Deep Neural Networks for Artistic Style Transfer

Zhenyang Xu
Email: deanxu@seas.upenn.edu

CIS-5810 Computer Vision Final Project Report
University of Pennsylvania, Philadelphia, PA 19104

Abstract

With the popularity of deep neural networks in application of robotics, data science, and artificial intelligence fields, computer science technologies have been widely used in all aspects of nowadays society. People can now train deep neural networks to perform different kinds of tasks in industries to help to solve realistic problems with incredible fast speed and high efficiency[1]. One popular field of computer vision applying deep neural networks is called Artistic Style Transfer (AST). Different artworks usually have different artistic styles or genre, such as classic, cubism, impressionism, post-impressionism, pop art, surrealism, fauvism, art nouveau and so on. Different artistic styles express artworks with different techniques and there is no easy way for them to transfer between each other. In old days, transfer artworks such as oil paintings from one style to another usually requires an expertise artist to recreate the whole painting from scratch, which is time-consuming and money-consuming. Now with the aid of deep neural networks (DNN), we can train a specific model to perform such tasks. Given a content/original image and a target/style image we can use DNN to generate the output image with the same content of original image while deploying the same artistic style of the target image. In this work, we experiment with multiple different DNN models on both image input and video input to generate style transferred outputs. We followed previously work published by Gatys et.al[2] and make modifications to the whole process to adapt to more sophisticated network such as VGG16[3] and VGG19[3]. Hyperparameters for deep learning are tuned for optimization during training and testing phases of this project and the generated images and videos are presented in this work.

Introduction

Deep neural networks (DNN) in recent years shows its powerful application in large model building for numerical predication and categorical classification[1]. It has penetrated its influence into industries where big data processing and analysis are hard requirement such as quantitative analysis, data marketing, and financial database

analysis[4]. Many internet providers now expect they can generate images and videos in different styles in fast pace and large amount like tiktok or Youtube. Our research in this work provides a feasible way of generating image and video contents deploying different artistic style in a computational way. This will avoid people spending endless time editing these art resources.

DNN Architecture

Style transfer means transferring features and colors from original/content image into output image while keeping the same style with the style image. In our DNN architecture, we tested different DNN models which including ResNet18[5], VGG16[3], VGG19[3], InceptionV3[6], InceptionV4[7]. A pre-trained model is initialized at startup to speed up the training process. Among them we found the VGG19 gives the lowest training loss and generated reasonable output images compared with other deep neural network architectures. Possibly due to its most numbered parameters provide a more flexible model to be trained.

Loss Function

In the DNN which used in this work, a new loss function is constructed for performing artistic style transfer. The first part of loss function is called content loss[2]. This part loss computes the summed pixel-wise difference between the input content image and output content image. This part loss ensures us getting similar visual content between input and output content images.

The second part loss is called style loss[2]. This part loss computes the summed pixel-wise difference between the input style image and output content image. This part loss ensures us getting similar visual style between input and output style images.

During loss function computation, a technique called Gram Matrix[8] is used to compute the inner product space between a set of input image vectors: $G_{ij} = \langle v_i, v_j \rangle$, it's useful to elevate the input dimensions and perform nonlinear prediction or classification for deep neural networks.

During training, a total variation regularization term is introduced to the total loss function[9]. This regularization term function as the smooth factor to alleviate the abrupt changes between different boundaries and edges in output images. It basically computes the summation of absolute value of neighbourhood pixels in output images, minimization of this regularization term results in a more smoothed transition between boundaries and edges in output images.

Another regularization term we introduced to our loss function is called the photorealistic regularization term[10]. This term used the Matting Laplacian of the original content image times the output image vector to form a quadratic form: $\sum_i V_c[O]^T M_I V_c[O]$, where M_I is the Matting Laplacian matrix[1]. This term helps

preserve the original content in the content images.

So the final loss function can be expressed as follows:

$$\begin{aligned} Loss(\vec{C}, \vec{S}, \vec{O}) = \alpha L_{content}(\vec{C}, \vec{O}) + \beta L_{style}(\vec{S}, \vec{O}) \\ + \lambda_1 \sum_{i,j} |x_{i,j} - x_{i+1,j}| + |x_{i,j} - x_{i,j+1}| \\ + \lambda_2 \sum_{i} V_c[O]^T \cdot M_I \cdot V_c[O] \end{aligned} \tag{1}$$

The first term is content loss and the second term is the style loss as defined above. The third term characterizes the variation regularization term and the forth term characterizes the photorealistic regularization term.

For video input style transfer, we can split a video into a series of frames/images. A temporal loss is defined to compute the pixel-wise difference between two consecutive frames[11]. Minimizing temporal loss makes sure that the stylistic contents between two frames show some consistency as the video slide between consecutive frames. In order to achieve this effect, we apply temporal loss on all layers of the deep neural network architecture. To speed up training process for video style transfer, we also make some changes to the training process. Instead of training the whole sets of frames of video at one time, we train video frames in small batches much like the process for stochastic gradient descent. We do so for two reasons here: first, train whole video frames will make the training process slow and unstable, and second, train a small trunk/batch of consecutive frames will ensure the video frames have consistency.

