Style Transfer Comparison Using Different Optimization Methods

Nilesh Pandey

April 26, 2018

Abstract

Android applications like prisma made huge buzz when it was launched back in end of 2015. Every other person was interested to have their pictures as if it was painted by famous artists like picasso. Convolutional Neural Network is tool of our choice and state of art. When we pass an image through Convolutional Neural Network, the image is transformed through series of transformation. The Style transfer aims to extract features from the layers of our choice, and this feature maps are used to compare with feature of the other two images which are content and style image. Style transfer aims to decrease the loss between them till the target image has resemblance to both content and style image. In this project we try to explore more techniques and perform extensive experiments on the models and hyper-parameters . We then try to give visual presentation to our result.

1 Introduction

[1]"A Neural Algorithm of Artistic Style" by L. Gatys, A. Ecker, and M. Bethge. This awesome people where able to extract the activation feature map from layers of Convolutional Neural Network (CNN), which were used to compare with feature map of other image. There main aim was not just comparison but gradually decrease the loss between this extracted feature map. For comparison and gradually decreasing the loss between the extracted feature map, they used gram matrix. It is complicated loss function compared to more commonly used loss function that we prefer when comparing with ground truth. The best example of their work in real time is snapchat filters, and very popular software application prisma. In this project we aim to hack the neural style transfer, and examine how we could improve it, and what result we will observe by hacking into this project.

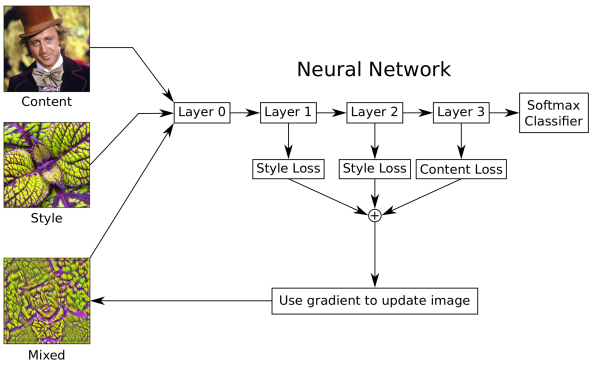

Figure 1: Flow chart for Style Transfer [5]

2 Related Work

Closely Related work "Deep Dream" [14] is done by google engineer "Alexander Mordvintsev". Deep dream also takes the feature maps, but here it tries to enhance the underlying patterns in the image. The Neural Network (NN) used hers it is Inception model. People have achieved great result using Inception model.

Another notable project inspired from style transfer is fine grain visual recognition. In fine grain grain visual recognition, engineers are trying to use the underlying method to differentiate between very similar images.

3 Models

These are some of the models we will be using, and may be we could use some more for better comparison.

- **VGG models** [12][13] are very simple and clean. There are two versions for the VGG model namely VGG16 and VGG19. Fig.[2] shows how simple architecture VGG models have. The only difference between VGG16

and VGG19 is the inclusion of Three more layers in VGG19. Simple architecture of VGG facilitates fast propagation of gradients when it is updating the input weights of the generated image.

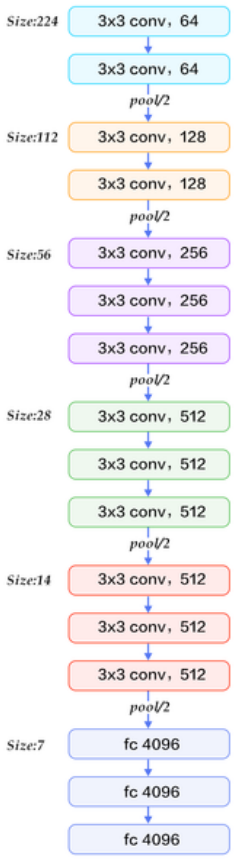

Figure 2: VGG-16 Model [12]

- **Inception (GoogleNet)**[16] [15] Inception has 22 layers, but the architecture has lost its simple complexity compared to previous models (fig.[3]). The complex layers are introduced to reduce the memory size and parameters calculated. We can see this by the pre-trained models provided to us. VGG-19 pre-trained model size is 516mb, where as inception model trained on same data set has 26.6mb. This shows that the memory consumed by inception is almost 12x less than VGG model. Inception model also has successfully removed fully connected layer that can be seen in right most part in the fig.[2]. Fully connected layers is costliest part of the architecture in comparison on parameters calculation.

