Fish Touchers Real-Time Style Transfer Project

Team Fish Touchers: Song Wang, Jingqi Huang, Yifan Huang, Changhan Ge
PID: A53275641, A53281935, A53272363, A53264280
Github: https://github.com/ChanghanGe/ECE285sp19-FishTouchers-Project
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, CA 92093
{sowang,jih023,y5huang,chge}@eng.ucsd.edu

Abstract

This paper is the final project report of the ECE 285 Machine Learning for Image Processing in Spring 2019 at University of California San Diego. We choose the Project B - Style Transfer as our research topic. Style transfer has been a research hot spot in computer vision area for a long time. With the development of machine learning, several convolutional neural network (CNN)-based style transfer approaches were proposed. In this project, we realized real-time style transfer by building and training a feed-forward image transformation network which solves the neural optimization problem proposed by Gatys et al..

1 Introduction

Neural style transfer for images is a machine learning technique to project the artistic style of image one onto the content of image two. In this way, the generated image has the content of the image two, but with the style of image one. This is hard to solve in the conventional computer vision; however, this problem has been made possible and easier with the introduction of machine learning.

This topic was first introduced by Gatys el. al in the A Neural Algorithm of Artistic Style [1]. According to this paper, first, we use Convolutional Neural Networks that are trained on object recognition to develop to extract the content of the image. This structure of network makes the content information more and more obvious along the processing hierarchy. Now we can reconstruct an image from the features from each layer. This means the higher layers can capture higher-level of content, which represents the object and the arrangement of the input image, instead of capturing the exact pixel values of the image like in the lower layers. Then, a feature space originally designed to capture texture information is used to extract the style of the input image. The feature space contains correlations between different filter responses in each layer. Then, combining the feature correlations of many layers, we obtain a representation of the texture information of the input image. Later, after doing content/feature reconstruction [7] and style reconstruction, which have a similar method used for texture synthesis [5], we can manipulate with the representations to produce a mixture of the content from one picture and the content from the other.

Gatys's paper described a method that needs input of both content image and the style image every time while running the task. While the result for the style transfer of this method is great, the speed for processing each image is low since it requires to solve an optimization problem while processing each content and style. This is not ideal for real time style transfer.

Since then, much modifications have been made to improve the efficiency style transfer. Such as [15] by Google and [16] by Luan et al explore the possibilities to do fast style transfer using a combination of various networks or loss functions. One of the most mentioned work is Perceptual losses for real-time style transfer and super-resolution by Justin Johnson et al [12]. This method can

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

train a network for a specific style by adding an image transformation network in front of the loss network. The transformation network is a feed-forward CNN that roughly follows the procedures created in [10], but with some modifications. This image transformation network consists a couple residual blocks using the structure of [11]and some non-residual convolutional layers with spatial batch normalization [2] and ReLU nonlinearities. This network first down-sample then up-sample with convolutional layers [3, 4], which is more computational effective and increases the effective receptive field sizes. The loss network uses perceptual loss rather than using per-pixel loss. The per-pixel loss does not capture the perceptual differences between output and the ground-true images. As the output image shift by one pixel, the per-pixel loss would increase tremendously. However, perceptual loss is a combination of feature reconstruction loss and style reconstruction loss in the loss network and perform gradient descent to change the weight of the image transform network, so that it can represent the perceptual and semantic difference between images. Later, we added total variation regularization loss to our algorithm. It gives penalty to local variation of the output image. This loss is used to increase smoothness and suppress noise of the output image. After the network is trained, the loss network is not needed for the style transfer. In this way, all input image can be style transferred with a much faster speed for a specific style and this method is suitable for real time processing such as video style transfer and gaming applications.

2 Neural Style Transfer

2.1 Loss Functions

In style transfer, several different loss functions are defined to evaluate the difference in content and style between input image, content image and style image.