Figure.1 shows the deep neural network architecture for artistic style transfer[2]. On the left, the features of content image and style image are extracted and stored in memory. Style image vector (labeled with $\vec{a}$) is propagated through style convolution deep network (on the left), each hidden layer is followed by a convolution layer and pooling layer. On the other hand, the content image vector (labeled with $\vec{p}$) is propagated through content convolution deep network (on the right). In the middle, a random white noise image (labeled with $\vec{x}$) is passed through the network to combine each layer's style image vector $\vec{a}$ and content image vector $\vec{p}$. An image pyramid is formed during training this deep neural network. Through minimizing the loss function defined in equation.1, we back-propagate the whole architecture to compute the gradients which can be used to update weights later. The output image is synthesized through overlapping all partial images within this image pyramid.

AST Examples

Figure.2 shows an example of artistic style transfer between different famous oil paintings[2]. Figure.2(a) shows the original/content images featuring some European building block near a river. Figure.2(b)-(f) show the output images generated through deep neural network with different target/style images (plot on the lower-left corners along with the output images). Figure.2(b): with style image as The

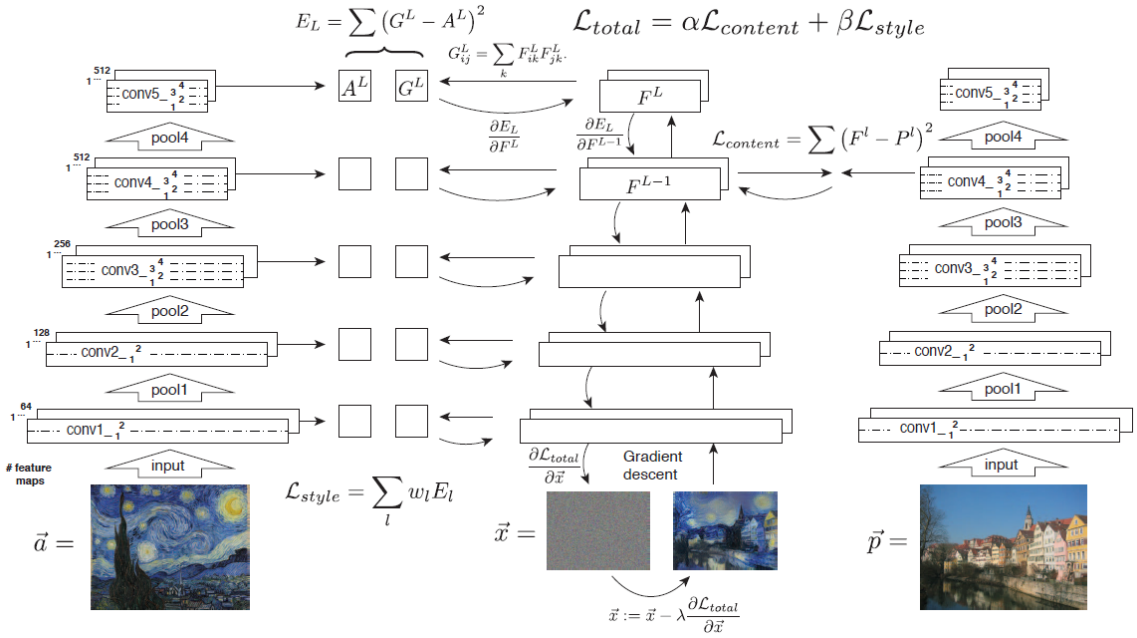

Figure 1: A figure showing the deep neural network architecture for artistic style transfer. Modified from[2]

