Project Proposal

Optimize and extend the function of A Neural Algorithm of Artistic Style

Chieh-Ju Lu
Department of Data Science
University at Buffalo
Buffalo, NY 14260
chiehjul@buffalo.edu

Abstract

Introduce the project "A Neural Algorithm of Artistic Style" [1] I chose for CSE 676 final project, and discuss the process that I propose a new model to improve the original model. I offered some methods combined with two courses, deep learning as well as the course computer vision and image processing, to modify the model.

1 Introduction

This part discusses the reason why I chose "A Neural Algorithm of Artistic Style" [1] as my project, and briefly reviews the concept of it. Also, I listed some ideas about what methods I would try to improve the model.

1.1 Motivation

In this semester, I also take a course about computer vision and image processing. So I'm interested that how can I use the methods I learn in that course to combine with deep learning, and that's the reason I chose this topic. I think it is brilliant that this paper uses neural network models to distinguish style and content, then extracts the style to apply to a photograph. I'm not sure if I can provide an obvious improvement on this project, but I would like to try everything I learned this semester and figure out what will the model change if I apply any other image processing methods.

1.2 Brief review

The paper I chose "A Neural Algorithm of Artistic Style"[1] is about combining two different images to create an image with a brand new view using deep learning skills. In this paper, they use VGG (Visual Geometry Group) Network which is a Convolutional Neural Network architecture to fulfill this function, and found that the representation of content and style are separable. There are two input images, one is an art image, another one is a photograph we want to change its style. The model captures the texture information of the art image, and preserves the content of the photographic image. Then the model will combine these two images with the original content and the style. As the hidden layer increasing, the more similar between two input images and the output image. The most significant contribute of this paper is providing algorithmic understanding of how to capture the content and the style of an image separately through neural representation.

1.3 Ideas

First of all, I'm interested in the VGG network, according to the paper [1], it can rival humans' performance on a common visual object recognition benchmark task, yet it is also quite slow and need to take a large number of resources. I would like to use different ways to implement this model, including modifying regularization and activation functions, apply different models, and use different loss functions.

Besides, I will also try to apply other methods from the course computer vision and image processing to deal with the image, maybe there are some methods I can extract features and styles more efficiently.

2 Methods

2.1 Description

I separate my implementation process into three parts, which is also the order I try to optimize the model, these steps are original model implementation, improving methods with image, and proposed model. I briefly introduced the model from source code [2], and some issues when I implemented it in the beginning. Then, I discussed some methods I tried to apply to model, and explain the reason why I chose them.

2.2 Original model implementation

To start with, I used the source code from [2] and try to use the test case author provided to present the result.

However, there are some issues when I tried to execute the code, and I modified some parts of it in order to run the model. For example, I made some changes to the library used in the code. The user [2] used scipy.misc to read as well as resize images, but scipy.misc has been deprecated in Scipy, and these functions are not available in the new version of Scipy. Therefore, I use cv2 to replace scipy.misc to read the image and do the resize.

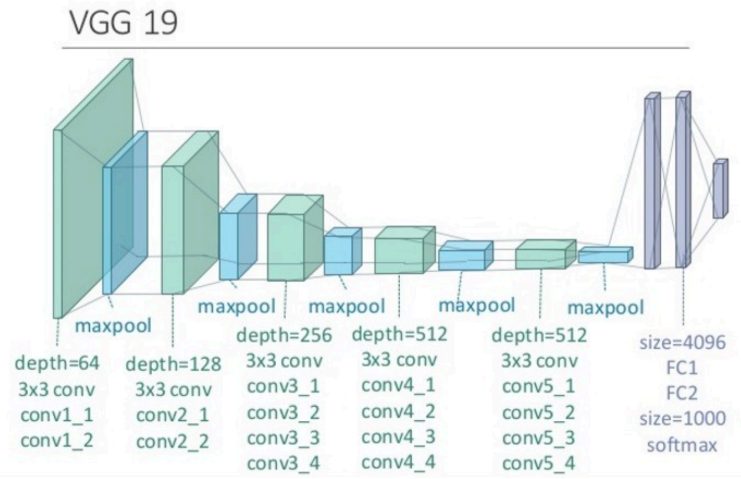

Figure 1: VGG19 structure [3]

In this code, they use VGG19 model without 3 fully connected layer, which only including 16 convolutional and 5 pooling layers, to do the style transformation. Each convolution layer applied ReLU as activation function then do max pooling to reduce the computational cost. The model can do content reconstructions and style reconstructions at the same time.