2.1.1 Content Loss

The content loss function quantitatively describes the difference between the input image x and the content image p. Suppose $F^l_{i,j}(x)$ and $P^l_{i,j}(p)$ are the i,j-th feature representation of the input image x and the content image p at the l-th layer in the neural network, the loss function is defined as the Euclidean Distance between $F^l_{i,j}(x)$ and $P^l_{i,j}(p)$, given by

$$\mathcal{L}_{\text{content}}(\mathbf{p}, \mathbf{x}, l) = \frac{1}{2} \sum_{i,j} (F^l_{i,j}(x) - P^l_{i,j}(p))^2 \tag{1}$$

2.1.2 Style Loss

While the content loss penalizes the input on its deviation from the content feature, the style loss function is also defined to measure the difference in style between the input image and style image. Here, the Gram Matrix of the i,j-th feature response of the l-th layer represents the style as Eq.2.

$$G^l_{i,j} = \sum_k F^l_{i,k} F^l_{j,k} \tag{2}$$

Basically, the Gram Matrix is the inner product between the vectorized feature map i and j at l-th layer. Denote $G^l_{i,j}(x)$ and $A^l_{i,j}(a)$ as the Gram Matrices of the input image x and style image a at the l-th layer, the contribution of this layer to the total style loss is given by the mean-squred Euclidean distance between them, expressed as

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G^l_{i,j}(x) - A^l_{i,j}(a))^2 \tag{3}$$

where N_l is the number of feature maps, each of which has the size of $M_l = H \times W$ (height by width). By adding up the style loss of all the layers, we can obtain the total style. Usually, we weight the contribution of each layer with some factor w_l. In this project, we equally weight each layer as $w_l = \frac{1}{L}$. Hence the total style loss can be expressed as

$$\mathcal{L}_{\text{style}}(\mathbf{a}, \mathbf{x}) = \sum_{l=0}^{L} \frac{1}{L} E_l \tag{4}$$

At this stage, we combinedly consider the effect from both content loss and style loss by defining the total loss function as follow

$$\mathcal{L}_{\text{total}}(\mathbf{p}, \mathbf{a}, \mathbf{x}) = \alpha \mathcal{L}_{\text{content}} + \beta \mathcal{L}_{\text{style}} \tag{5}$$

where $0 < \alpha, \beta < 1$ is the proportion of content and style, respectively. In this paper, we define and change $\eta = \frac{\alpha}{\beta}$ to evaluate the effect of the ratio. Detailed experiment will be discussed in Section 4.

During the training procedure, we perform backpropagation on total loss as usual while the goal of the neural network is to minimize the total loss in order to make the output image have both similar feature responses on content and style at l-th layer.

3 Real-Time Style Transfer

Our machine learning model composes two different neural network-a residual network serving as image transformation network f_W and a pretrained VGG16 serving as loss network ϕ. The structure of our style transfer model is shown as Fig.1. The details of the image transformation network and loss network are be discussed in section 3.1 and 3.2.

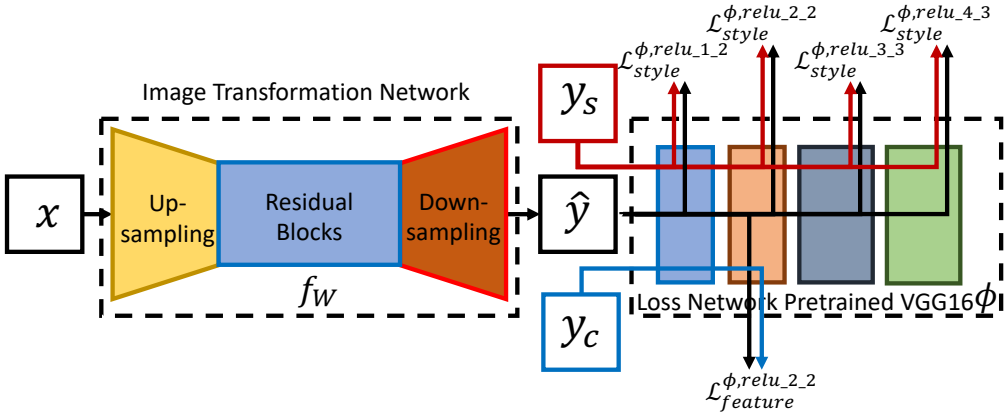


Figure 1: Structure of Style Transfer Model

3.1 Image Transformation Network

We follow the pipline in [10] and architecture in [11], implemented a deep residual convolutional neural network with five residual blocks as our image transformation network which transforms the input images $\mathbf{x}$ to output images $\mathbf{y}$ via $\mathbf{y} = f_W(\mathbf{x})$, where W stands for the weights of the network.

