Neural Style Transfer for Image and Video

Yifan Wang
Robotics
University of Pennsylvania
Philadelphia, PA 19104
yyifan@seas.upenn.edu

Zilin Ren
Data Science
University of Pennsylvania
Philadelphia, PA 19104
zilinr@seas.upenn.edu

1 Project Summary

Neural Style Transfer (NST) is a very interesting application of Deep Learning. The basic idea is to extract content and style features from content and style images respectively, and recombine these two features into a target image. Afterwards, the target image is recursively reconstructed online, based on the differences between the generated image and the content and style images. For this project, we implement Neural Style Transfer (NST) that begins by producing artistically-styled photos and then move on to videos. We experiment on three different optimizers: LBFGS, SGD, and ADAM on images and videos. We also experiment with a temporal loss to reduce the problems from generating videos.

2 Related works

NST is a category of algorithms developed to take an input image and reproduce it in accordance with a predetermined artistic aesthetic. The first such algorithm that offered a quick and effective approach to creating stylized images was the Neural-Style algorithm. Gatys shows how high-performance convolutional neural networks can be used to transfer image style between arbitrary images, resulting in high perceptual quality separation and recombination of the natural images' content and style [1]. Conceptually, it is a texture transfer algorithm that first applied VGG network and Gram matrix.

But the method of Gatys cannot capture the long-term correlation of the images. At the same time, the method of using the Gram matrix to represent style features has limitations in terms of stability and texture quality. In addition, the above method only extracts the high-level features of the image and discards the low-level information, which causes the loss of the details in the stylized image. Moreover, the algorithm mentioned in [1] does not consider the stroke change, semantic information, and depth position information of the image, which will lead to unreasonable stylization. Therefore, follow-up studies have made improvements to the shortcomings of Gatys's method.

Berger and Memisevic et al.[2] improved by adding Markov structure to the features, the generated image can satisfy long-term consistency, which can be used for texture generation with global symmetry and the transformation of image seasons. Pei Wang found the lack of robustness of the pre-trained VGG network [3] because residual connections produced feature maps of small entropy, which were not suitable for style transfer. They proposed a simple yet effective solution based on a softmax transformation of the feature activations that greatly improve the quality of stylization results. Focusing on low-level information in content images, Li et al. [4] introduced Laplacian loss to impose additional constraints on low-level features. The high-level semantic information extracted in the VGG network is supplemented in detail by additionally describing the low-level information in

the content image using the Laplacian matrix. Subsequent research introduces semantic information to enhance control over the generated images. Luan et al. [5] mapped the semantic features of content images and style images one by one through artificial control, so that style transfer occurred in sub-regions with the same semantics, preventing the overflow of styles in each region. Penhouest and Sanzenbacher [6] improved on Luan et al. [5] and introduced automatic segmentation of image semantics to simplify the workflow.

3 Methods

Based on the papers we read, we implement the code to perform Neural Style Transfer. Our code could be mainly divided into four parts: preprocessing the input image, defining the loss function, building the model, and training with different optimizers. We have a content image as input image, and a style image that used to define the style of the output image. VGG-19 is the base Neural Network, and we embedded it with accumulated style loss and content loss between certain convolution layers.

