Comparison of Normal Image Style Transfer Using Convolutional Neural Networks and Fast Image Style Transfer

Tianxiang Chen, Shengze Gao
999473181, 1002935942
Department of Electrical and Computer Engineering
University of Toronto
`tianxiang.chen@mail.utoronto.ca`
`shengze.gao@mail.utoronto.ca`

Abstract

Image Style Transfer, stylizing an image with the styles of other images including artworks, photographies and etc, is a popular topic nowadays. It is traditionally implemented by Convolutional Neural Networks. We implement and compare two distinctive models for Image Style Transfer, the Normal Image Style Transfer and the Fast Image Style Transfer, with the factors of both the image quality and the speed of generation. Based on the result of comparison, conclusions are drawn that the model of Fast Image Style Transfer can generate a stylized image with the quality as high as the model of Normal Image Style Transfer, but with the speed of three orders of magnitude faster. Furthermore, the implementation method of Fast Image Style Transfer can be considered as a general method of speeding up the generation process of Convolutional Neural Networks.

1 Introduction

Image Style Transfer is a subtopic of *image transformation* which is a broad topic including many classic tasks such as image processing, computer vision and etc. Hence, the researches on Image Style Transfer should be influential to its related tasks.

There exist two approaches for Image Style Transfer, one is the Normal Image Style Transfer using a convolutional neural network implemented by Gatys *et al* [1], another one is the Fast Image Style Transfer using a regular convolutional neural network as well as a deep residual convolutional neural network which is implemented by Johnson *et al* [3].

For the first approach, Normal Image Style Transfer, one Deep Convolutional Neural Network is implemented to optimize the weighted perceptual loss function of the content image (image that requires to transform style) and the style image (image with a style the content image needs to transform to). When inputting the content and style image into Convolutional Neural Network, the content and style representations of images should be developed respectively. Since the representations of the content and the style image are independent and separable demonstrated by Gatys *et al* [2], one mixed image with smooth and continuous visual experience can be obtained by combining the content and style image with presented weights. After setting the weights of content image loss and style image loss properly, some example images generated by the training process can be viewed by figure 1.1.

As shown in figure 1.1, the basic structure of mixed images comes from the content image whereas the textures and colors of mixed images inherit from the style image. The visual quality of the mixed image improves with the increasing of iteration numbers of training process.

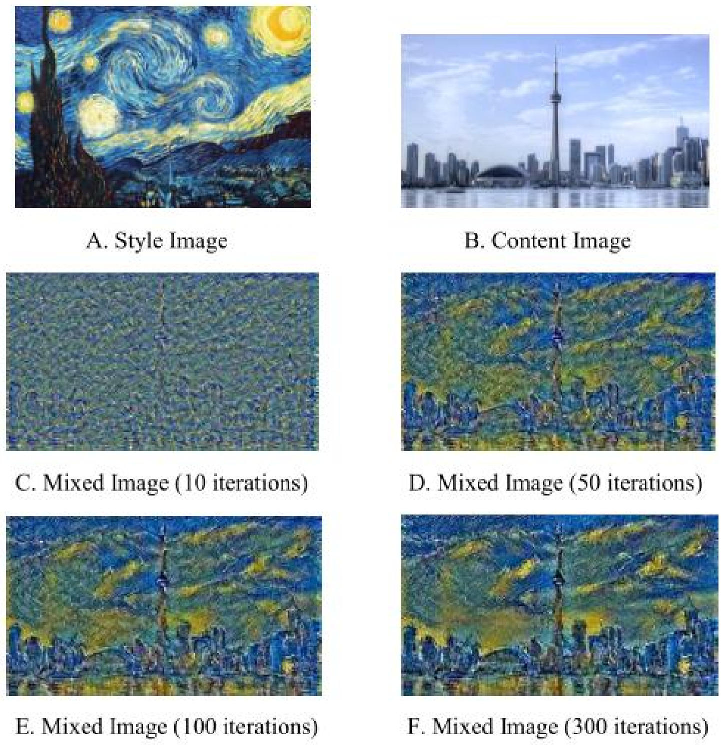
A. Style Image
B. Content Image
C. Mixed Image (10 iterations)
D. Mixed Image (50 iterations)
E. Mixed Image (100 iterations)
F. Mixed Image (300 iterations)

Figure 1.1: The training process of Normal Image Style Transfer. The content image is shown in **A** (Photo: CN Tower). The style image is shown in **B** (*The Starry Night* by Vincent van Gogh, 1889). The mixed images with various numbers of iterations are shown in **C**, **D**, **E** and **F**.

