Oil Painting Style Transfer for Images Using Both Traditional and Deep Learning Methods

Chuan Sun
University of Kansas
Lawrence, KS
chuansun@ku.edu

***Abstract*—In this paper, the problem of transferring image style to that of oil paintings is investigated. We discussed and applied two approaches, which are the traditional approach and deep learning approach, on representative image categories. Output images are compared and analyzed with the effect of various changing parameters. Traditional approach generates ideal and synthetic oil styles, while the deep learning approach gives more authentic style textures which are discernible in frequency domain.**

Keywords—Style transfer, Oil Painting, Edge Detection, Deep Learning, Gradient Descent

I. INTRODUCTION

With the rapid development of image processing and computer vision, the analysis on image styles has been making huge progress. Image styles are characterized by the artists' painting custom and genre, which in practice corresponds to a bunch of painting styles such as watercolor, sketch, oil painting, sand painting and so on. Transferring the style of an image to another is one of the main focus and practical application amidst this domain, which aims to take an arbitrary image as input, and generate an output with the same content but in the desired painting style.

A number of recent works have been conducted on analyzing the style of an image and trying to convert it to another. Traditional ways tend to utilize image processing techniques including image sharpening, edge detection, segmentation, reconstruction, etc. [1], to compute style features mathematically and apply the extracted features onto new images.

The work by Alejo Hausner [2] proposed a method to simulate image tile mosaics using Voronoi diagram and distance measurement, which generates a mosaic-like filter style on images while preserving its contents. Fan Hui, Chen Zheng and Li Jinjiang designed a sand style transfer algorithm [1] using fuzzy enhancement and texture synthesis. These methodologies make full use of mathematical tools such as gradient, differential, probabilistic modeling, Manhattan distance, etc. to simulate the desired style patterns to be applied.

On the other hand, the flourishing development of neural network and deep learning provides a different approach for style comprehension and transformation. Lots of research work have been done on representing image content and style using deep learning techniques such as CNNs and VGG networks. For instance, the work by Leon A. Gatys separates the representation of image content from its style through convolutional neural networks [3]. Justin Johnson replaced the traditional pixel-level loss with perceptual loss defined on high level features [4] for the neural network doing image transformation.

Deep learning methodologies enable better generalization on feature extraction and representation through a stack of neural layers such as normalization, convolution, pooling, or even long short term memory layers for preserving long sequence information. While at the same time, they require ground truth target tags to do supervised training in most cases.

In this paper, we explore the methodologies of transferring the style of an image to that of an oil painting. To achieve a good blend between oil painting style and image content, the oil brush patterns are analyzed. Then both traditional method and deep learning networks are applied to simulate the oil painting style and compared to see their pros and cons.

II. METHOD

Oil painting styles contain a characteristic visual appearance pattern originated from the oil brush, which differentiates itself from other painting styles. We start from constructing a transformation to simulate this characteristic pattern using traditional image processing techniques.

A. Traditional Approach

Since the idea comes from producing oil painting styles by simulating every physical stroke of an oil painting, we first need to determine the locations, magnitudes and directions of all potential brush strokes to be put on the canvas. Based on the knowledge learned during the course, edge detection algorithms provide a superb solution for all these three requirements. Gradients of the image are calculated based on formula 1.

$$\nabla f(x,y) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{1}$$

The gradients on both horizontal and vertical axis determine the magnitude and the size of the stroke, denoted as

$$g(x,y) = \sqrt{{g_x}^2 + {g_y}^2} \tag{2}$$

In our method, the pattern of every stroke is approximated by an eclipse, whose length and size are determined by the magnitude of every gradient. The direction of the eclipse is determined by

$$\theta(x,y) = \arctan\left(\left[\frac{g_y}{g_x}\right]\right) \tag{3}$$

We evenly distribute every stroke across the entire width and height of the canvas on a predefined brush width, and add some random noise to the stroke order so that every stroke looks different and randomized in both size and location.

To approach more resemblance of manually crafted paintings, the original image is first slightly blurred using a

medium filter to smooth out sharpness from real world images. Then the stroke orders parameterized by brush size are generated with random noise, determining the oil brush stroke locations. Finally different eclipses are constructed on these locations based on edge magnitude and orientation.

