

Model Evaluation for Neural Style Transfer

Project Title : **Neural Style Transfer**

Course Name : **Capstone II**

Course Facilitator : **Marcos Bittencourt**

Group Number : **Group - 9**

Student Names and IDs :
1) Dilsher Singh – 100768315
2) Rajan Walia – 100727112
3) Fulya Caliskan– 100766627
4) Peter Kusiak – 100399575

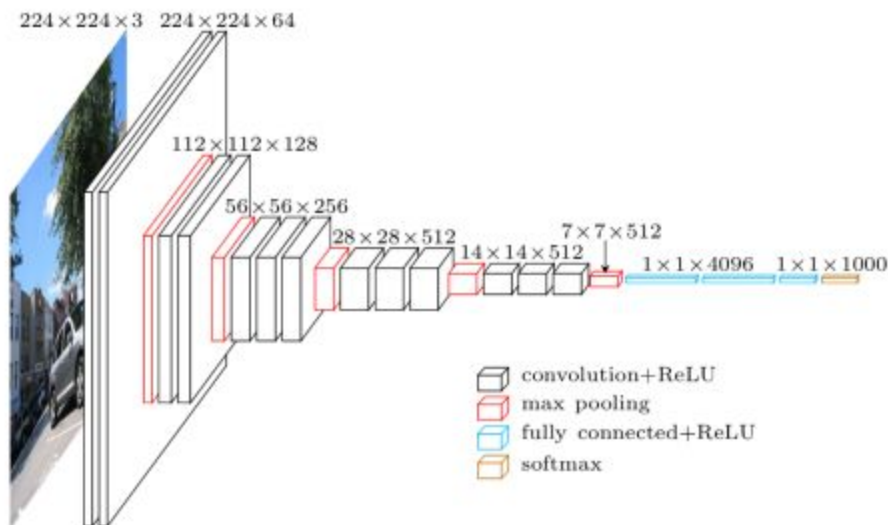
Date of Submission : **March 6, 2020.**

Quality Assurance

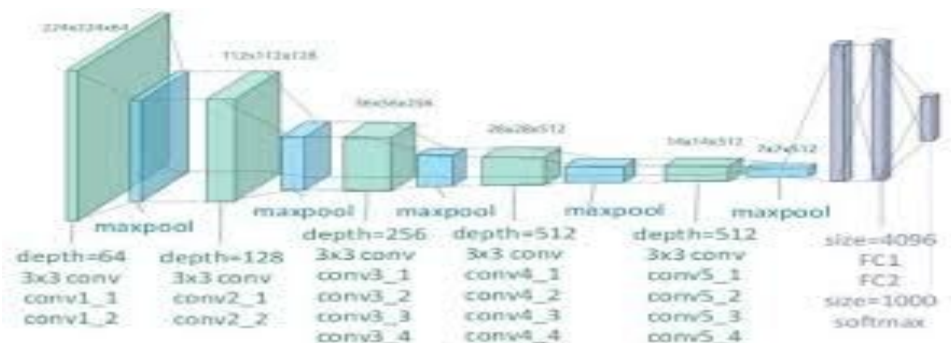
Models used

Using pre-trained models is one of the prerequisites for the project. Below are pre-trained models and examples of their architecture that may be used as a part of this exercise:

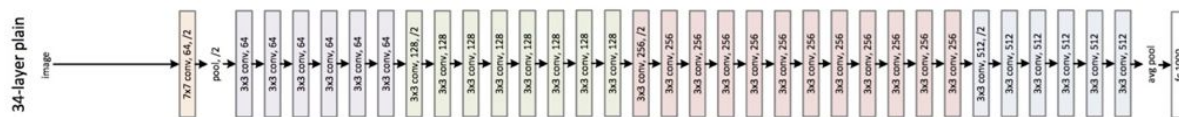
- VGG - 16



- VGG- 19



- ResNet



Images Used



Wave picture - Style image



Sea Turtle - Content image

Model Evaluation Metrics

The evaluation is made on the basis of below key metrics -

Content Loss

The content loss is a function that describes the distance of content from our input image x and our content image:

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

x represents any image. $F_{ij}^l(x) \in C_{ij}^l(x)$ and $P_{ij}^l(p) \in C_{ij}^l(p)$ describe the respective intermediate feature representation of the network with inputs x and p at layer l .