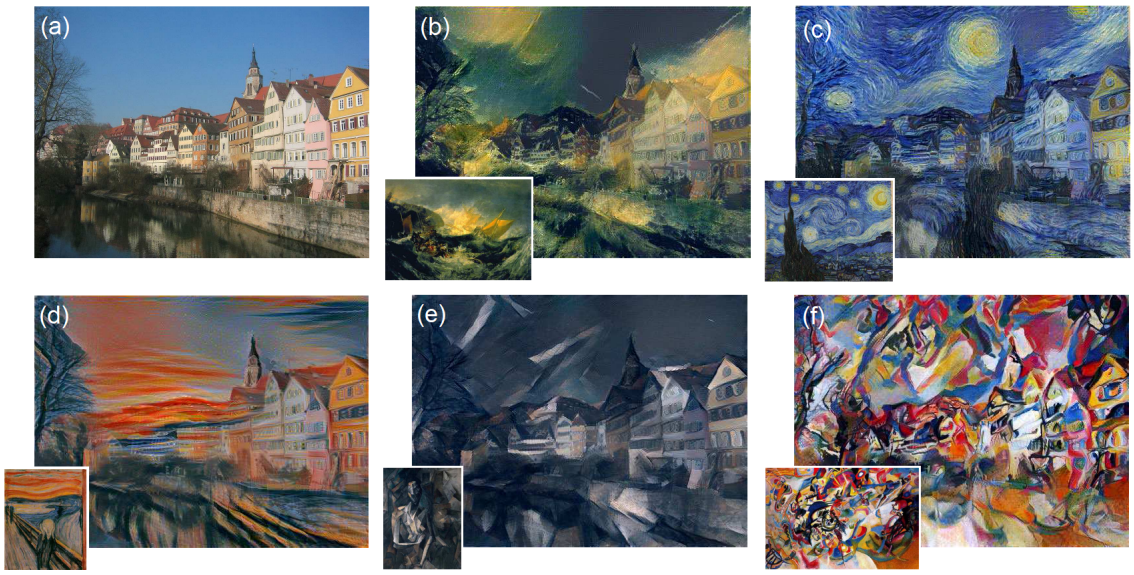

Figure 2: A figure showing examples of artistic style transfer between different famous oil paintings.[2]

Shipwreck of the Minotaur by J.M.W. Turner, 1805 (Style: Romanticism). Figure.2(c): with style image as The Starry Night by Vincent van Gogh, 1889 (Style: Impressionism). Figure.2(d): with style image as The Scream by Edvard Munch, 1893 (Style: Expressionism). Figure.2(e): with style image as The Seated Nude by Pablo Picasso, 1910 (Style: Cubism). Figure.2(e): with style image as The Composition VII by Wassily Kandinsky, 1913 (Style: Abstract Art).

We can see from these examples that artistic style transfer using deep neural networks is not simply extract the color spectre and texture from the original images and apply to the target images. The interplay of convolution layers within this deep neural network will transfer the original image from a view of whole, imitating the brush strokes, color tone, canvas layout of the style images and apply them into the content image. It is extremely useful for transferring plain-styled images into ones with strong artistic styles[1].

Experiments and Results

Figure.3 shows the artistic style transfer results generated from our own deep neural network. A VGG19 convolution deep neural network is trained to perform this task. In order to speed up convergence, a pre-trained model is used to initialize all the parameters and weights. Before training, the input image is resized as 256 $\times$ 256 and 1024 $\times$ 1024 respectively. During Experiment, 1024 $\times$ 1024 input takes much longer time to be trained compared to 256 $\times$ 256 input, probably due to the 4$\times$ times bigger number of parameters within the architecture.

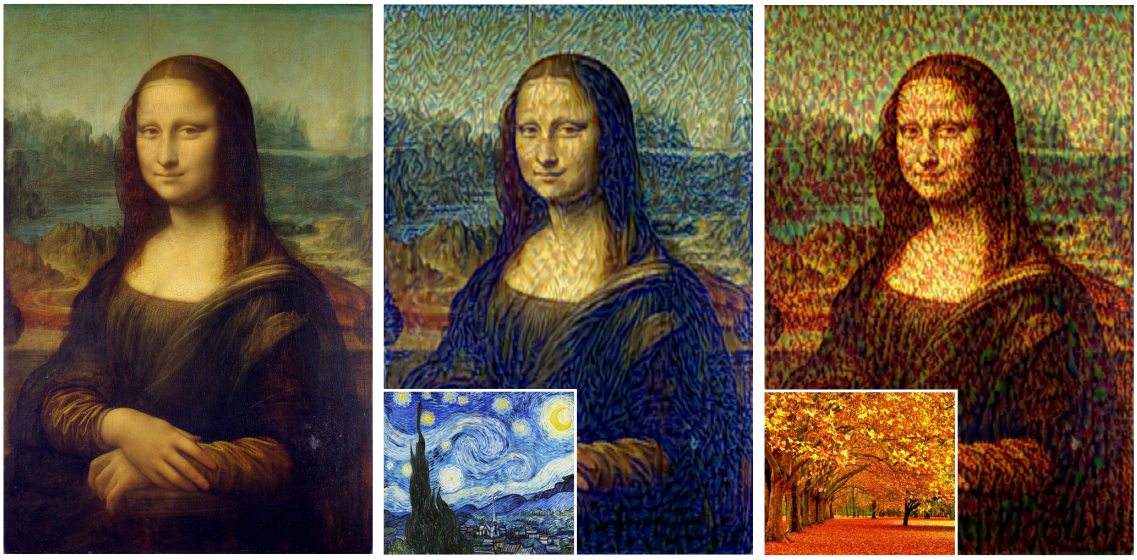

Figure 3: A figure showing the style transfer of Mona Lisa with different style image.