Figure 3: Inception -v1 model [16]

- **ResNet** consists of 152 layers. ResNet introduces many new problems to the general assumption such, as the model goes deeper the performance on test increases. Also, the method provided by ResNet was further optimized with variation of inception in upcoming years. For this project, we will be using variant of ResNet and **Inception -v3**.

We will be writing our whole code using TensorFlow frame work for this project, and the language of choice will be python.

For the dataset we will be using pre-trained data model like the above mentioned.

4 Style Transfer

Style Transfer [1] [7] in the layman term is an attempt to reduce the loss between the content image activation map and generated image activation map extracted from specified layer of the model to which each image is passed. Also, simultaneously

doing the same with style image and generated image. The attempt is to reduce the loss overall and to makes sure that the generated image is updated as the loss(loss with style image + loss with content image) is minimized. Another way to put the above sentence is that we are looking for low frequency features from the content image, and high frequency features from the style image to be on the Mixed image.

Gram Matrix, we calculate gram matrix for the style and generated image to find the style loss. We make an attempt to reduce content and style loss, which will make the image resemble to both like we can see from fig.[1]

$$GramMatrix = \sum_{N}^{i} M \times M^T \tag{1}$$

Cross entropy is the loss function that we usually use, but beside cross entropy we have other loss functions like square mean loss(), sigmoid loss,softmax loss,log loss(),huber loss().

Challenges we face doing the style transfer is selecting the best model from (VGG-16, VGG1-19, Inception-v1, Inception-v3/v4). The reason we face challenge in selecting the model is because of the complexities of the models, **loading time for the models,back propagation duration, Optimization methods for the model, Number of iterations, weights for the loss that we need to experiment.**

The following figures in this project shows that the output from any model would be different even if we try using same hyper-parameters, because of this reason we will do the experiment on different models by using different loss methods, and optimization methods[9]. The result would be compared against each other.

Figure 4: Style Transfer with VGG-19 using Adam optimization

4.1 Experiment With VGG16

Experiment List

- **Layers** The layers we selected for VGG-16 for content loss is **conv4_2** and for the style loss are **conv1_1, conv2_1, conv3_1, conv4_1, conv5_1**
- **Optimization Methods**

Methods	Lowest-Loss	Iteration	Loss at 500th
Adadelta	21.974645	500	21.974645
Adagrad	1.450987	500	1.450987
ADAM	0.069237	500	0.069237
RMS prop	0.078371	459	0.089102
LBFGS	0.056519	500	0.056519
Momentum	0.06688	459	0.131853