The input image is firstly forwarded to several upsampling layers. After the upsampling, the image will be passed through 5 residual blocks. For each residual block, the input is forwarded to two different branches - a shallow convolutional network and a shortcut linking the input to output, where input and the output of the shallow network is added up. This structure is designed by Microsoft Research Asia to solve the optimization difficulty of deep feed-forward convolutional neural network. At the end, the residual blocks are followed by the the same number of downsampleing layers as upsampling layers.

Based on the same method in [12], the $\log 2f$ convolutional layers with stride $\frac{1}{2}$ are used for sampling. There are two major benefits for doing upsampling and downsampling with convolutional layers. The first one is related to computation. A 3×3 convolution with DC filters on an input with the shape of $C/D \times H/D \times W/D$ has the same times of multiply-adds ($9C^2HW$) as a 3×3 convolution with C filters on an input of the shape $C \times H \times W$. Therefore, the downsampling enables us to implement a large network with same computational cost. Another benefit is of the effective receptive field sizes. The receptive field size at the input is crucial to high-quality style transfer since it changes a large part of the image. Without downsampling, each 3×3 convolutional layer increases the receptive field by 2. In contrast, each 3×3 convolutional layer increases the receptive field by 2D if the input

image is passed through a downsampling network with factor D. Hence, by downsampling, we will have a larger receptive field.

In our model, except for the output layer which is followed by tanh activation to ensure the pixel range consistency, other non-residual convolutional layers are followed by batch normalization and Relu activation to ensure the linearity. Also, all the convolutional layers use 3×3 kernels other than the first and last layers use the 9×9 kernels.

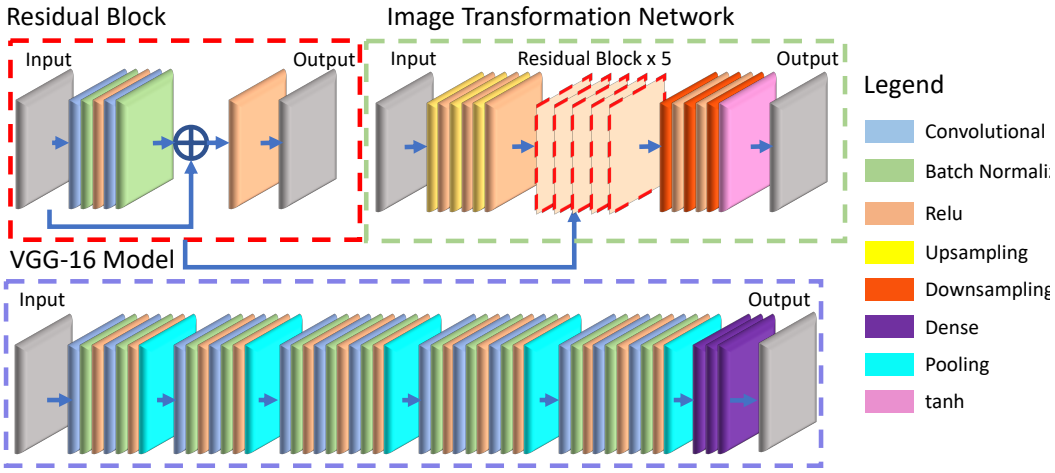


Figure 2: Structure of Image Transformation Network and Loss Network

The input and output of the image transformation network have the same dimension of $3 \times 256 \times 256$, where 3 and 256 is the number of color channels (RGB) and width (length) of the picture, respectively. At the test stage, image can be of any resolution since the image transformation networks are fully connected. The structure of the image transformation network and residual block is shown in Fig.2