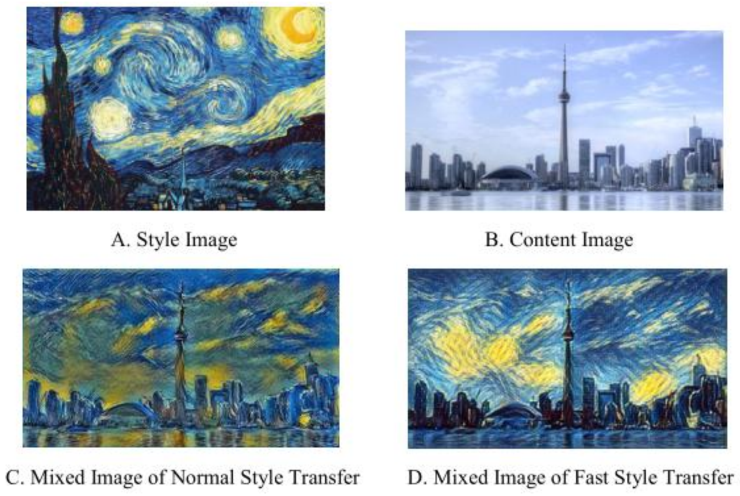
A. Style Image
B. Content Image
C. Mixed Image of Normal Style Transfer
D. Mixed Image of Fast Style Transfer

Figure 1.2: Example mixed images of Normal Image Style Transfer and Fast Image Style Transfer approach.

For the second approach, Fast Image Style Transfer, one Deep Residual Convolutional Neural Network is especially used to generate the result mixed image directly. It is a feed-forward neural network trained by a regular Convolutional Neural Network similar to Normal Image Style Transfer approach. The training process of this model includes both parts of neural networks whereas the generation process only involves the feed-forward neural network. The generation process for this model is simply a forward pass through the trained feed-forward network, thus it has a generation speed three orders of magnitude faster than Normal Image Style Transfer. Figure 1.2 displays the example result mixed images generated by both models.

By viewing figure 1.2, the mixed images generated by two models are similar, which proves that Fast Image Style Transfer is able to generate mixed image with the perceptual quality as high as the Normal Image Style Transfer. Section 4 (Comparison and Experiment) contains more detailed quantitative comparison of these two models.

2 Formal Description

Based on [1][2], we studied and verified the algorithms for Normal Image Style Transfer and Fast Image Style Transfer. In the following subsection, we will introduce this two algorithms based on our understandings from experience of verifying the algorithms from the paper.

2.1 Normal Image Style Transfer

The Normal Image Style Transfer discussed here is proposed by L. A. Gatys, A. S. Ecker and M Bethge[1][2].