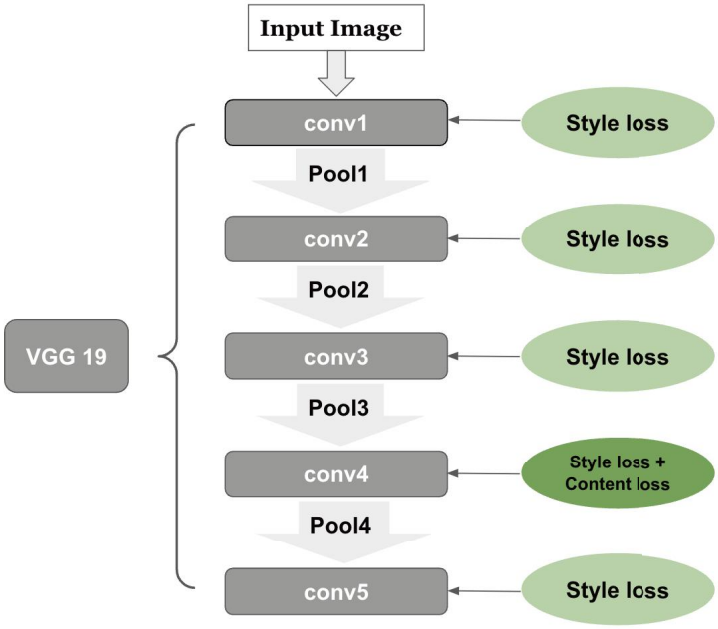

Figure 1: The structure of the neural network

The above steps are introduced more specifically below. First, we utilize a VGG-based neural network to represent the image embeddings. We also use some techniques to enhances stability and robustness of the stylization quality such as trying different optimizer among LBFGS, Adam, and SGD. Then we design two loss functions: content loss and style loss to measure the content and style discrepancy of training image to the reference images. Specifically, for style loss we use mean square error and the gram matrix to extract style patterns. For content loss, we also use mean square error to identify the differences between the new transfer-styled image and the original image. We come up with an approach that can quantify the generated image quality based on the comprehensive consideration of style and content loss, given that we would first eye-ball the image to decide on the weights assigned to each loss. Then we try different proportion of content and style loss to see the outcome of the style transfer, which leads to a better visual experiences of the transferred image. After coding the main structure, we train on small images at first and move to larger image with

optimized calculations, and finally transfer to videos.

4 Experiments and Results

The most important outcome of this project is that we separate the representations of content and style in the convolutional neural network and manipulate both representations independently to generate a series of new, style changed images. To further improve the visual effect of the transferred image, we compare the different loss weights and optimizers. Then, we extend the image style transfer to the video style transfer and improve it to obtain the robust outcome.

4.1 The Effect of Loss Weight

The effect of the fusion between the style and the content differs case by case, which indicates that the performance of the code depends on the adjustment of the parameters, especially the proportion of style and content loss. When Figure 2 and Figure 3 are used as the content image and style image respectively, the output image Figure 5 has a great visual effect as the style weight is 1000000 and content weight is 1. In the meantime, for merging the content image (Figure 2) and style image (Figure 4), the outcome in Figure 6 shows that the style would overlay the content when the proportion parameters set as the previous case. To address this problem and obtain a reasonable transferred image, we change the weight of the style loss to 50000, which produces a much better outcome as shown in Figure 7.

Figure 2: The content image

Figure 3: The style image 1

Figure 4: The style image 2

Figure 5: The merged image of content image and style image 1

Figure 6: The failed merged image of content image and style image 2

Figure 7: The successful merged image of content image and style image 2

4.2 Optimizers Experiments on Images

Another important factor that influences the style transfer effects is the choice of optimizer. We tried three optimizers: LBFGS, ADAM, and SGD for updating the image. We took the average style loss and content loss of 100 rounds where each round updated the original image 300 times, and the losses for each optimizer are as follows:

Optimizer	SGD	Adam	LBFGS
Style Loss	1.2306	2.4756	2.4851
Content Loss	14.7295	17.9835	19.9371

Table 1: The optimization results of different optimizer

By comparing the style and content loss, we would find there isn't much difference among them. Simply from looking at the resulting image, we find out that LBFGS optimizer has the best performance among the three, which produces a more vivid visual experience while maintaining the originality of image.