3.2 Loss Network

VGG networks [6] were originally proposed by K. Simonyan and A. Zisserman from Robotics Group at University of Oxford to solve the ImageNet [8] Classification Challenge [9]. It has two different models-VGG16 and VGG19, which has 16 layers and 19 layers respectively. It achieves around 92.7% top-5 classification accuracy on ImageNet dataset. Comparing with the previous ImageNet challenge champion-AlexNet, the performance improvement of VGG was achieved by replacing the large kernel-sized filters with multiple 3 kernel-sized filters one and after another. In this project, since the VGG19 doesn't shows a significantly better performance than VGG16, for the sack of simplification of the model, we decided to adopt the feature layers (structure is shown in Fig.2) in pretrained VGG16 to serve as loss network which measures the perceptual differences in content and style between two images.

Specifically, although the implementation of pooling layer expanded the receptive field of the model, we drop out all the MaxPooling layers in our final version in order to keep more the information. The style loss is evaluated at the 2nd relu layers of the first two convolutional blocks and the the 3rd Relu layers of the last two convolutional blocks, while the feature loss is evaluated at the second Relu layer of the second convolutional block.

3.3 Dataset

In this project, we trained and tested our networks on the COCO-2015, a large scale object detection, segmentation, and captioning dataset composing 330K images, shown as Fig. 3(a). During the training process, we cropped the images in to 256×256 to assure the dimension consistency. The style images are from Internet including Marilyn Monroe, Starry Night and The Scream, shown in Fig.3(b), 3(c), 3(d).

(a) Sample in COCO-2015 (b) Style: Marilyn Monroe (c) Style: Starry Night (d) Style: The Scream

Figure 3: Training Sample and Styles

3.4 Loss Functions

3.4.1 Feature Reconstruction Loss

Since our goal is realizing the style transfer rather than exactly transforming the input image to the target image, we hope the output image can have a similar feature representation of the target. Hence, the feature reconstruction loss function is defined to evaluate the feature representation difference, expressed as

$$\mathcal{L}_{\text{feat}}^{\phi,j}(\hat{y}, y) = \frac{1}{C_j H_j W_j} ||\phi_j(\hat{y}) - \phi_j(y)||_2^2 \tag{6}$$

where ϕ_j is the activations of the j-th layer of the loss network, $C_j \times H_j \times W_j$ is the shape of the feature map $\phi_j(y)$ when the j-th layer is an convolutional layer. Essentially, this is the sqaured Euclidean distance between the j-th feature map of the output image $\hat{y}$ and target image y.

3.4.2 Style Reconstruction loss

Similar to feature reconstruction loss, Gatys et al.evaluate the image styles, including color, pattern and texture, by calculating the Gram Matrix G_j^{ϕ} of the activation of the j-th convolutional layer, given by follow

$$G_j^{\phi}(x)_{c,c'} = \frac{1}{C_j H_j W_j} \phi_j(x)_{h,w,c} \phi_j(x)_{h,w,c'} \tag{7}$$

Then, the style reconstruction loss is defined as the squared Frobenius norm of the differences between style of the output and targe images, shown as

$$\mathcal{L}_{\text{style}}^{\hat{y},y} = ||G_j^{\phi}(\hat{y}) - G_j^{\phi}(y)||_F^2 \tag{8}$$

3.4.3 Pixel loss

Following [7], the euclidean distance between the target y image and the output image $\hat{y}$ is defined as the pixel loss. This loss function only works when we already have the oracle target image y which the network is expected to match. Suppose the y and $\hat{y}$ have the same shape of $C \times H \times W$, the pixel loss function can be expressed as Eq.9.

$$\mathcal{L}_{\text{pixel}}(\hat{y}, y) = \frac{||\hat{y} - y||_2^2}{CHW} \tag{9}$$

3.4.4 Total Variation Regularization

In order to achieve a higher spatial smoothness of the output image $\hat{y}$, we adopt the Total Variation (TV) Regularization in [7]. For a continuous function $f : \mathbb{R}^H \subset \Omega \to \mathbb{R}$, the TV regularization can expressed as:

$$\mathcal{L}_{\text{TV}}(f) = \int_{\Omega} ((\frac{\partial f}{\partial u}(u,v))^2 + (\frac{\partial f}{\partial v}(u,v))^2)^{\frac{\beta}{2}} du dv \tag{10}$$

where β is empirically set to 1. Since the output image $\hat{y}$ is discrete, we can rewrite the TV regularization with finite-difference approximation as

