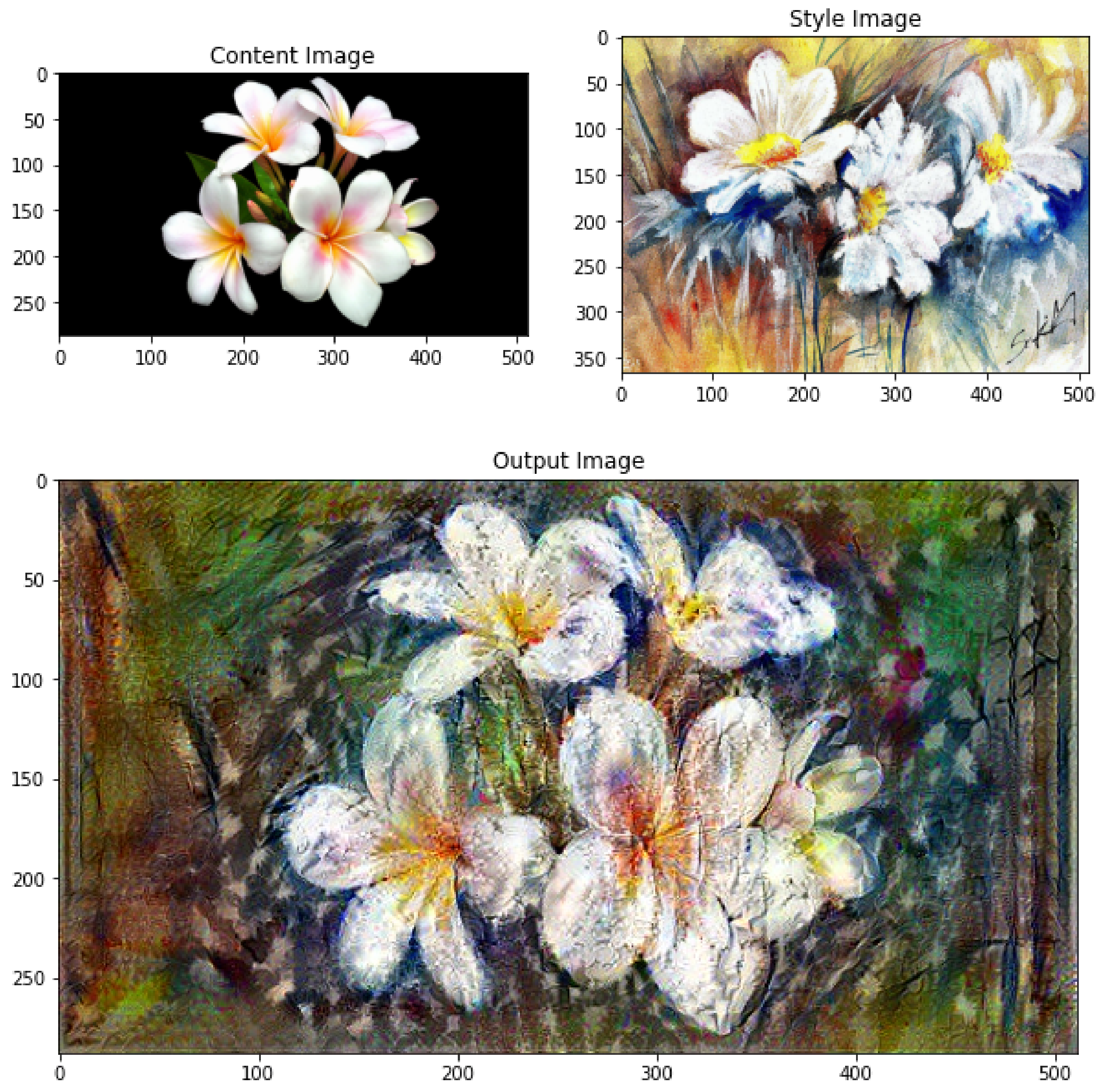


Figure: On top left is an image obtained from online. On right one of my paintings I painted in 2018. Bottom image is the style transfer image.