Different combinations of hyper-parameters of Stochastic Gradient Descent (SGD) also have been tried to optimize the whole process. For learning rare, we adopt 1e-5 as its value for this experiment. A larger learning rate value such as 1e-3 may cause the training loss drops a lot in magnitude but typical problems such as over-shooting and parameter oscillation also arises during training phase (shown in Figure.4(d)). If we choose a smaller learning rate value such as 1e-7, it will take the network much more number of epochs to reach to the optimal point so as to slow down the whole training process (shown in Figure.4(c)). Consider the limit of computational cost in time and memory, we choose the learning rate to be 1e-5 and train 100 epochs for each image. A typical training loss and validation loss against number of epochs is

shown in Figure.4(a) and Figure.4(b).

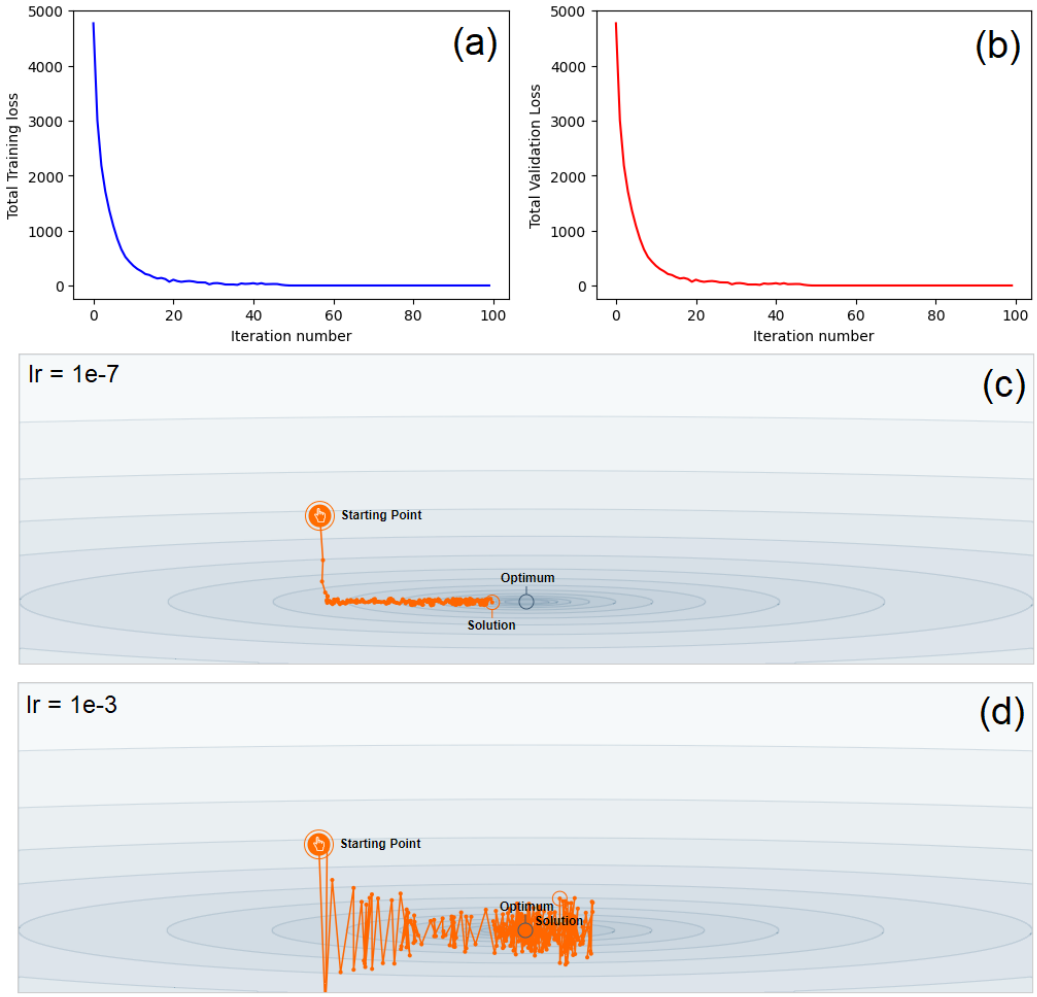

Figure 4: A figure showing the training results of AST in our deep neural network. Plot(a) and (b) show the typical training loss and validation loss curve as functions of number of epochs. Plot(c) shows a schematic plot of the under-training problem if a smaller learning-rate (such as 1e-7) is picked for this task and plot(d) shows a schematic plot of the over-shooting problem if a larger learning-rate (such as 1e-3) is picked for this task[12].

For optimization algorithm, we have tried torch.optim.SGD[13, 14] and torch.optim.Adam[15]. Both methods gives similar results. When deploying torch.optim.SGD method, a momentum factor of 0.9 is used for perform the stochastic gradient descent. This hyper-parameter is tuned to achieve the best performance.