2.3 Improving methods with image

First of all, according to the paper [1], they capture style from the color and localized structures of input art image, and capture content from the photographic image. I think that we can try to transfer the photographic image into gray-scale, in which the image matrix will change from height*width*3 into height*width only in order to decrease the parameters in our models.

However, this method didn't work since the VGG model's architecture cannot be changed. If I remove the first layer, it will fairly affect the performance of the model because the next layers cannot get the value they need as their input. If I replace the first layer with random weights, this also affects the training of other layers, meaning that the VGG model may not able to do the training.

Besides, I would like to use some feature extract methods to replace the loss function, I thought that if I can use SIFT (Scale Invariant Feature Transform), which divides the image into 16 patches, then computes a histogram of gradient orientations for all pixels inside each sub-patch, to extract the important features, then use Euclidean distance to compute the distance between these features, this may provide more accuracy loss between two images. However, some feature extraction model like SIFT, even if they can catch some significant features in an image, it is intractable for gradient descent. Hence, for the deep learning model, this method cannot provide any benefits because unable to do training.

2.4 Proposed model/algorithm

First of all, I tried to replace max pooling with average pooling for decreasing the loss. Max pooling will select the maximum value of the current view, which is the brighter color in the image. Even if max-pooling can extract important features like an edge from the image, the image will be sharper. Hence, I assumed that average pooling will be smoother than max-pooling so that we can decrease the variance between the original image and output image, which can decrease the content loss.

Also, I noticed that VGG 19 use ReLU as the activation function, the computation expensive of ReLU is less than tanh and sigmoid because it just needs to determine if the value is greater than zero. However, in some cases, ReLU could have "dead" problems which cause the learning rate cannot to be updated. I'm not sure if this model has that issue, but I would like to use leaky ReLu to displace ReLU in order to figure out if this change can decrease the loss. In addition, I will also try to apply tanh as a substitution of ReLU to know if this affects the performance of the data.

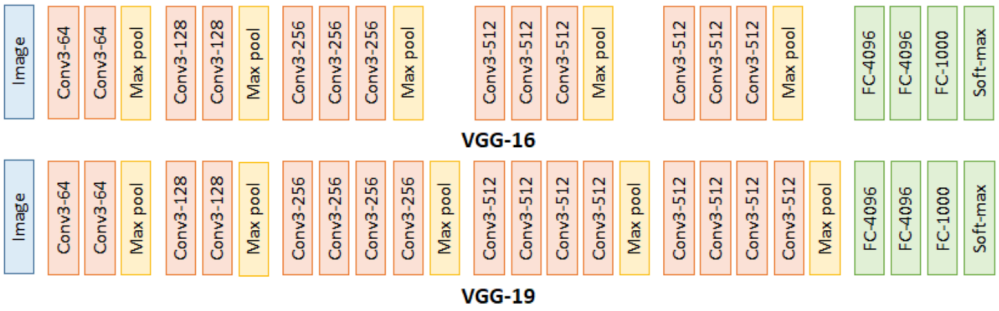

Figure 2: VGG16 and VGG19 [4]

Furthermore, I would like to use VGG16 instead of VGG19 in this model to decrease the execution time, and want to understand what will be different if I do the replacement. VGG 16 essentially is the same structure as VGG19, yet less three convolution layers. They also apply ReLU as an activation function in each convolution layer and use max-pooling layers to decrease the computation expensive. Then apply three fully connected layers and soft-max to present the probability of classification. In this style transformation model, we don't need to use fully connected layers, so I will remove them in my model.

3 Result

3.1. Compare loss between original model and modified model

The following table is the performance with different methods I try to apply to the model, each case used the same input style image and photographic image from the source code[2], and the iteration is 1000. The loss including content loss, style loss, and tv loss. Content loss is the MSE of the features of the original image and output image. Style loss is the style dissimilarity between the style image and output of each layer. And the tv loss is the total variation loss that encourages spatial smoothness in the generated image.

	Original model	**Modifying model**			
Different methods		**Average pooling**	**Leaky ReLU**	**Tanh with the final layer**	**VGG16**
Runtime	37 min 59 sec	36 min 18 sec	37 min 40 sec	34 min 53 sec	31 min 34 sec
Content loss	864545	84554.8	1.65628E+06	84857.9	45027.2
Style loss	245190	34940	566431	32754.3	9680.14
TV loss	82180.1	33135.6	60313.7	32841.8	50232.6
Total loss	1.19192E+06	152630	2.28302E+06	150454	104940

Table 1: Performance with different modified models

For changing max pooling, although average pooling increased the style loss, the content loss actually decreases as expected. Also, the tv loss is decreased. The total loss decreased 87.1946 %, which seems good. Yet the performance of using Leaky ReLu causes more content loss, which means it cannot preserve the content feature in the original image, so I have to use other methods to change the activation function for better performance. Thus, I tried to apply tanh just in the final layer in order to preserve enough features in style and content in previous layers. The result seems pretty well with tanh, the model runtime decrease, and the style loss as well as tv loss both perform better than average pooling. The total loss in decrease by 87.37717% in the original model.