B. Deep Learning Approach

The traditional method turns out to work quite well on transferring image into an oil painting style, while at the same time having limitations as well, especially in terms of style generalization. In this project, the deep learning methodology is further added to explore the effect of neural network layers on style representation and loss optimization during the style transfer process.

This new approach proceeds our work from unsupervised computation into supervised learning domain, where a desired style example will be required as input, and the model is trained to generates an output image in the style of this input.

Convolution Neural Networks are proficient at understanding image data through layers of feature extraction and pooling summarization. When applying such a network onto our research, we need to make use of this advantage on our problems, and make convolutional layers extract the representation of both image contents and image styles. With the model stacking more and more convolutional layers, it tends to extract higher and higher level of information, from image surface patterns to edges and segments, finally to abstract objects. In order to separate image style from content, we consider the lower layers as representation of styles, and higher layers as representation of contents.

(1) Model Architecture

The structure and weights of the VGG-19 model is utilized here to stack a bunch of convolutional layers together, which consists of 5 model blocks, with each block containing different number of convolutional layers and a max pooling layer, depicted in Fig 1.

Layer (type)	Output Shape	Param #
input_4 (InputLayer)	(None, None, None, 3)	0
block1_conv1 (Conv2D)	(None, None, None, 64)	1792
block1_conv2 (Conv2D)	(None, None, None, 64)	36928
block1_pool (MaxPooling2D)	(None, None, None, 64)	0
block2_conv1 (Conv2D)	(None, None, None, 128)	73856
block2_conv2 (Conv2D)	(None, None, None, 128)	147584
block2_pool (MaxPooling2D)	(None, None, None, 128)	0
block3_conv1 (Conv2D)	(None, None, None, 256)	295168
block3_conv2 (Conv2D)	(None, None, None, 256)	590080
block3_conv3 (Conv2D)	(None, None, None, 256)	590080
block3_conv4 (Conv2D)	(None, None, None, 256)	590080
block3_pool (MaxPooling2D)	(None, None, None, 256)	0
block4_conv1 (Conv2D)	(None, None, None, 512)	1180160
block4_conv2 (Conv2D)	(None, None, None, 512)	2359808
block4_conv3 (Conv2D)	(None, None, None, 512)	2359808
block4_conv4 (Conv2D)	(None, None, None, 512)	2359808
block4_pool (MaxPooling2D)	(None, None, None, 512)	0
block5_conv1 (Conv2D)	(None, None, None, 512)	2359808
block5_conv2 (Conv2D)	(None, None, None, 512)	2359808
block5_conv3 (Conv2D)	(None, None, None, 512)	2359808
block5_conv4 (Conv2D)	(None, None, None, 512)	2359808
block5_pool (MaxPooling2D)	(None, None, None, 512)	0

Fig. 1. Architecture of VGG-19.

Each convolution layer applies a number of feature filters to extract different features from the input image, and the model summarizes the extracted feature using max pooling layers in a hierarchical manner. The pretrained ImageNet weights enable strong capability in object recognition and feature extraction. Among this architecture, the first convolution layer in every block is extracted as our style layers, and the last convolution layer serves as our content layer. Style layers are stacked with the content layers to form our deep model.

(2) Loss Function

Calculating loss has been crucial for the model evolving towards its best solution. For this project, the model should keep a decent balance between style transformation and content preservation, so the loss should take both content loss and style loss into consideration.

When the original content image x and the model final output y are fed into the network, their model output for filter i at layer l are denoted as ${C_i}^l(x)$ and ${C_i}^l(y)$. The content loss is retrieved from the difference between model output and the target at each layer, which is denoted by

$$ {L_{content}}^l(x,y) = \sum_i({C_i}^l(x) - {C_i}^l(y))^2 \qquad (4) $$

This content loss ensures that the model output does not end up far away from the content input. In order to transfer the input image toward the desired style input, the style loss is defined in a similar way, with the only difference that style representation is not calculated from model output, instead from the inner correlation between all filters inside a layer, formulated by:

$$ {L_{style}}^l = \frac{1}{4N^2M^2}\sum_{i,j}({G_x}^l - {G_y}^l)^2 \qquad (5) $$

where i, j represents any two filters inside a layer, N is the number of filters inside layer l, M is the size of the image(width * height), and ${G_x}^l, {G_y}^l$ denotes the inner product between two filter map in layer l fed with the input x and y respectively.