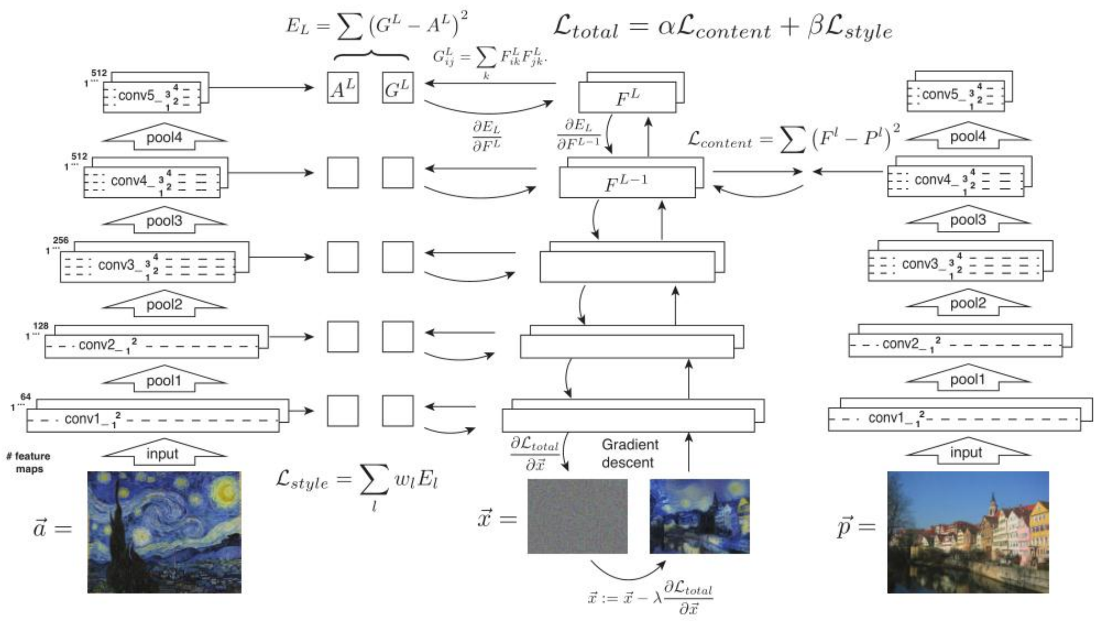
$E_L = \sum (G^L - A^L)^2$
$\mathcal{L}_{total} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style}$
$G^L_{ij} = \sum_k F^L_{ik}F^L_{jk}$
A^L
G^L
F^L
F^{L-1}
$\frac{\partial E_L}{\partial F^L}$
$\frac{\partial E_L}{\partial F^{L-1}}$
$\mathcal{L}_{content} = \sum (F^l - P^l)^2$
conv5_1
conv4_1
conv3_1
conv2_1
conv1_1
pool4
pool3
pool2
pool1
input
512
256
128
64
# feature maps
$\mathcal{L}_{style} = \sum_l w_l E_l$
$\frac{\partial \mathcal{L}_{total}}{\partial \vec{x}}$
Gradient descent
$\vec{a} =$
$\vec{x} =$
$\vec{p} =$
$\vec{x} := \vec{x} - \lambda \frac{\partial \mathcal{L}_{total}}{\partial \vec{x}}$

Figure 2.1.1: Algorithm for Normal Image Style Transfer. Credit:[2]

As shown in the figure 2.1.1, the algorithm contains three parts. On the left and right side, the two networks are for content and style features extraction. They are called only once during one Style Transformation and the extraction results are stored for further use. The middle network, which performs a function of combining the content extraction with style extraction stored before and form the final result picture. By iteratively running the middle network and use the newly-generated combined picture as the initial starting point as the next iteration, it updates and minimizes the losses for the combined image.

The algorithm can be explained in detailed as follow.

Following the discussion in the paper, the algorithm firstly calculates the extraction result for the style and content image and stores them in matrix P and A, respectively.

For the content representation, each layer in the middle network in figure 2.1.1 defines a non-linear filter bank whose complexity increases with the position of the layer in the network. Each layer l has N_l distinct filters mapping to M_l features. the input image $\vec{x}$ will be encoded in each layer of the

Convolutional Neural Network by the filter defined for that layer and the corresponding results is stored in a matrix F^l, where F^l_{ij} is the result of the i^{th} filter at position j in layer j.

For the loss function, the paper defines $\vec{p}$ and $\vec{x}$ be the original image and the image generated, and P^l and F^l be their respective feature representation in layer l, and then the squared-error loss could be expressed as

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2}\sum_{i,j}(F^l_{ij} - P^l_{ij})^2 \tag{2.1}$$

And the derivative of this loss with respect to the activation in layer l equals:

$$\frac{\partial \mathcal{L}_{content}}{\partial F^l_{ij}} = \begin{cases} (F^l - P^l)_{ij}, & \text{if } F^l_{ij} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

For style representation, it uses the Gram matrix, which performs an inner production between the matrices, to capture the feature correlations. The Gram matrix G^l_{ij} is

$$G^l_{ij} = \sum_k F^l_{ik} P^l_{jk} \tag{2.3}$$

The loss of layer l for the style representation can be expressed as

$$E_l = \frac{1}{4N_l^2 M_l^2}\sum_{i,j}(G^l_{ij} - P^l_{ij})^2 \tag{2.4}$$

And the total style loss is

$$\mathcal{L}_{style} = \sum_l \omega_l E_l \tag{2.5}$$

where ω_l is the a weight factor for each layer contributing to the loss.

Finally, for each iteration of running, the algorithm tries to minimize the losses from the summation of the style and content, which is

$$\mathcal{L}_{total} = \alpha \mathcal{L}_{content} + \beta \mathcal{L}_{style} \tag{2.6}$$

where α and β are he weighting factors for content and style reconstruction, respectively.