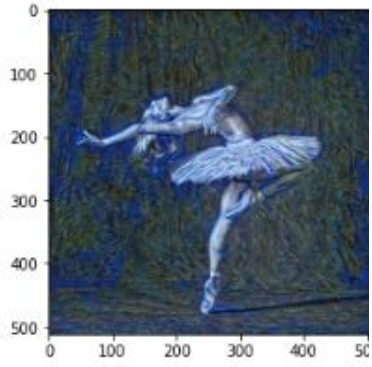

Figure 8: The visual result for SGD optimizer

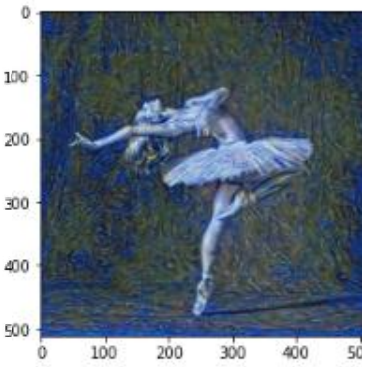

Figure 9: The visual result for Adam optimizer

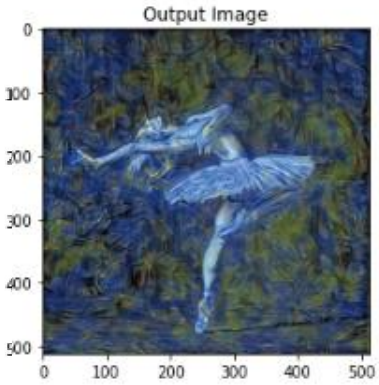

Figure 10: The visual result for LBFGS optimizer

4.3 Consistency between Frames for Video Style Transfer

We start from generating the video as a collection of images, train them individually, and combine them together as a video. When we use the LBFGS optimizer, we observe that the video has some obvious discontinuities between frames, which are shown as "flashes" in the video. This is also reflected by the high content loss of models trained by LBFGS optimizer. Comparing Figure 11, Figure 12 and Figure 13, the background color of Figure 11 and Figure 13 is darker than Figure 12, and the strokes and edges are quite distinct among them. These color and stroke changes resulted in flashes in the video.

Then, we tried to use Adam optimizer. The video no longer has flashes, and the strokes become much clearer and obvious than before (see Figure 14, Figure 15 and Figure 16). However, the change of background strokes is quite drastic even when the background is relatively static in the original video.

Finally, we switch to SGD optimizer. The resulting video preserves the style traits of the style image, and there are no flashes. Compared to Adam optimizer, SGD provides less obvious changes of static background, though the problem still exists.

In order to solve the changing background problem, we try to add a temporal loss in the model. It compares the change between the previous frame and current frame, including the difference of content and style with designated weights assigned to each. The resulting video gives quite interesting result. The background has minor changes and relatively static compared to previous ones, but the

Figure 11: The visual result for LBFGS optimizer 1

Figure 12: The visual result for LBFGS optimizer 2

Figure 13: The visual result for LBFGS optimizer 3

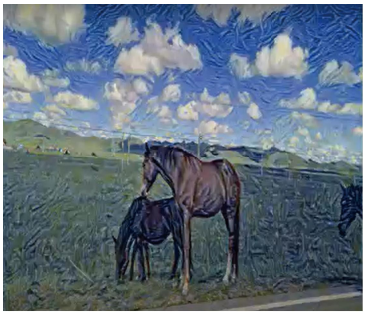

Figure 14: The visual result for ADAM optimizer 1

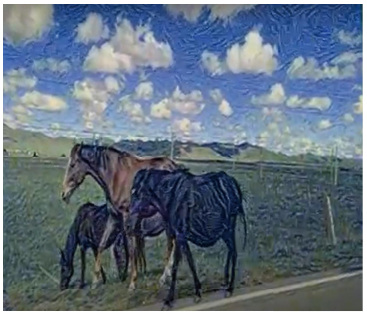

Figure 15: The visual result for ADAM optimizer 2

Figure 16: The visual result for ADAM optimizer 3

moving objects are blurry with "shadows" from the past frames as shown in Figure 18 and 19. Thus, it may be better to simply apply this temporal loss to the static parts of the video, while training the moving objectives without it.