Experiment Different GD Algorithms

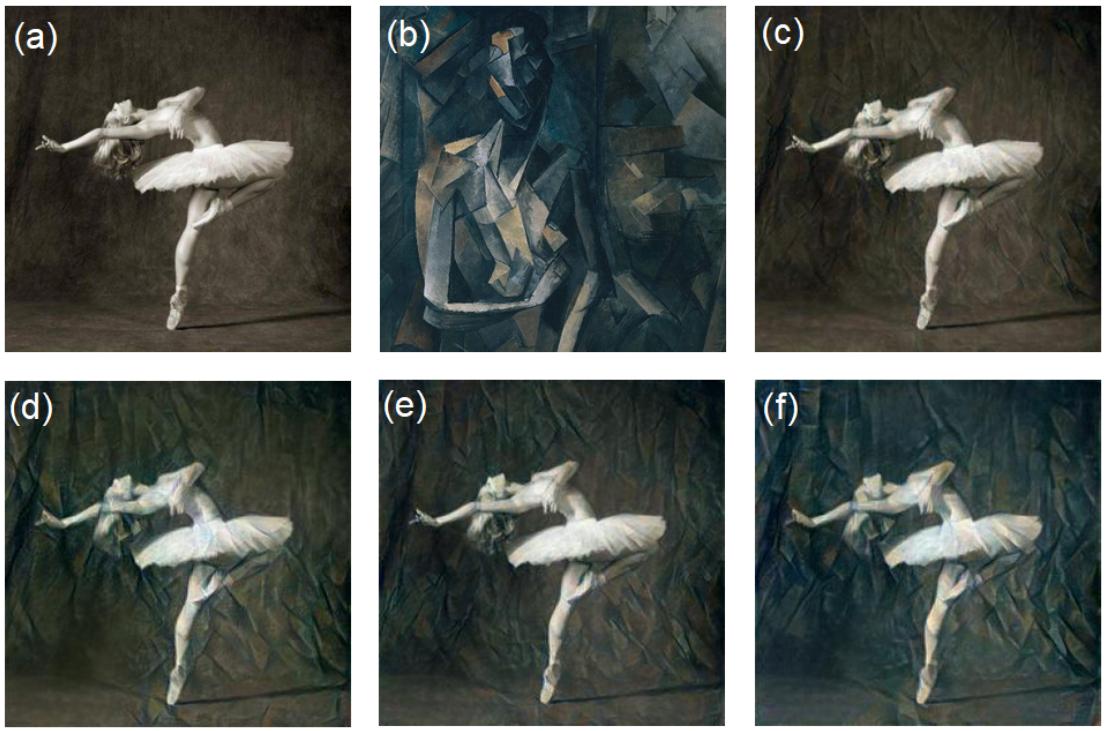

Figure 5: A figure showing the testing results for different combination of SGD hyper-parameters. Figure(a) shows the original content image (ballet dancer). Figure(b) shows the target style image(Seating Nude, Picasso). Figure(c) shows the result of SGD with Nestrov method ($\rho = 0.9$) with $\eta = 1 \times 10^{-4}$. Figure(d) shows the result of SGD with Nestrov method ($\rho = 0.9$) with $\eta = 1 \times 10^{-5}$. Figure(e) shows the result of Adam method with $\eta = 1 \times 10^{-4}$. Figure(f) shows the result of Adam method with $\eta = 1 \times 10^{-5}$.

Figure.5 shows the testing results of different gradient descent algorithms and learning rates. We tested a series of combinations of GD algorithms and learning rate values and below are examples of a few of them. The Figure.5(a) and Figure.5(b) shows the content image and style image respectively. Figure.5(c) and 5(d) are the results of torch.optim.SGD with momentum method, we pick the learning rate to be 1e-4 and 1e-5. Figure.5(e) and 5(f) are the results of torch.optim.ADAM, we pick the learning rate to be 1e-4 and 1e-5. As can be seen from the above figure, learning rate 1e-5 generates better and stable output images compared to 1e-4, probably because with large number of epochs (like 100 here), the SGD method has better chance to converge at the optimal point in energy landscape of deep neural network. Also the torch.optim.ADAM perform a little better than the traditional SGD algorithm as can be seen in above figure. But Adam method takes much longer time to be trained compared with SGD method. So it is up to personal's discretion to pick Adam method or SGD method.

