NEURAL STYLE TRANSFER

INTRODUCTION

The project focuses on the transfer or adapt the style of a image to another using pretrained vgg19 model.

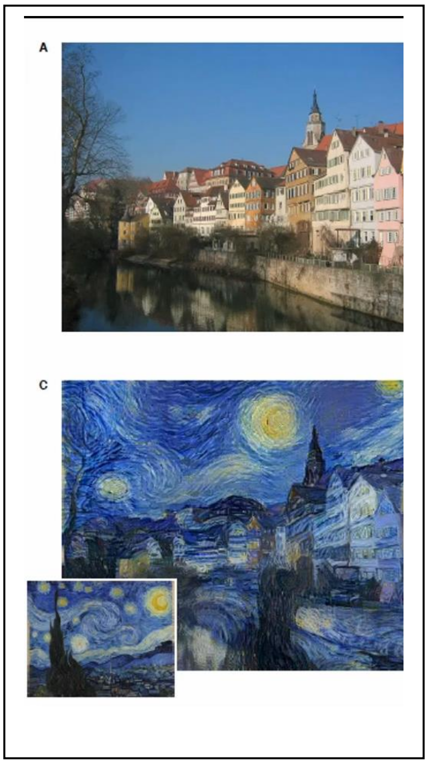
The objective of this project is to understand the logic and concepts so that it can be practically applied by using machine learning and python along with its libraries.

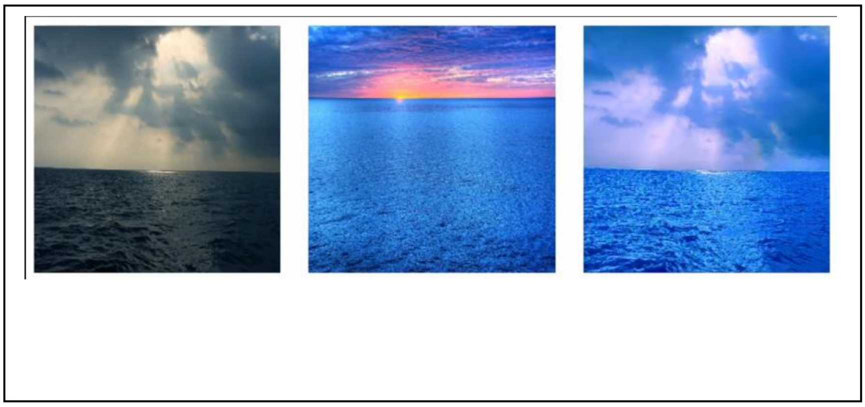
The Neural Style Transfer is one of topics that is famous among ML enthusiasts. Many have tried to approach the problem and many models have came up with different approaches. Many competitions where held in past to address this problem.

There are mainly two types of style transfer

1)Artistic (picture shown sideways)

2)Photo-Realistic(picture shown below)





FAILED APPROCH

According to the theory paper and resources used in my study we have extracted the layers from the pretrained models. I have used vgg19 model .

Content image is defined as the image on which style has to be applied. For extraction of the features for this image is ‘conv_4’ layer in the vgg19 model

For the style image (from which style image is to be applied) features are extracted from the layers ‘conv_1’, ‘conv_2’, ‘conv_3’, ’conv_4’, ‘conv_5’.

Now a Gram Matrix is defined. This matrix is the feature matrix of the style image. It is independent of the orientation of the image (i.e. it will be same for the mirrored images). Gram-matrix(a matrix comprising of correlated features) for the tensors output by the style-layers. The Gram-matrix is essentially just a matrix of dot-products for the vectors of the feature activations of a style-layer. If an entry in the Gram-matrix has a value close to zero then it means the two features in the given layer do not activate simultaneously for the given style-image. And vice versa, if an entry in the Gram-matrix has a large value, then it means the two features do activate simultaneously for the given style-image. We will then try and create a mixed-image that replicates this activation pattern of the style-image.

If the feature map is a matrix F, then each entry in the Gram matrix G can be given by:

Mathematically Gram Matrix is defined as

$$G_{ij} = \sum_{k} F_{ik} F_{jk}$$

Then the paper defined 2 losses →

1)The Content Loss→ given a chosen content layer l, the content loss is defined as the Mean Squared Error between the feature map F of our content image C and the feature map P of our generated image Y.

Mathematically it is defined as

$$\mathcal{L}_{content} = \frac{1}{2}\sum_{i,j}(F_{ij}^{l} - P_{ij}^{l})^2$$

2) The Style Loss →The loss function for style is quite similar to out content loss, except that we calculate the Mean Squared Error for the Gram-matrices instead of the raw tensor-outputs from the layers.

Mathematically it is defined as

$$\mathcal{L}_{style} = \frac{1}{2}\sum_{l=0}^{L}(G_{ij}^{l} - A_{ij}^{l})^2$$

Then the total cost of the step was defined as the liner combination of the both the losses (alpha*L_1 + beta*L_2 {similar}).

The main concept of this model was that the content and style matrix where initiated as the noise and the error between both is minimized. To preserve the originality of the content image it was preserved int the dummy variable. A new noise and feature tensors where made from minimizing the loss and implemented over the dummy variable made.

The optimizer used is Adam

This gave us image with the style.

However this was not a good approach and model failed because of the following reasons →

1)Random Patterns: Noise lacks any meaningful structure or content. The lack of any recognizable features makes it harder for the optimization algorithm to find a good balance between preserving content and applying style.