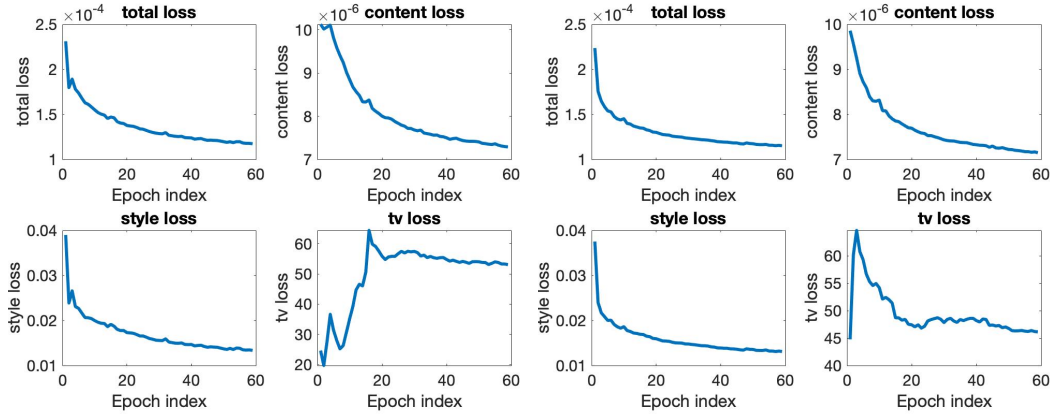
$$\mathcal{L}_{\text{TV}}(\hat{y}) = \sum_{i,j} \sqrt{(\hat{y}_{i,j+1} - \hat{y}_{i,j})^2 + (\hat{y}_{i+1,j} - \hat{y}_{i,j})^2} \tag{11}$$

4 Experiments & Results

For every following experiment, we built our network with Pytorch and trained it with 1 NVIDIA 1080 Ti GPU for 60 epochs on the server provided by UC San Diego data science center. We analyzed the loss changing trend, compared the effect of the content-loss-to-style loss ratio, the effect of TV loss, and the comparison between FeiFei Li's Approach and Gatys' approach. We also enabled the model to do real-time style transfer and test the function on a short video.

4.1 Loss Changing Trends

Firstly, we evaluated the trends of loss changing. As shown in Fig.4(a) and 4(b), the content loss and style loss are decreasing with the increasing of epoch. Although some oscillations appear during the training process, it doesn't affect the overall performance shown in total loss. This indicates that the content and style of transferred image becomes less and less different from the content image and style image, respectively. The only anomaly here is the TV loss which shows a trend of increasing first and then decreasing, and however is supposed to have the same trend as others. We considered this effect as a result of non-ideal η and the ratio between TV loss and style loss. In order to prove this, we reevalute the trend by setting the proportion of content loss, style loss, and TV loss as $1:1:1000$, shown as Fig.5. We clearly see that the tv loss decreases with the increasing of epoch, which validates our analysis. Since the trends of all the kinds of loss are correct, our model works well and we can now conduct the following experiments.