Style Loss

The style loss is the mean squared distance between the feature correlation map of the style image and the input image:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2$$

G_{ij}^l and A_{ij}^l are the respective style representation in layer l of input image x and style image a . N_l describes the number of feature maps, each of size $M_l = \text{height} \times \text{width}$.

The goal is to minimize the sum of Style Loss and Content Loss. The Gradient Descent algorithm as an optimization technique will be used to minimize the loss.

Loss Curves

Deep Learning is used only in the second part of the project (ie to style the image), hence the below part is only related to the second part. The two sample images used are the “*Green Sea Turtle grazing seagrass*” and “*The Great Wave off Kanagawa*”

The loss is minimized till 1000 iterations. For better visualisations, the learning curves are plotted only till 200 iterations. Below are the loss curves and the images generated for each of the pre-trained models used -

VGG - 19

Loss Curves-

```
Text(0, 0.5, 'Style scores')
<Figure size 2160x720 with 0 Axes>
```

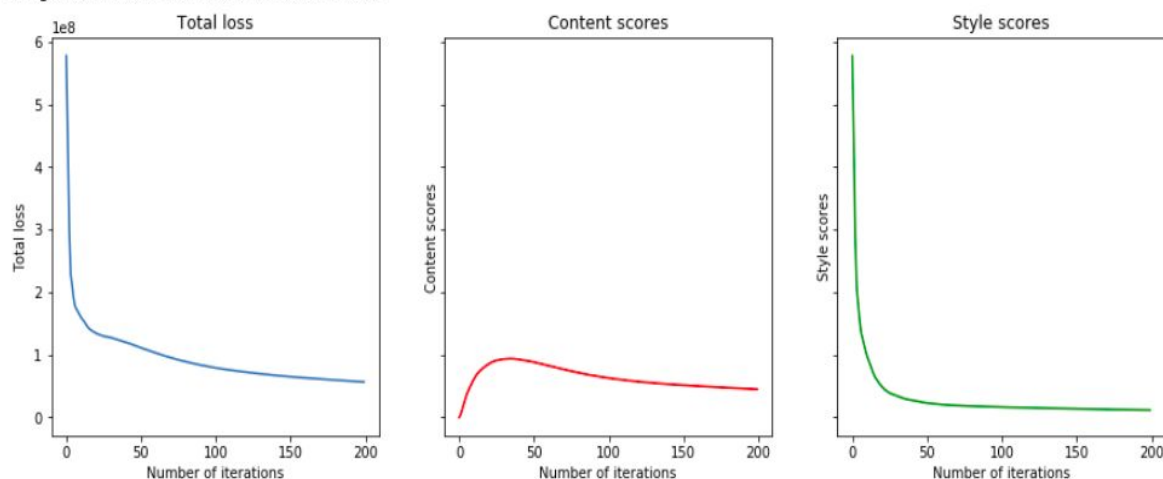


Image generated -



VGG - 16

Loss Curves-

```
Text(0, 0.5, 'Style scores')  
<Figure size 2160x720 with 0 Axes>
```