The algorithm iteratively minimizes losses for equations above and uses the result of last running as the staring point as the new iteration. By performing this, it get a better result as it runs more and more iterations.

2.2 Fast Image Style Transfer

Normal Image Style Transfer method described in 2.1 gives a good result but requires a long runtime for iteratively running the algorithm to find the optimal solution using gradient descent. In 2016, J. Johnson, A. Alahi, F. Li[3] proposed a new method which includes a pretrained neural network and eliminates the backward path in the original network.

As shown in figure 2.2.1, the system has:

- An image transformation network f_W that can be pretrained separately, where the W is a series of weights for the network.
- A loss network ϕ which is used for defining several loss function l_1, ...l_k.

The image transformation network for Fast Image Style Transfer algorithm roughly follows the network architecture proposed by Radford et al [6]. Specifically, it implements with five residual layers, several non-residual layers and an output layer using a scaled tanh according to Johnson et al [3]. The image transformation network takes the input images x and transform it into output ŷ via the mapping function $\hat{y} = f_W(x)$, then each loss function computes the differences between the output image ŷ and a target image y_i. It uses stochastic gradient descent to optimize the weighted loss functions:

$$W^* = arg\,\underset{W}{min}\,\mathbf{E}_{x,y_i}\Big[\sum_{i=1}\lambda_i \ell_i(f_W(x), y_i)\Big] \tag{2.7}$$

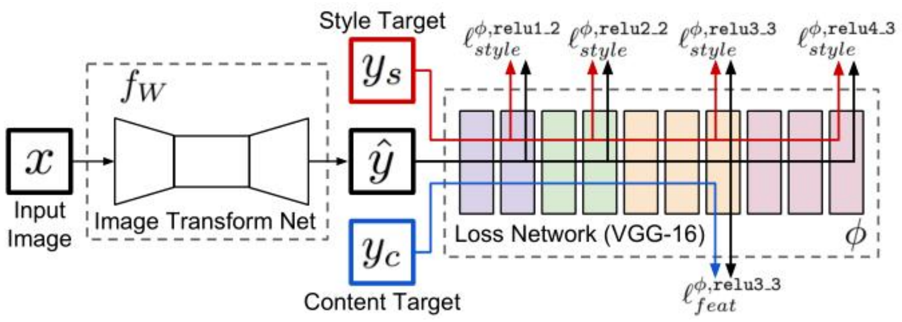
Style Target
y_s
$\ell_{style}^{\phi,\mathrm{relu1\_2}}$
$\ell_{style}^{\phi,\mathrm{relu2\_2}}$
$\ell_{style}^{\phi,\mathrm{relu3\_3}}$
$\ell_{style}^{\phi,\mathrm{relu4\_3}}$
f_W
x
Input Image
Image Transform Net
$\hat{y}$
y_c
Content Target
Loss Network (VGG-16)
ϕ
$\ell_{feat}^{\phi,\mathrm{relu3\_3}}$

Figure 2.2.1: Algorithm for Fast Image Style Transfer. Credit:[3]

3 Related Work

Plenty of related works apply Convolutional Neural Networks for artistic style transfer tasks. For instance, Champandard [13] turns two-bit doodles into artworks and Huang *et al* [14] stylize videos with artistic images. Here we pick two most distinctive algorithms to compare which are also widely implemented on related works mentioned before.

The original algorithm of Normal Image Style Transfer is put forward by Gatys *et al* [1,2] which trains Convolutional Neural Network for style transformation tasks and implements perceptual losses instead of per-pixel losses. Fast Image Style Transfer, presented by Johnson *et al* [3], is a modified algorithm of Gatys's work on style transformation tasks. It is an algorithm training a feedforward network and using the feedforward network to rapidly generate result mixed image. Meanwhile, Johnson also compares his algorithm to the algorithm proposed by Gatys, however, the comparison of mixed image quality is only limited into a qualitative comparison. In our works, the comparison of image quality is done by analyzing the coverage speed of weighted loss for the two algorithms as demonstrated in Section 4.2. Therefore, the quality of generated images can be quantitatively justified by observing the loss curves.

Specifically, our works is based on Hvass [11] and Watson [12] which implement Normal Image Style Transfer algorithm and Fast Image Style Transfer algorithm respectively. To obtain better results and experiment our comparison method, the code is modified as it is shown in the github link [15]. The key attribute for our works is providing a scientific quantitative method for comparing the results of style transfer models.