(a) $\eta_{\text{tv}} = 10^{-14}$ (b) $\eta_{\text{tv}} = 10^{-9}$

Figure 4: Loss Trends

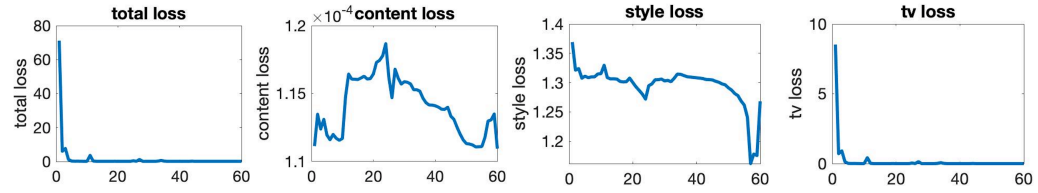


Figure 5: Trends of TV loss

4.2 Analysis on the η

Secondly, we analyzed the effect of changing η. As shown in Fig.6, we tested the neural network with $\eta = 1000, 3000, 6000$ and 13000, respectively. The content image is a man catching a fish, and the style image is a popular painting "The Scream". We can clearly see the trend that, with the increasing of η, the transformed images captured more features from the content image. Inversely, the features from style image contributed more in transformed images with the decreasing of η. Apparently, the ratio between content loss and style loss is a key factor in style transfer.

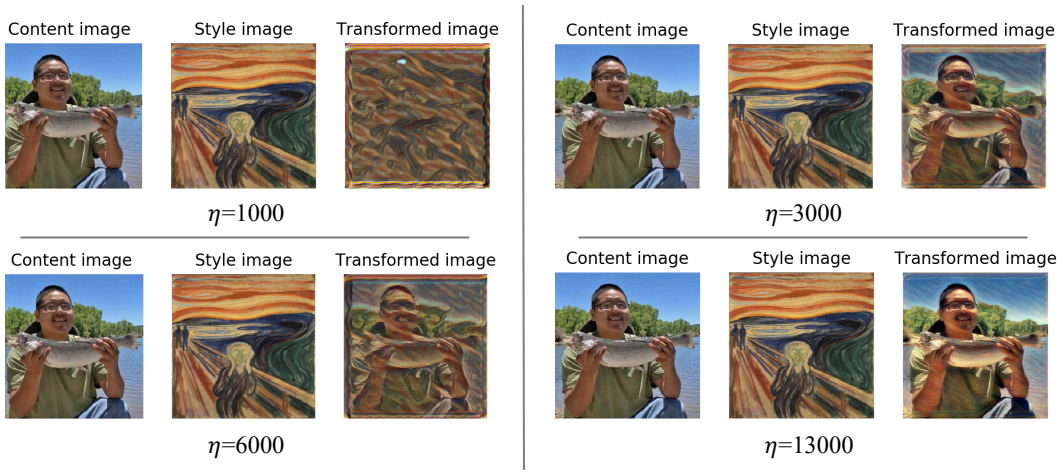


Figure 6: Comparison Between Different η

4.3 Analysis on the TV loss

Another parameter we evaluated is the TV loss. We fixed the $\eta = 3 \times 10^3$, and change the ratio between TV loss and style loss η_{TV} from 10^{-14} to 10^{-5}. In Fig.7, the rows are transformed image with different η_{TV} respectively. Decreasing the proportion of TV loss, we find that the transformed image becomes rougher and rougher. Hence, the higher the η_{tv}, the smoother the transformed picture and vice versa. It also shows that the TV loss is basically a kind of normalization method.