Things to report

- Plot with images from different Optimization

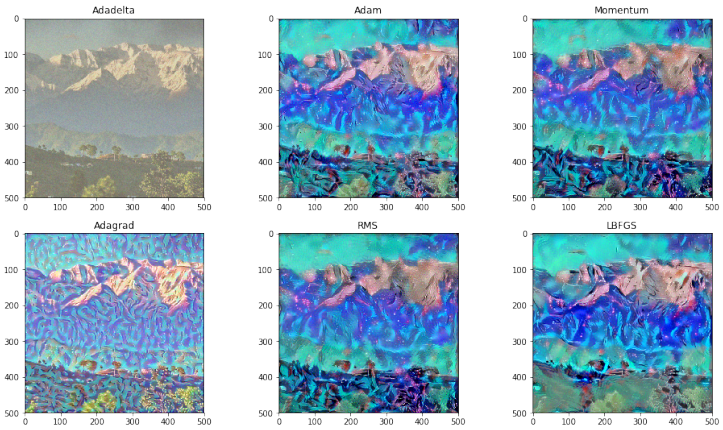

Figure 5: Style Transfer with VGG-16 using different optimization

- Loss Graph

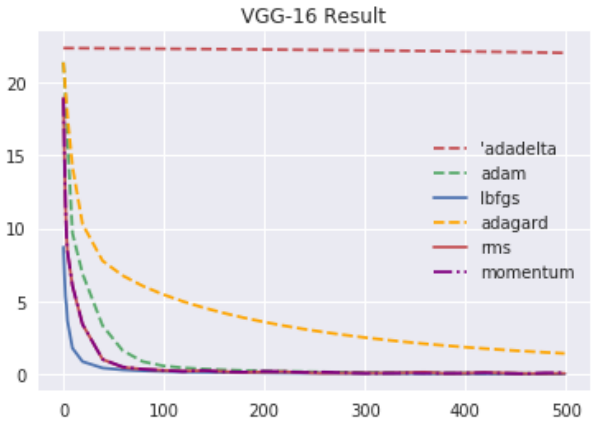

Figure 6: Style Transfer Loss graph with VGG-16 using different optimization

- Best Result

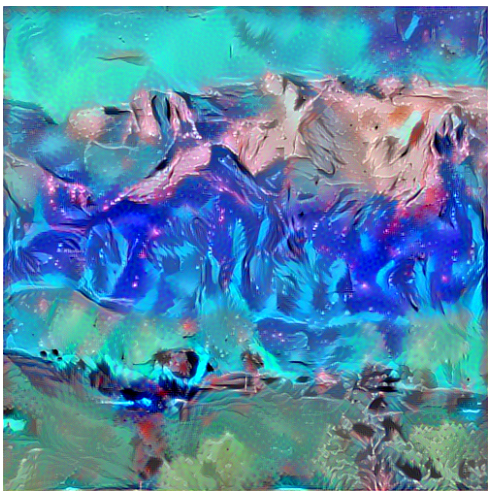

Figure 7: Style Transfer result with LBFGS

- Result

  From the loss graph and from the image result we can conclude that Adagrad and Adadelta are the worst performing optimization techniques for VGG-16. We also observe that RMS and Momentum have alomst equal performance, and their plot seem overlapping. **LBFGS** is the best optimization method, but depending on someone's taste one can argue **Adam** has better result.

4.2 Experiment With VGG19

Experiment List

- **Layers** The layers we selected for VGG-16 for content loss is **conv4_2** and for the style loss are **conv1_1, conv2_1, conv3_1, conv4_1, conv5_1**

- **Optimization Methods**

Methods	Low'-Loss	Iter	L'500th
Adadelta	33.90	500	33.90
Adagrad	2.65	500	2.65
ADAM	0.20	500	0.20
RMS prop	0.388	379	0.602
LBFGS	0.312	500	0.312
Momentum	139.6	0	139.6

L'500th - Loss at 500th
Iter - Iteration
Low'- Loss - Lowest Loss

Things to report

- Plot with images from different Optimization

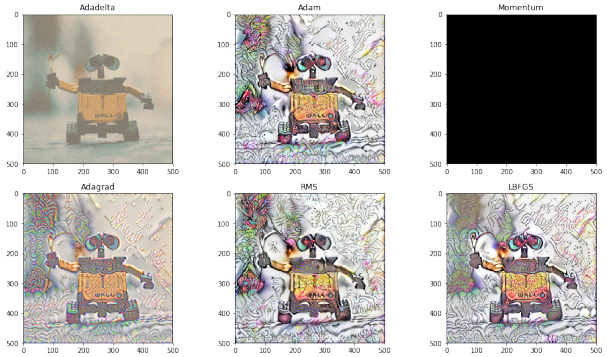

Figure 8: Style Transfer Loss graph with VGG-19 using different optimization

- Loss Graph

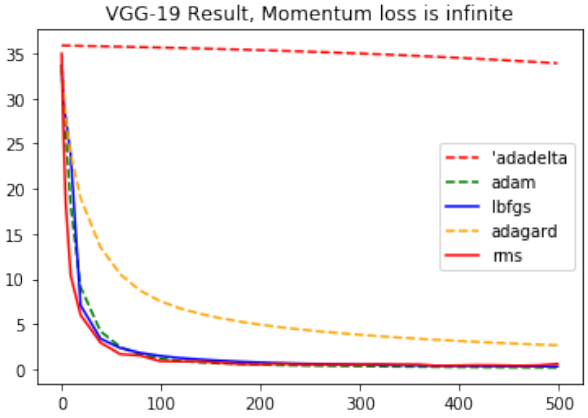

Figure 9: Style Transfer Loss graph with VGG-19 using different optimization

- Best result from the experiment

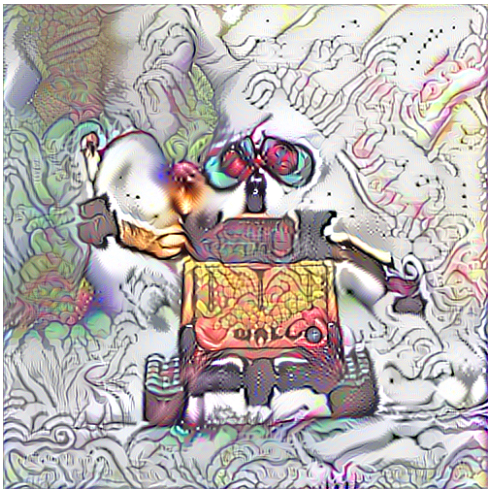

Figure 10: Style Transfer result with LBFGS

Figure 11: Style Transfer result with LBFGS

- Result

  Momentum seems to be having exploding or vanishing gradients problem if compared to VGG-16. This could be from the learning selection. Leaving out momentum irregularities, the result match to the result of the VGG-16 experiment. LBFGS[9] seems to be producing better and smooth result compared to other optimization techniques.