Experiment Different Ratio Parameters

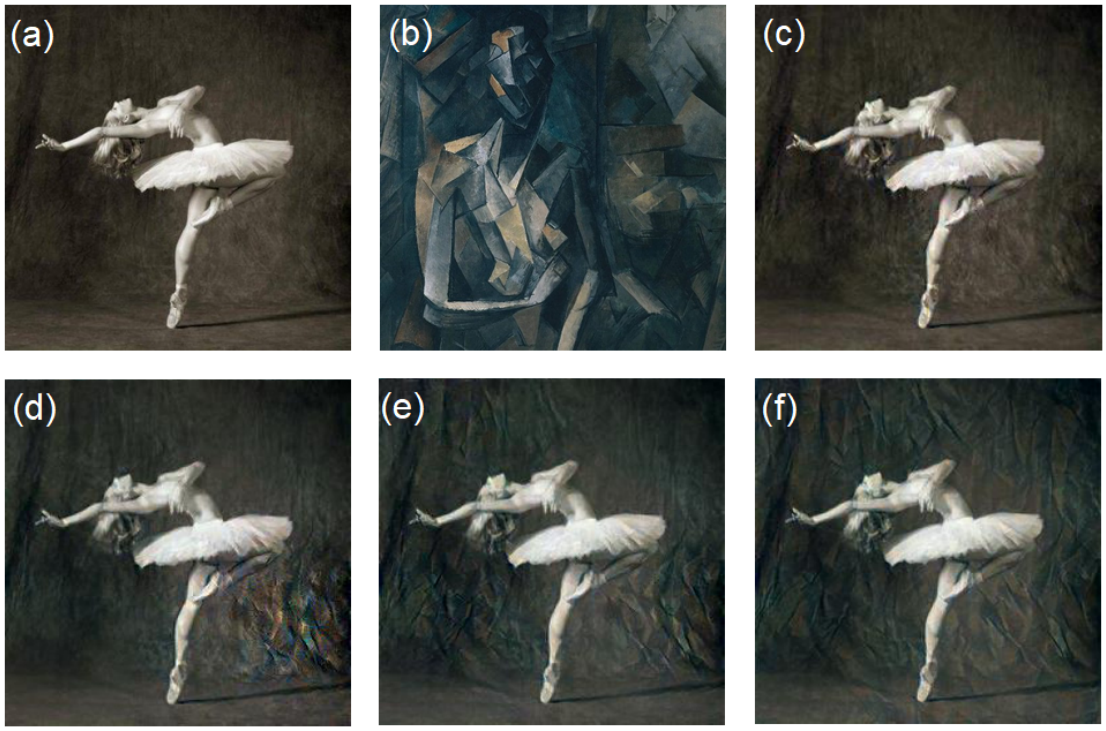

Figure 6: A figure showing the testing results for different ratio of content image to style image. Figure(a) shows the original content image (ballet dancer). Figure(b) shows the target style image(Seating Nude, Picasso). Figure(c) shows the result of content:style = 10:1. Figure(d) shows the result of content:style = 100:1. Figure(e) shows the result of content:style = 1000:1. Figure(f) shows the result of content:style = 5000:1.

Figure.6 shows the testing results of applying different content:style ratio during training phase. During training, a ratio parameter can defined to control the ratio of the content loss value to the style loss value. A larger content:style ratio means the generated image is closer to the style image, while a smaller content:style ratio means the generated image is closer to the content image. By tuning this parameter from 10 to 5000, we can see a clearly transition from Figure.6(c) to 6(f). With the ratio increasing, more and more features in style image penetrate into content image and make the final output more stylish.

It is worth mentioning that for different combinations of content and style images, a different ratio parameter need to be set to achieve optimal output. Some input images have strong styles in their own and need to use a larger ratio to overwrite the original style with new style. On the other side, some input images show neutralized style (especially for those taken in realistic photos), they can be tunned with relatively smaller ratio parameters.