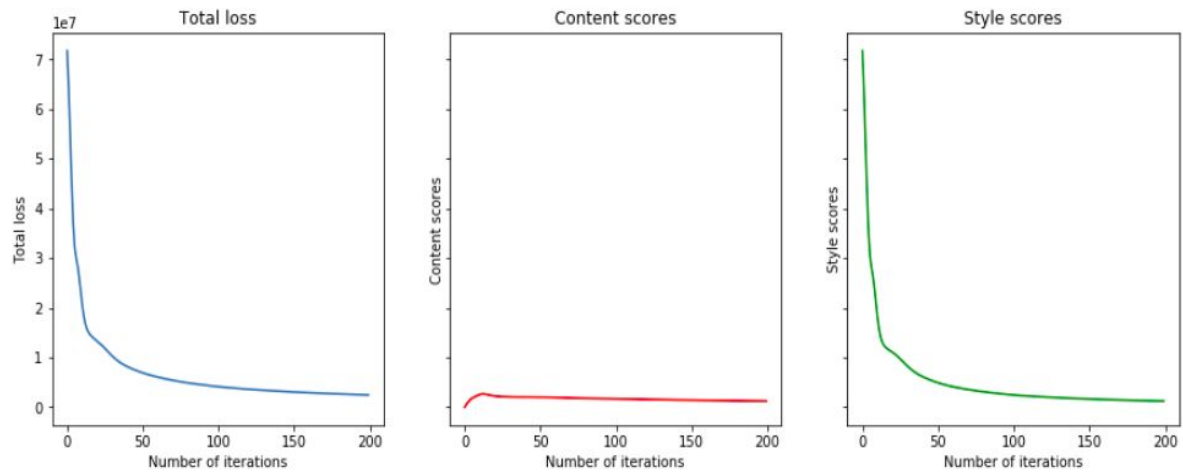


Image generated -



Resnets

Loss curves -

```
Text(0, 0.5, 'Style scores')  
<Figure size 2160x720 with 0 Axes>
```

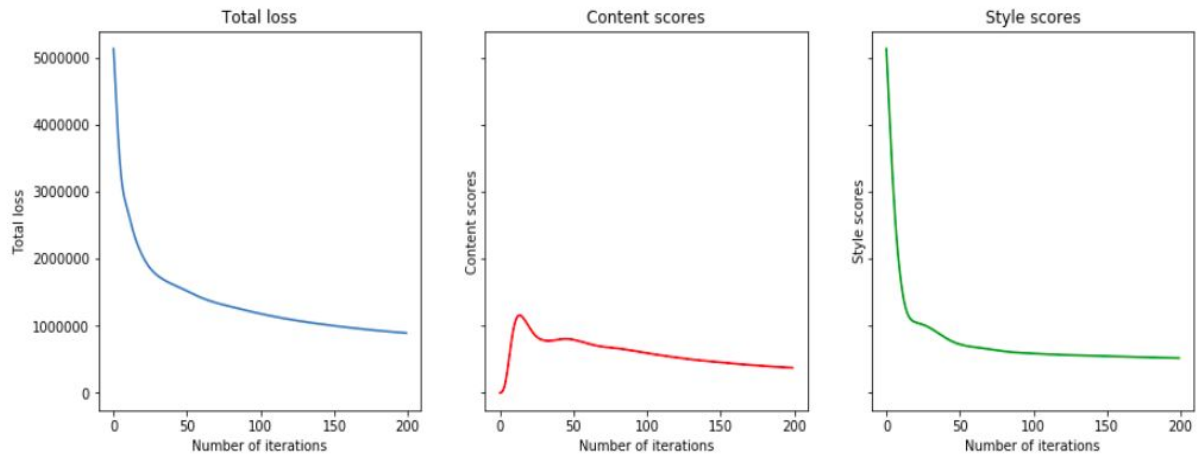


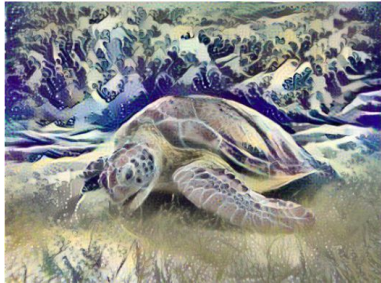

Image generated -



Visualizing outputs for different layers

In CNN architectures, different layers have different layer outputs. Therefore, the layer which we select will have a profound effect on the image generated. Below are the images generated for each layer and the learning curves for each of the following. The outputs are gathered from the VGG 16 model.