4.3 Experiment With Inception -v1

Experiment List

- **Layers**
  Content Layer is **Mixed_4b**
  Style Layer are **'Conv2d_1a_7x7', 'Conv2d_2c_3x3','Mixed_3b','Mixed_4d'**
- **Optimization Methods** We had performed experiment with different optimization methods, and for some reason it seems that only **LBFGS**[9] is able to produce the result. We will try infer the result as per our understanding.

Things to report

- Perfect result from the experiment
- Result

  It is tough to infer what exactly is happening, but from the experiments it seems that the Adam [9] and rest of the optimizer do have constantly decreasing loss but there is no sign of style gradients on the noise image. Only LBFGS [9] seems to be able to produce the result, but the result still lacks proper gradients from the style image.

5 Discussion

- **Best optimization technique** We can conclude that the test result do indicate that LBFGS [9] is the best optimizer for style transfer which agrees with the author of the neural style transfer paper [1], Dr. Gatys. LBFGS is able to produce the result with lowest loss value and very smooth images.
- **Shallow model or deep nets** We performed experiments with shallow model like VGG, and deep nets like inception. It seems that the deep nets do lack good amount parameters which contribute to the gradient information (Style) which we need on the noise image. Whereas VGG have good amount of information. VGG trained model are nearly 500mb in size compared to the Inception model which is 12x smaller than the VGG model. The other irregularities that we observe in deep nets is the usage of stride 2, It does makes the output image blurry and the image have checkered effect.
  We can conclude that given anytime VGG seems to be out performing deep nets in the art of style transfer.

References

[1] Gatys, Leon A., et al. "Image Style Transfer Using Convolutional Neural Networks." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, doi:10.1109/cvpr.2016.265.

[2] Gatys, Leon, et al. "A Neural Algorithm of Artistic Style." Journal of Vision, vol. 16, no. 12, 2016, p. 326., doi:10.1167/16.12.326.

[3] Lengstrom. "Fast-Style-Transfer." GitHub, 22 Jan. 2018, http://github.com/lengstrom/fast-style-transfer.

[4] Desai, Shubhang. "Neural Artistic Style Transfer: A Comprehensive Look." Medium, Artists and Machine Intelligence, 14 Sept. 2017.

[5] Hvass-Labs. "Hvass-Labs/TensorFlow-Tutorials." GitHub, 9 Dec. 2016.

[6] Tejani , Shafeen. "From Bits to Brains." Https://Shafeentejani.github.io/, Github, 3 Jan. 2017, http://shafeentejani.github.io/2017-01-03/fast-style-transfer/.

[7] Singla, Sahil. "Getting Inception Architectures to Work with Style Transfer." Medium, ML Review, 29 Aug. 2017.

[8] "CS 20: Tensorflow for Deep Learning Research." Stanford University: Tensorflow for Deep Learning Research, http://web.stanford.edu/class/cs20si/.

[9] Ian Goodfellow and Yoshua Bengio and Aaron-Courville.http://www.deeplearningbook.org,chp-8

[10] Author-Dawars,discussing style transfer experiments with different models. https://dawars.me/neural-style-transfer-deep-learning/ last-visit-04/26

[11] Source - Reddit https://www.reddit.com/r/MachineLearning/comments/7rrrk3/d_eat_your_vggtables_or_why_does_neural_style/

[12] VGG-16 architecture by Davi Frossard https://www.cs.toronto.edu/~frossard/post/vgg16/ https://www.quora.com/What-is-the-VGG-neural-network

[13] VGG-19 architecture for tensorflow taken from github (fchollet) https://github.com/fchollet/deep-learning-models/blob/master/vgg19.py

[14] Deep Dream by Alexander Mordvintsev https://github.com/google/deepdream

[15] Inception maintained by google on github https://github.com/tensorflow/models/tree/master/research/slim

[16] Explanation of Inception Model,last access-04-26 https://hacktilldawn.com/2016/09/25/inception-modules-explained-and-implemented/