Testing Different DNN Architectures

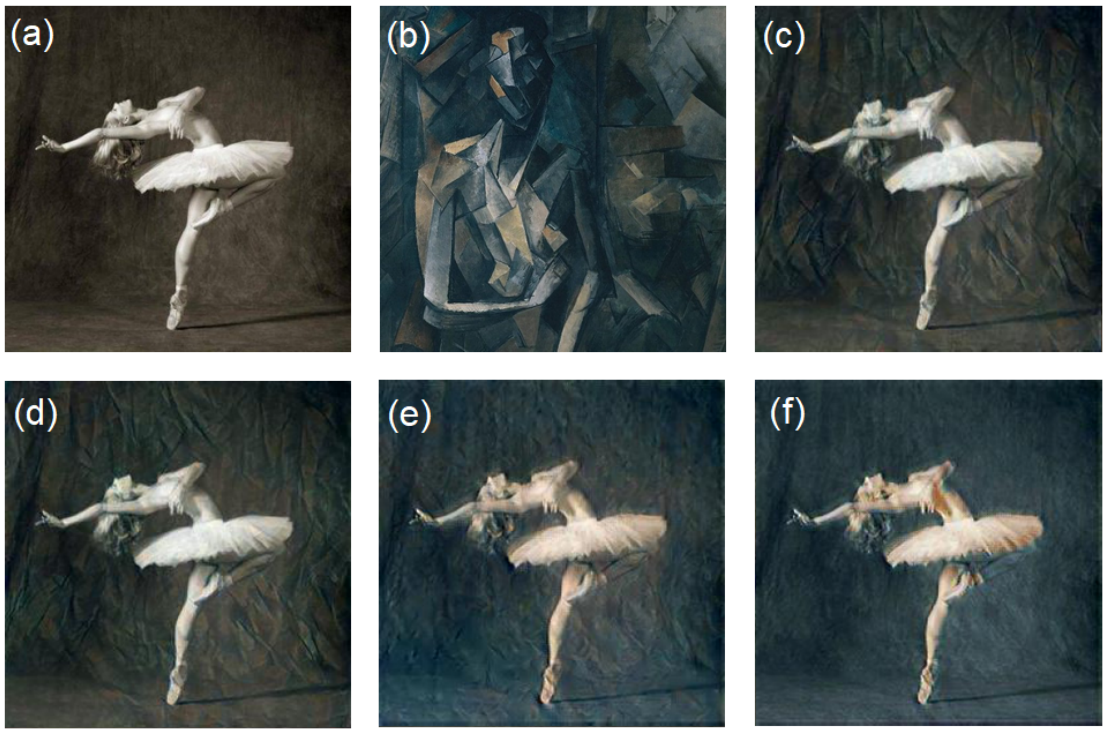

Figure 7: A figure showing the testing results for different DNN architectures. Figure(a) shows the original content image (ballet dancer). Figure(b) shows the target style image(Seating Nude, Picasso). Figure(c) shows the result of VGG16. Figure(d) shows the result of VGG19. Figure(e) shows the result of ResNet18. Figure(f) shows the result of InceptionV3.

Figure.7 shows the testing results of applying different deep neural network architectures. In this experiment, we tested ResNet18[5], VGG16[3], VGG19[3], InceptionV3[6], InceptionV4[7] and part of the results are shown in Figure.7(c)-(f). In general, the VGG19 generates more satisfying output images than other deep neural networks. Probably due to its large number of tunable parameters within the architecture.

Different architecture may be optimized with different set of parameters. It is observed that VGG16 and VGG19 may produce more satisfying images with relatively lower ratio parameters (around 1000), and InceptionV3 and InceptionV4 are more suitable for a higher ratio parameters (around 5000 10000). Also different DNN models converge with different speed, it is observed that InceptionV3 and InceptionV4 has the fastest speed during training as expected, but VGG16 and VGG19 also preserve relatively fast speed comparable to these two architectures. In our rest experiments, we mainly use VGG19 to accomplish computation as this structure produce relatively better image and consumes as fewer as computation time as InceptionV3 and InceptionV4.

Video Style Transfer

For video input style transfer, we can treat the content video as a series of frames. The different thing compare to images is that, video frames has consistency between consecutive images[16]. In order to perform artistic style transfer for videos we added a new loss function to compute the pixel-wise difference between consecutive frames[17]. The major problem for video style transfer is that it requires large amount of computation resources to accomplish[18]. A typical short video contains around 30-60 frames for each second, a 1 minute video than contains 1800-3600 frames or images to be processed[19]. Also in order to keep continuity between consecutive frames, the introduction of new loss functions will also cost additional time to compute.