The total style loss is the sum of all layer losses. To summarize, the style loss does not focus on the feature outcome generated by layers, but on the style resemblance between each two filter maps. Larger inner correlation between two filters indicates higher level of style resemblance, thus ensuring that the model output contains a similar style to the input style image.

(3) Gradient Descent

The model is trained for a predefined number of iterations. For each iteration, the model computes total loss and the optimal gradient towards minimizing the total loss using Adadelta optimizer. Calculated gradients are then applied onto the input content image for gradual evolving. The best model with lowest total loss is preserved as the final output.

The model outputs are compared with those generated by the traditional approach, with additional analysis in frequency domain. They will be discussed in the next chapter.

III. EXPERIMENT

For the experiment of this project, we conducted both traditional and deep learning methods for oil painting transfer. Five images that are representative in 5 domains (landscape, architecture, natural wonder, human being, object portrait) are

selected as content input image. For deep learning methodology, several different authentic oil paintings are selected as style input.

A. Traditional Approach

Fig 2 shows two samples of original content images and the corresponding outputs by the traditional method.

Fig. 2. Original input images and their corresponding outputs using Sobel edge detection and brush size of 3.

The constructed oil brush strokes achieve a decent effect of imitating authentic oil painting styles. Random noise added to brush order locations help randomize brush strokes in large monotonous color areas, so as to make it vivid enough to mimic real oil brush strokes.

(1) Tuning Brush Size

Outputs by different brush sizes are displayed in Fig. 3. It could be observed from Fig. 3 that larger brush size tends to coarsen the content objects. In our project, brush size was tuned to 5 so that it fits a proper ratio compared to real strokes.

(2) Tuning Edge Detection Methods

The effect of different edge detection algorithms on the output are also tested and analyzed. Three edge detection methods are tried and compared in this research, which are Sobel, Prewitt and Roberts. Fig. 4 shows the difference between their outputs.

Fig. 3. Outputs by different brush sizes. Top: brush size 2; middle: brush size 5; bottom: brush size 8.

Fig. 4. Output by different edge detection methods. Top: Sobel method; middle: Roberts method; bottom: Prewitt method.

The outputs by Roberts and Prewitt method look quite similar, demonstrating an obvious propensity of horizontal and vertical orientation on the strokes, thus causing some spinules protruding from smooth lines. While the Sobel method gracefully smoothes out this effect and makes better transition at straight lines.

(3) Tuning Histogram Specification

In order to better simulate the color and hue style of traditional oil paintings, we tried the histogram specification in the project to see its effect on the final output. The famous representative painting “Liberty Leading the People” was chosen to be the specification target, and Fig. 5 shows the histogram specification output.

Fig. 5. The effect of histogram specification. Top left: original image; top right: specification target; bottom: the model output after applying histogram specificaiton.

After applying histogram specification, the model output gains a dyeing effect towards the preset target painting, which mimics a traditional European oil painting style of the 1800's.

B. Deep Learning Approach

In terms of the deep learning approach, a typical oil painting image with prominent oil brush pattern is chosen as the first style input image for the model, shown in Fig. 6. Thanks to the Tensorflow API, it is possible to output and view the entire process of gradient descent during each training iteration. Fig. 7 demonstrates the evolving of the output image during the training process.

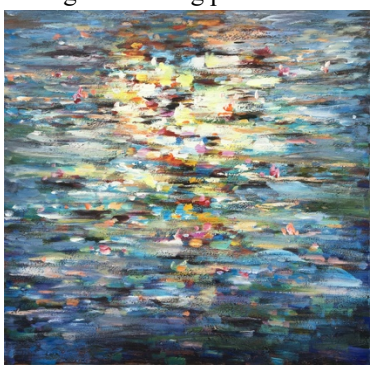

Fig. 6. The first target style image.

With the training going on, the pattern of the style image is bit by bit applied on the content image, which results in a visual effect of texture change. Practically, at the same time, too much training will result in a thorough blend of the content and style input, which to some extend ruins the content object representation. So the learning rate of the optimizer and training epochs have been tuned for outputting an exact degree of image style evolving. Besides, two weights are applied onto the content loss and style loss respectively, to quantitatively control the amount of style change being applied.