Moreover, I tried to use VGG16 structure instead of VGG19 in order to decrease the runtime, and the result looks well more than I expected, because the content loss actually decreases 94.7918% than the original model.

Overall, I found that when I replaced the ReLu with tanh in the final layer will have the least style loss as well as tv loss, and VCC16 structure provides the lowest runtime and content loss. Therefore, I decided to apply those two methods at the same time to know if I can make the model present both improvement. The following table is the comparison of the original model and the model that used VGG16 combined tanh:

	Original model	Modifying model (VGG16 + tanh)
Runtime	37 min 59 sec	31 min 56 sec
Content loss	864545	44875.3
Style loss	245190	10385.8
TV loss	82180.1	50380.9
Total loss	1.19192E+06	105642

Table 2: Compare the performance between original model and new model

According to table 2 above, we can find that the performance actually good as assumed. The modified model performs less runtime, and decreases all losses. While the tv loss cannot as low as either one of the separated modified models, the content loss and the style are lower than any models I tried before (table 1).

3.2 Compare result images

Figure 3: Input images from source code

The images above are downloaded from the source code, the left is the photo of the author, and the right image is the style image The Starry Night by Vincent van Gogh, 1889.

Figure 4: Images that combined style and content with different modified models.

The left image is from original model; the middle one is the model applied average pooling; the right image is the model with Leaky-ReLU.

For the model with changing max pooling to average pooling, the performance in total loss in Table 1 seems good; however, the output image is more blur than the output of the original model. I considered the reason is that average pooling cannot preserve significant features of the input photo, so it lost some content information. Hence, it is hard to evaluate if this is an improvement of the original model. Then, for another model apply Leaky-ReLU in every convolution layer, we can find that the color of floor tiles becomes single instead of gradient, also the wall is distorted. While this model has double content loss than the original model, the

resulting image surprisingly seems pretty well. I think that the resulting image is clear and perfectly performs the style.

Figure 5: Images that combined style and content with other different modified models.

The left image is from the modified model which apply tanh in the final layer; the middle one is the model with VGG16; the right image is the model with VGG16 structure and apply tanh in the final layer.

In addition, we can find that while the model applied tanh in the final layer perform well in its loss, the output image seems pretty worst due to blur and some noise in the image. We can find that the performance in loss cannot always represent the quality of the output image. Fortunately, the output images with VGG16 model seem well, it shows more focus on style than the content of the original photo but there is no strange noise in the resulting image. Same as the final model which uses VGG16 and tanh as the final layer activation function, we get a complete output image.

4 Conclusion

In fact, it is difficult to judge the quality of the model in the topic of style transformation, because the result of the transformation is quite subjective. Even if the total loss be reduced after modifying, the resulting image could be quite blurred. Hence, I cannot make sure that the proposed methods are actually feasible because is no certain standard for optimizing the model. However, I still learn a lot during this process, the knowledge I learned in deep learning is mainly focused on the theory, there are still several challenges when I tried to implement it. We need to consider lots of factors to do a small modification in a neural network structure, and the result will not always perform as expected. I do some research about style transformation on the Internet, and they're actually various brand new methods on this topic. Some people offer the model to transfer video style, and there are also some faster models like "Fast Style Transfer in TensorFlow" [5] which use the same VGG19 structure but different layers. I think those methods are really impressive, and it is really tough to achieve the same level, yet I have a chance to learn how to optimize a model.

References

[1] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576, 2015.

[2] Anish Athalye. (2015-2019) Git hub: https://github.com/anishathalye/neural-style. Released under GPLv3

[3] Mark Chang. Applied Deep Learning 11/03 Convolutional Neural Networks: https://www.slideshare.net/ckmarkohchang/applied-deep-learning-1103-convolutional-neural-networks

[4] Sik-Ho Tsang. Review: VGGNet — 1st Runner-Up (Image Classification), Winner (Localization) in ILSVRC 2014: https://medium.com/coinmonks/paper-review-of-vggnet-1st-runner-up-of-ilsvlc-2014-image-classification-d02355543a11

[5] Logan Engstrom. Fast Style Transfer in TensorFlow: https://github.com/lengstrom/fast-style-transfer