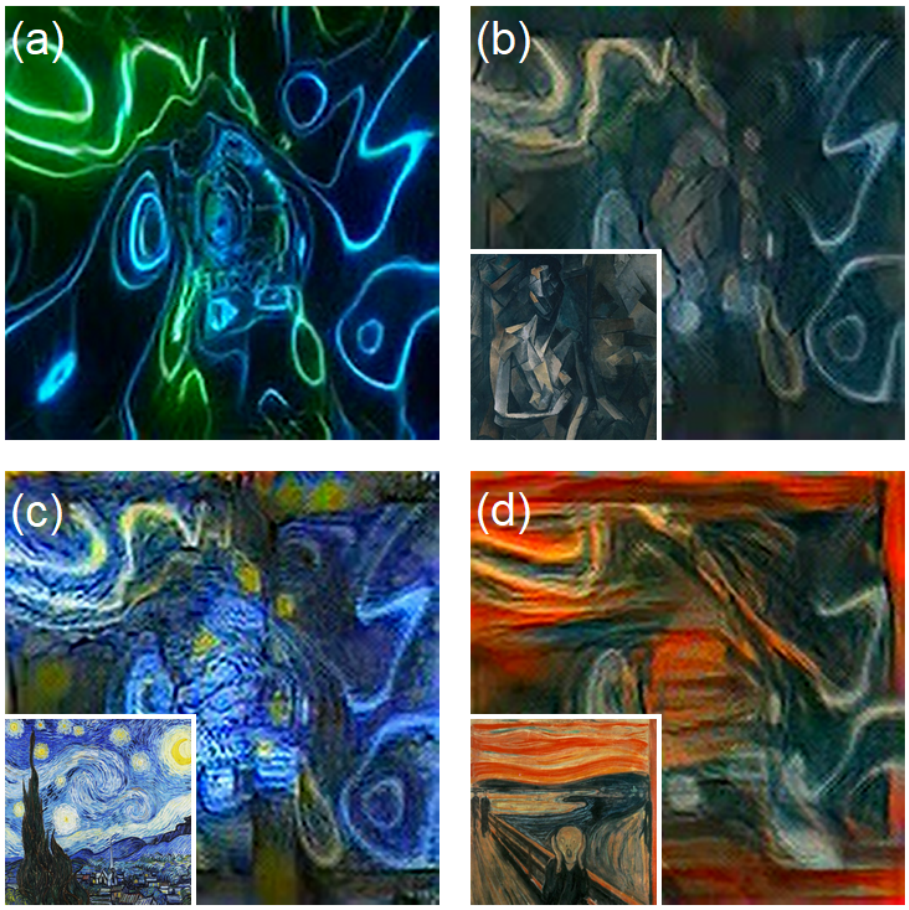

Figure 8: A figure showing screenshots of the generated videos after artistic style transfer. Plot(a): original content video (dizzy effect). Plot(b): generated video with style image as The Seated Nude by Pablo Picasso, 1910 (Style: Cubism). Plot(C): generated video with style image as The Starry Night by Vincent van Gogh, 1889 (Style: Impressionism). Plot(d): generated video with style image as The Scream by Edvard Munch, 1893 (Style: Expressionism).

One way to circumvent this dilemma is that before training, we lower the resolution of input video to 256×256 to speed up training. After training, we use pre-trained model to convert video styles in high resolution[20]. The Figure.8 shows the screenshots of the generated videos after artistic style transfer. In order to speed up training, a 4 second short video with some dizzy effect is used as the original content video, we tested with different style images as listed from Figure.8(b) to 8(d). The output video looks reasonable with both the texture structure and color tone being swapped in styled videos. These video can be watched in this Link.

Conclusion

In this work, we researched the deep neural network architectures for artistic style transfer. A loss function with multiple regularization terms is proposed to construct the neural network. Pre-trained model with different architectures are used during our network training phase.

In order to better understaing and optimize the deep neural network architecture, we experimented and tested the effect caused be picking different GD algorithms, ratio parameters and DNN Architectures to final output images. Overall, the results generated through Adam method with VGG19 architecture and ratio parameter set between 100-1000 looks reasonable and cost affordable computation resources within this project.

For video style transfer, we developed a process to style transfer small videos with low resolution. A new regularization term is proposed to keep the continuity of consecutive frames within the video. The output video shows reasonable result.

Github Repository
The above work is uploaded to following Github repository:
Deep-Neural-Networks-for-Artistic-Style-Transfer

References

[1] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural style transfer: A review. *IEEE Transactions on Visualization and Computer Graphics*, 26(11):3365–3385, 2020.

[2] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 27(30):2414–2423, 2016.

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Computer Vision and Pattern Recognition (cs.CV)*, 14(9):1556, 2014.

[4] Anat Levin, Dani Lischinski, and Yair Weiss. A closed-form solution to natural image matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):228–242, 2008.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Computer Vision and Pattern Recognition (cs.CV)*, 15(12):385, 2015.

[6] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *Computer Vision and Pattern Recognition (cs.CV)*, 15(12):567, 2015.

[7] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *Computer Vision and Pattern Recognition (cs.CV)*, 16(2):261, 2016.

[8] Wikipedia contributors. Gram matrix. Accessed: 2004-07-22.

[9] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *Computer Vision and Pattern Recognition*, 1603(8):155, 2016.

[10] Haozhi Huang, Hao Wang, Wenhan Luo, Lin Ma, Wenhao Jiang, Xiaolong Zhu, Zhifeng Li, and Wei Liu. Real-time neural style transfer for videos. *C2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 11(09):7044–7052, 2017.

[11] Monika Bansal, Munish Kumar, Monika Sachdeva, and Ajay Mittal. Transfer learning for image classification using vgg19: Caltech-101 image data set. *Journal of Ambient Intelligence and Humanized Computing*, 14(4):3609–3620, 2023.

[12] Google AI contributors. The stochastic gradient method. Accessed: 2018-11.

[13] Aleksandar Botev, Guy Lever, and David Barber. Nesterov's accelerated gradient and momentum as approximations to regularised update descent. *Machine Learning (stat.ML); Machine Learning (cs.LG)*, 16(19):81, 2016.

[14] Farshed Abdukhakimov, Chulu Xiang, Dmitry Kamzolov, and Martin Takáč. Stochastic gradient descent with preconditioned polyak step-size. *Machine Learning (cs.LG); Optimization and Control (math.OC)*, 10(20):93, 2023.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Machine Learning (cs.LG)*, 12(69):80, 2015.

[16] N. Ashikhmin. Fast texture transfer. *IEEE Computer Graphics and Applications*, 11(9):38–43, 2003.

[17] H. Lee, S. Seo, S. Ryoo, and Yoon. K. Directional texture transfer. *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, 10(10):43–48, 2010.

[18] M. Cimpoi, S. Maji, and A. Vedaldi. Deep filter banks for texture recognition and segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 33(2):3828–3836, 2015.

[19] D. L. K. Yamins, H. Hong, C. F. Cadieu, E. A. Solomon, D. Seibert, and DiCarlo. J. J. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, 40(3):112, 2014.

[20] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis synthesis. *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, 95(10):229–238, 1995.