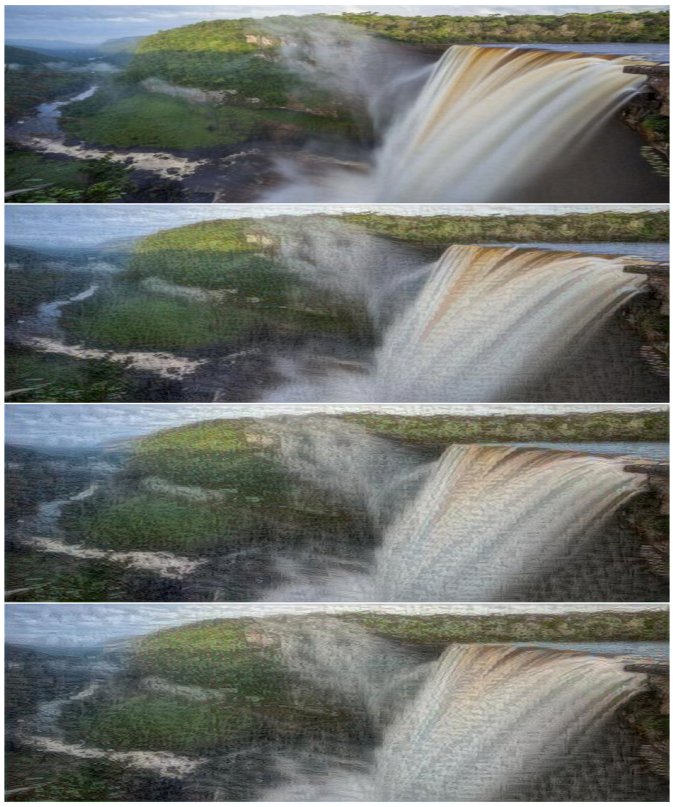

Fig. 7. The original content image and the model output after 10, 30 and 45 iterations of training.

Unlike the traditional approach, the deep learning approach depends on the input style image. The effect of different style inputs on model output is analyzed and displayed in Fig. 8.

Fig. 8. Model outputs on different input style images. Left hand side are style inputs, right hand side are image outputs.

Different style inputs tend to exert some influence on the texture of the output images in terms of pattern style, color, distribution and hue, since the style input is convoluted with each layer, entangling every pixel intensity in the style input with the correlation of the layer filter maps. Thus choosing a decent style input would be decisive for the output appearance and performance.

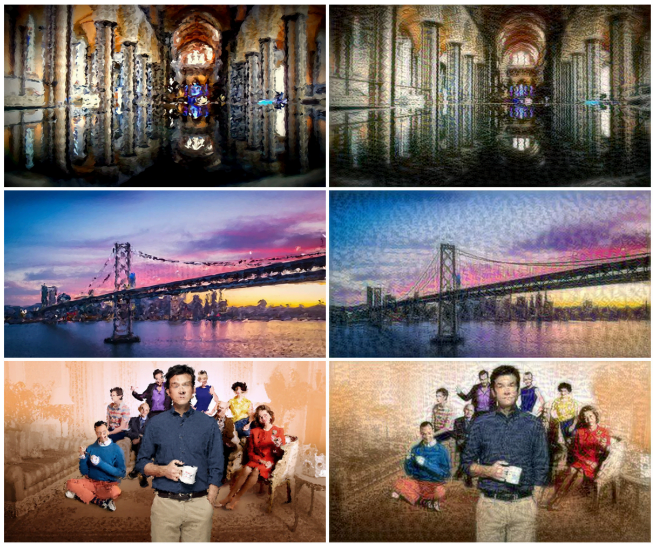

Fig. 9. Output comparison bewtween traditional approach and deep learning approach. Left hand side: by traditional approach; right hand side: by deep learning approach.

Outputs by deep learning model gives a more "real" appearance just like a scanned real world oil painting, compared to the output by traditional approach, which appears more synthetic and ideal. Fig. 9 demonstrates some output comparisons between the traditional approach and the deep learning approach.

If taking a close look at deep learning outputs, it could be observed that the patterns added by the network are similar to a periodic "noise" which are supposed to be apparent in the frequency domain. We deep dived a little bit into the frequency domain using Fourier transform to see this effect. Fig. 10 shows the Fourier spectrum of original images and their corresponding output images by the deep model.