4 Comparison and Experiment

We perform experiments on quantitative comparison of Normal Image Style Transfer and Fast Image Style Transfer with the factors of image quality and generation speed respectively. Prior works only perform the quantitative comparison of the two algorithms with image generation speed and compare the image quality visually; we compare both factors quantitatively and scientifically prove that Fast Image Style Transfer algorithm can generate a image with the quality as high as Normal Image Style Transfer, but with a generation speed three orders of magnitude faster.

4.1 Training Details

For the Convolutional Neural Networks used for training dataset and optimizing perceptual loss functions, we implement with 16-layer VGG network (VGG-16) [4]. Specifically, for Normal Image Style Transfer model, layers conv1_2, conv2_2, conv3_3 and conv4_3 of VGG-16 loss network are used for calculating style image loss, and layer conv3_3 of VGG-16 loss network is used for calculating content image loss. For Fast Image Style Transfer, exactly same layers are used for the loss network, whereas five residual blocks with the architecture of [7] are implemented for the feedforward transformation network. Moreover, the loss network of the Fast Image Style Transfer model is trained by the Microsoft COCO dataset [5].

For this experiment, images with three different sizes (852x480, 426x240 and 256x144) and identical content (Photo: CN Tower) are implemented as the dataset to test the image quality and generation speed for the two models.

4.2 Comparison of Image Quality

In order to quantitatively compare the image quality of the mixed images generated by the two models, one Convolutional Neural Network exactly the same to the loss network of the two models is constructed. Similarly, this test convolutional neural network takes a style and a content image as input, and contains layers conv1_2, conv2_2, conv3_3 and conv4_3 of VGG-16 network for computing the style image loss and layer conv3_3 of VGG-16 network for computing the content image loss.

To quantify the image quality, firstly, setting the mixed images obtained from the generation process of the two modals as the input content images of the test convolutional neural network respectively. Meanwhile, the input style image for the test network should be the same image as the style image for producing mixed images. Through the training process in the test network, the weighted loss curve can be drawn for these two result mixed images. The faster the curve can be coveraged, the better quality the image has. This comparison method works because both algorithms implement the same loss function of Equation 2.7. Figure 4.2.1 presents the comparison of weighted loss curves for mixed images from the two models.

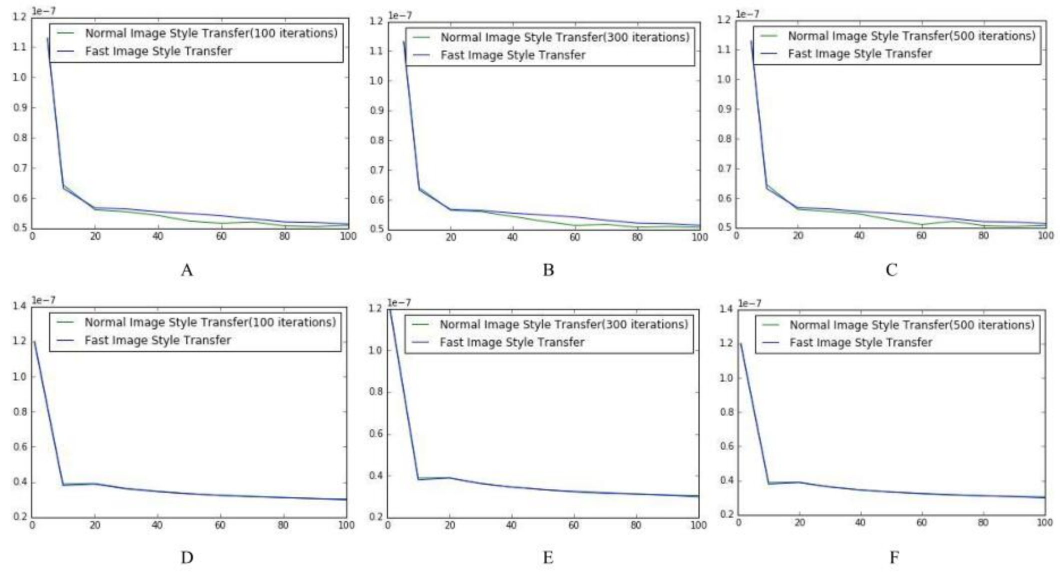

Figure 4.2.1: The comparison of weighted loss curves of mixed images from the two models. The mixed images for curves shown in A, B and C has a image size of 256x144. The mixed images for curves shown in D, E and F has a image size of 426x240. The mixed images generated from Normal Image Style Transfer model in A and D are trained with the iteration number of 100, whereas in B and E is 300 and in C and F is 500.