Pros and Cons of VGG and Inception and Key findings

Model	Image Produced	Pros & Cons
VGG 16 & VGG 19	 <p>Successful but still the details of the input image are noticeable. Style effect is lower.</p>	<p>In 2014, 16 and 19 layer networks were considered very deep.</p> <p>VGG is deeper and has more parameters, as well as using only 3x3 filters.</p> <p>It is painfully slow to train.</p> <p>The network architecture weights themselves are quite large (in terms of disk/bandwidth).</p> <p>Due to its depth and number of fully-connected nodes, VGG is over 533MB for VGG16 and 574MB for VGG19. This makes deploying VGG a tiresome task.</p> <p>We still use VGG in many deep learning image classification problems; however, smaller network architectures are often more desirable.</p>
Resnet	 <p>more successful in style transfer.</p>	<p>ResNets were learned with network depth of as large as 152. It achieves better accuracy and is more efficient than VGGNet.</p> <p>The architecture is similar to the VGGNet consisting mostly of 3X3 filters.</p> <p>Even though ResNet is much deeper than VGG16 and VGG19, the model size is actually substantially smaller due to the usage of global average pooling rather than fully-connected layers — this reduces the model size down to 102MB for ResNet50</p>

Final Model

As per the findings, we will be using VGG 16 model as it is sufficiently deep and is able to preserve a lot of content as well as the style (which includes textures and colors). In addition to that, it is lighter than that of Inception network and ResNets which means that it will be able to make the computations faster than other models.

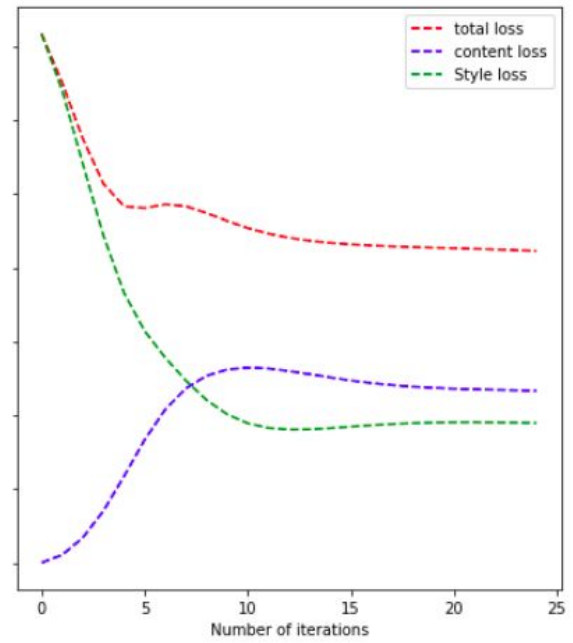
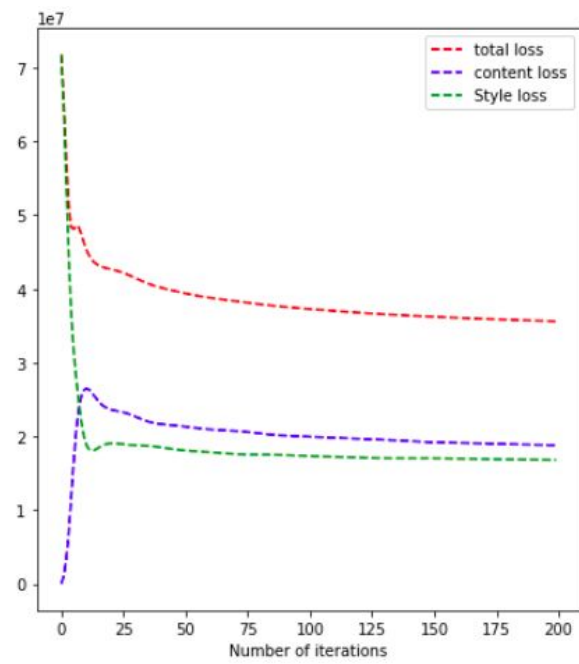
Tweaking the model