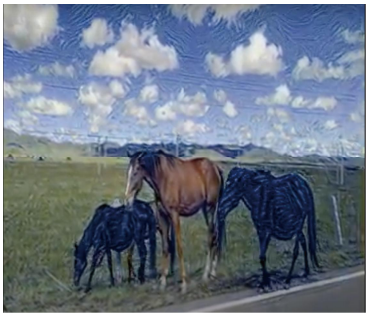

Figure 17: The visual result for SGD optimizer generated video

Figure 18: The visual result for SGD optimizer generated video with temporal loss

Figure 19: The visual result for LBFGS optimizer generated video with temporal loss

5 Conclusion and Discussion

5.1 Results

The ratio of weights for style and content loss will impact the style-transferred images. Thus, we need to adjust the ratio between them according to input images. In the future, it will be promising to come up with some automatic weight adjustment algorithms according to the transferred image performance.

Comparing the three optimizers: SGD, ADAM and LBFGS, we find out that LBFGS provides pretty good style transfer for image, but it provides non-stable style-transferred videos. On the contrary, SGD and ADAM provided more smooth transfer for video regarding its color stabilization and moving object's motion change.

We used a temporal loss for preserving some commonalities between neighboring frames in training. In general, training with this loss function blurs the moving object along its motion trajectory, but reduces changes of the static background. It is better to separate moving object from static background if we want to incorporate the temporal loss in training.

5.2 Learning Experiences

By completing this project, we have a deeper understanding about the image and video style transfer using convolutional neural networks. These are the main takeaways:

- Learn modern computer vision tasks and able to implement from research papers.
- Deliver a comprehensive academic report about the whole process with solid codes.
- Apply Machine Learning to the project such as building Neural Network with Pytorch.
- Learn more state-of-the-art technology in Neural Style Transfer.
- Apply the Style Transfer technique to image then video in the end.

5.3 Future directions

Although we have already addressed basic style transfer for image and video, there are still some improvements to do:

- First, different style loss and content loss functions are worth trying, which may lead to a better separation of the images' style and content. Some other loss functions, such as L1 loss and Huber loss, are possible choices.
- Second, an algorithm for auto-adjusting weights between style loss and content loss might be designed according to the scores that describe the performances of these two aspects. This could be instructive to produce images and videos with the best visual experience.
- Third, other kinds of base neural network besides VGG are worth to be attempted such as ResNet and AlexNet. And we can also further adjust the structure of our trained neural network. So far, the style loss is embedded in between each convolution layers, while the content loss calculation only appears before the last convolution layer.
- We can try to adjust the input image size given the original size instead of setting a fixed set of values. And we should try on different size of images, and try to capture more details in larger images.
- We tried using Automatic Mixed Precision (AMP) when training large-scale images, but we had some problems in successfully adopting it for our codes. The training does not converge in the end. This is still a good next-step to work on.

References

[1] L. A. Gatys, A. S. Ecker and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2414-2423, doi: 10.1109/CVPR.2016.265.

[2] Berger, Guillaume and Roland Memisevic. “Incorporating long-range consistency in CNN-based texture generation.” ArXiv abs/1606.01286 (2017): n. pag.

[3] Wang, Li, and Vasconcelos. (2021). Rethinking and Improving the Robustness of Image Style Transfer. 124-133. 10.1109/CVPR46437.2021.00019.

[4] Li, Shaohua et al. “Laplacian-Steered Neural Style Transfer.” Proceedings of the 25th ACM international conference on Multimedia (2017): n. pag.

[5] F. Luan, S. Paris, E. Shechtman and K. Bala, "Deep Photo Style Transfer," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6997-7005, doi: 10.1109/CVPR.2017.740.

[6] Penhouët, Sebastian, and Paul Sanzenbacher. "Automated deep photo style transfer." arXiv preprint arXiv:1901.03915 (2019).