From the Fourier spectrums, we could observe that compared with original input images, the Fourier spectrums of output images tend to have more white dots scattering around the center point, distributed roughly like a round circle ring. This is an indication of the background texture change in the output images. It could be further inferred that, changing the style of an image will result in a change in frequency domain that could be detected and made use of for further enhancements, such as applying low/high pass filters or bandpass filters to achieve more verisimilitude style transfer effect.

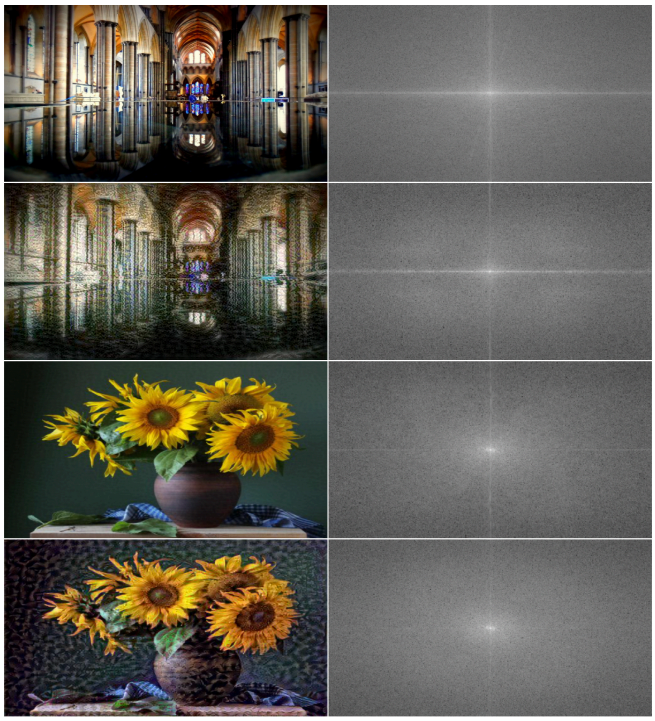

Fig. 10. Fourier spectrum of original images and output images by the deep model. Left hand side: original and output images; right hand side: Fourier spectrum of corresponding images.

IV. Conclusion

In this paper, we focus on the problem of transferring the style of an image into that of oil paintings. Two approaches are discussed and applied, which are the traditional approach and the deep learning approach.

For the traditional approach, we use edge detection techniques to calculate gradient magnitudes and orientations, and simulate the brush strokes using eclipse based on the these gradients. Output images by the traditional approach generally looks like ideally synthesized oil paintings.

For the deep learning approach, we make use of several layers of the VGG-19 network, splitting them into content layers and style layers. Losses are constructed for them respectively to make the model evolve towards a balanced output between content input and style input. The final outputs by deep learning approach appear visually more close to authentic oil paintings, dependent on a proper style input. We analyzed the impact of various parameters on the final output for both approaches, such as brush size, edge detection methods, with/without histogram specification, learning rate, etc., and discussed about representing the style change in frequency domain.

References

[1] H. Fan, Z. Chen, and J. Li, Image Sand Style Painting Algorithm, Applied Mathematics & Information Sciences, vol. 8, no. 2, pp. 765–771, 2014.

[2] Alejo Hausner. 2001. Simulating decorative mosaics. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques (SIGGRAPH '01). Association for Computing Machinery, New York, NY, USA, 573–580.

[3] 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), Las Vegas, NV, 2016, pp. 2414-2423, doi: 10.1109/CVPR.2016.265.

[4] Johnson J., Alahi A., Fei-Fei L. (2016) Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9906. Springer, Cham.

[5] Chen, Dongdong & Yuan, Lu & Liao, Jing & Yu, Nenghai & Hua, Gang. (2017). StyleBank: An Explicit Representation for Neural Image Style Transfer. 2770-2779. 10.1109/CVPR.2017.296.

[6] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. 2017. Universal style transfer via feature transforms. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 385–395.

APPENDIX

A. Source Code for the Traditional Approach

https://github.com/telenovelachuan/digital_image_processing/blob/master/final_project/traditional.ipynb

B. Source Code for the Deep Learning Approach

https://github.com/telenovelachuan/digital_image_processing/blob/master/final_project/deep_learning.ipynb