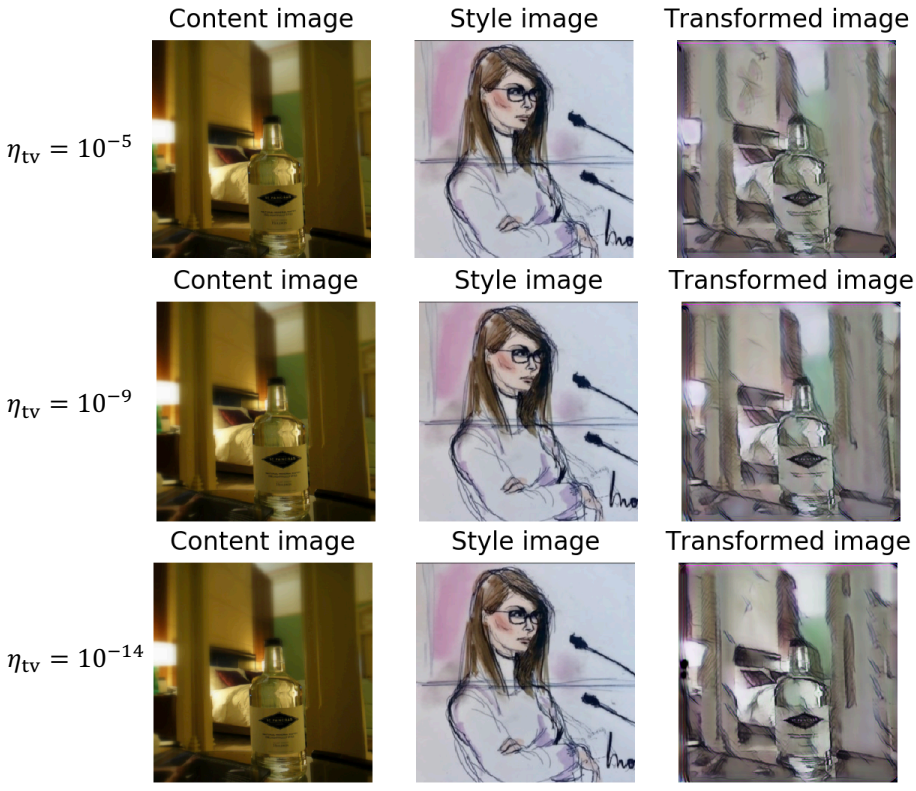


Figure 7: Comparison Between Different TV loss

4.4 Comparison Between Li's approach and Gatys' approach

Feifei Li's approach and Gatys' approach are very different. We evaluated the quality of the transferred picture produced by these two methods. As shown in Fig.8, the four columns from left to right are content images, style images, transferred images based on Li's approach, transferred images based on Gatys' approach. We can clearly see that quality of images produced by Gatys' approach is obviously better (smoother, showing less artifact) than those produces by Li's approach. This result is theoretically reasonable since Gatys' approach train model on each image individually. But undeniably, Li's approach achieves a faster training speed and is more applicable in real-time style transfer, especially when the demand of quality is not very strict.

Figure 8: Comparison Between Li's approach and Gatys' approach

4.5 Enabling Real-Time Style Transfer

For realizing the real-time style transfer, we chose a short video filmed in 2000 by Hong Kong TVB, which is an interview of contemporary President of People's Republic of China - Mr. Zemin Jiang, and set The Scream as style. We successfully ran the real time style transfer with a rate of 30 frames per second. The real-time style transferred video is already uploaded to github repository. A screenshot is shown in Fig.9.

Figure 9: Screenshot of Real-time Style Transferred Video

5 Conclusion

In this project, we firstly implemented the neural style transfer proposed by Gatys et al, and then realized the real-time style transfer based on Feifei Li's approach. We conducted several experiments to evaluate the loss changing trends, the effect of changing the loss ratio, and compare the pros and cons of Li's appraoch and Gatys' approach. At the end, we successfully tested the real-time style transfer with a short video.

Acknowledgments

We want to give special thanks to Prof. Charles Deledalle and teaching assistant Mr. Inderjot Singh. They provided the fruitful literature where our project is based on, and gave us great support and guidance from the beginning to the end.

References

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

[2] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).

[3] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[4] Noh, Hyeonwoo, Seunghoon Hong, and Bohyung Han. "Learning deconvolution network for semantic segmentation." Proceedings of the IEEE international conference on computer vision. 2015.

[5] Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "Image style transfer using convolutional neural networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[6] Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[7] Mahendran, Aravindh, and Andrea Vedaldi. "Understanding deep image representations by inverting them." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[8] Deng, Jia, et al. "Imagenet: A large-scale hierarchical image database." 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009.

[9] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[10] Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).

[11] Gross, S., Wilber, M.: Training and investigating residual nets. http://torch.ch/blog/2016/02/04/resnets.html (2016)

[12] Johnson, Justin, Alexandre Alahi, and Li Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution." European conference on computer vision. Springer, Cham, 2016.

[13] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer–Verlag.

[14] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.

[15] Ghiasi, Golnaz, et al. "Exploring the structure of a real-time, arbitrary neural artistic stylization network." arXiv preprint arXiv:1705.06830 (2017).

[16] Luan, Fujun, et al. "Deep photo style transfer." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.