2) Gradient Vanishing or Exploding: Neural style transfer typically involves optimizing the generated image to minimize the difference in both content and style representations between the generated image and the content/style images. However, starting with noise may lead to gradients becoming either

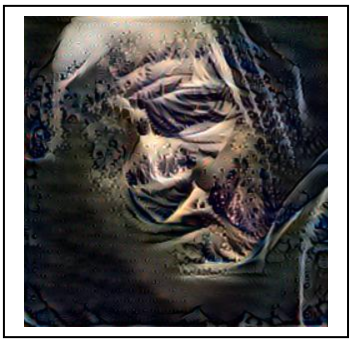
too small (vanishing gradients) or too large (exploding gradients) during optimization.

3) Longer Convergence Time: Due to the lack of meaningful content in the initial noise image, the optimization process may take longer to converge to a satisfactory result.

The failed model is also attached in the GIT-HUB repo

Lets see the results obtained by this method

As you can see below the final image is just a mess and took over 50 minutes to produce with 4000 steps used.



THE APPROCH THAT I FINALLY USED

Now learning from the previous model used I have slightly different approaches.

The major changes I made in this are →

1)Changing optimizers →LBFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno.) is used instead of Adam optimizer . It is more suitable for this code and gives faster convergence than Adam

2)The loss is normalised and linear combination is not used instead the addition of the both the losses defined in previous section is directly added and tried to be minimized

3) The Image is not initiated as noise tensor instead as feature tensor. This is the biggest change made as the major loss of the content in the previous method was due to the fact that the image was initiated as noise tensor.

The previous code was updated and functions where redefined using pytorch(in last model tensorflow was used).

RESULTS AND SIGNIFICANCE

Here is the loss obtained

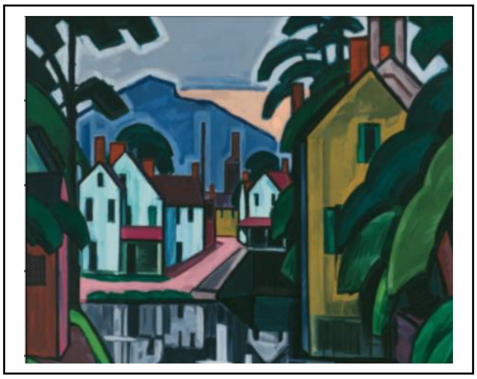
After 50 steps loss is → Style Loss : 23.104406 Content Loss: 9.193089

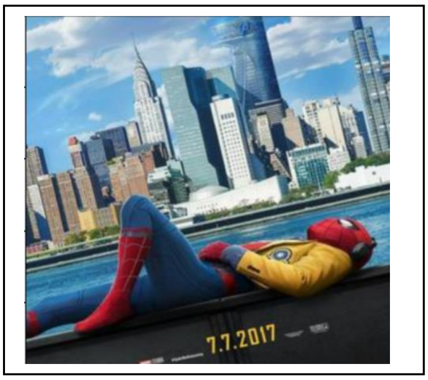
After 500 steps loss is→ Style Loss : 0.603409 Content Loss: 6.164933

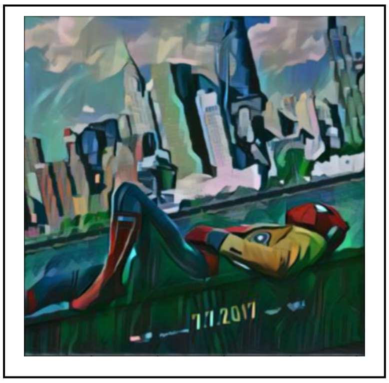
After 450 steps loss is reduced around 97%

Content Loss is not altered significantly due to the fact that originality of the image should be preserved.

Lets see the results obtained by this method







As you can see results are
quite satisfactory despite the run time was around 40 seconds with only 500 steps.

The results can be made more accurate by running more steps

For reference in first approach we have to use image in input size of 256,256 to get output while in second we have used 512,512.

More accurate images can be obtained by number of steps and weights of style.

Second way was more efficient in terms of computation also with greater accuracy.

Second way was more effective both in terms of accuracy and computational power.

The most important insight I have gained is that altering and making calculations a bit easier than before can give more good results. Complex model is not always the best solution.

CONCLUSION

Some improvements that can be made in this field are→

1)Improving algorithm efficiency for real-time and high-resolution transfer is a critical problem.
High-Resolution Outputs: Creating models that can process and generate high-resolution pictures while preserving a considerable amount of content fidelity and style features

2)The Quality and Diversity of Style Transfer
Better Style Illustration: Complex styles can be better captured and represented, particularly in elaborate and abstract art forms.

3)Multiple Style Blending→
Developing methods for smoothly combining different styles to provide more imaginative and varied results.

4)Content-Aware Style Transfer→Improving the capacity to apply styles in a way that takes into account the content, making sure that significant elements are retained and suitably emphasized

Initializing pictures as noise tensor is not much effective idea when it comes to artistic style transfer but it may be used in Photo-Realistic style transfer as it requires to blend two images and this logic might work there.

REFERENCES

https://arxiv.org/pdf/1508.06576.pdf

https://arxiv.org/pdf/1603.08155.pdf

https://pytorch.org/docs/stable/index.html

https://keras.io/api/applications/vgg/

https://www.tensorflow.org/api_docs

Report by → Prahas Tambe

22117104 , Mechanical Engineering