VGG 16 has a 16 layer architecture, therefore each layer will output a different image. The initial layers will preserve a high level of content and the minimum amount of style. Below are the different image outputs for different layer as an input -

Layer 1



Image from layer 1 output. As we can see that the content is preserved more than the style because initial layer will output very sharp images, and there is no enough space to fill the gap for the style



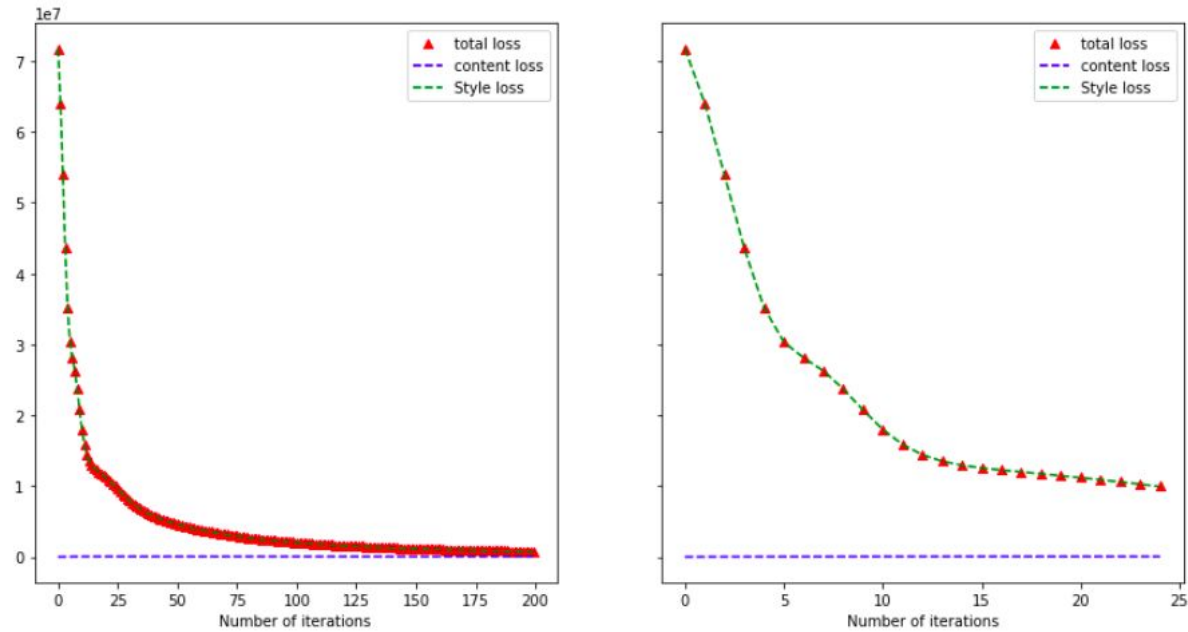
Loss curves for the first 200 iterations and first 25 iterations respectively.

Layer 5



Image from layer 5. There are more elements of style than that of the content. This is because, the deeper layer produces more blurred output as compared to the initial layers, hence the leftover space is occupied by the style

<matplotlib.legend.Legend at 0x7f30359ede48>



Loss curves for the first 200 iterations and first 25 iterations respectively.

Based on the above image represented, we will be using the output of layer 5 of the VGG 16 network

Code

Below is the github link for the code -

https://github.com/RajanAhluwalia/capstone2-Neural-Style-Transfer/blob/master/Neural_Style_Transfer_with_Eager_Execution.ipynb