Derived from figure 4.2.1, the coverage speed for images coming from both models are similarly fast, which proves that the mixed image produced by Fast Image Style Transfer algorithm has an image quality as high as the mixed image generated by Normal Image Style Transfer algorithm.

4.3 Comparison of Generation Speed

To compare the generation speed of these two models, dataset of content images (Photo: CN Tower) with three different sizes are inputted into the two models, Normal Image Style Transfer and Fast Image Style Transfer, and the artwork, The Starry Night by Vincent van Gogh, 1889, is considered as the style image for both models. The comparison of generation speed is shown in table 4.3.1.

image size	Normal Image Style Transfer			Fast Image Style Transfer	Speedup		
	100 iterations	200 iterations	300 iterations		100 iterations	200 iterations	300 iterations
256x144	158.6s	469.1s	779.1s	0.711s	223.1x	659.8x	**1096x**
426x240	425.6s	1281.1s	2120.2s	1.791s	237.6x	715.3x	**1184x**
852x480	1679.3s	5072.2s	8410.7s	6.809s	246.6x	744.9x	**1235x**

Table 4.3.1: The time (in second) for generating mixed images from the two models with various resolutions and numbers of iterations. Speedup calculates the ratio of generation time of Normal Image Style Transfer model over the generation time of Fast Image Style Transfer model. Both models run on the same hardware environment. Both models implemented by CPU, thus the generation speed is slow as it is shown in the table.

Based on the result shown in table 4.3.1, the generation speed of Fast Image Style Transfer model is significantly faster than Normal Image Style Transfer model. As the increasing of the image size, the speedup magnitude increases. Compared to the 500-iteration column of Normal Image Style Transfer, the generation speed of Fast Image Style Transfer method is approximately three orders of magnitude faster.

4.4 Discussions

Through the comparison of Normal Image Style Transfer and Fast Image Style Transfer, the result mixed image quality for the two models are similar, whereas the generation speed of Fast Image Style Transfer is three orders of magnitude faster than Normal Image Style Transfer model. However, in order to train the feedforward transformation network used for generating the mixed image, Fast Image Style Transfer algorithm has to implement a large dataset, Microsoft COCO dataset, to train a regular convolutional neural network for high quality mixed images, which takes several days on CPU. Once the transformation network training process of Fast Image Style Transfer completes, the generation process using pretrained network is distinctively faster than the Normal Image Style Transfer algorithm since it is just a forward pass through the trained feedforward network.

Furthermore, the feedforward architecture implemented in Fast Image Style Transfer can be generalized in many other fields to speed up the generation process of a model with a convolutional neural network. Many works in other related fields implement feedforward networks architecture, such as Dong *et al* [8] in super-resolution problems, Cheng *et al* [9] in colorization problems and Long *et al* [10] in semantic segmentation problems.

5 Limitations

The limitation for our works in mainly caused by hardware issues. Although, for the Fast Image Style Transfer algorithm, it only takes several seconds to generate result images from the pretrained transformation network, the training process of the pretrained network might take tremendous time, depends on the hardware specifications. From the [3], Johnson claimed that with the GPU training (GTX Titan X GPU) it takes ten to twenty hours. For us, it may take couple of months with the laptop we have. Thus, we can only use the trained style images (*The Starry Night* and *Candy*) from Watson's github [12] as the pretrained network data. Also, because of the limit of our hardware resources, the generation speed for both implemented algorithms is slower than the speed presented in [3].

The major limitation for the quantitative comparison method proposed in Section 4 is that it require relatively strict conditions for the compared models. For instance, method of obtaining weighted loss functions and the layer architecture of the loss networks for the two models have to be the same; otherwise, the comparison result should be unfair and invalid.

6 Conclusion

Based on the result of comparison, the generated image of Fast Image Style Transfer contains the quality as high as the image generated by Normal Image Style Transfer, but with a generation speed about three orders of magnitude faster. By applying the dataset of input content images with various sizes, the quantitative comparison of image quality and generation speed for Normal Image Style Transfer and Fast Image Style Transfer can be thoroughly completed. Moreover, the dataset for comparing the image quality also includes mixed images with various numbers of iterations in the Normal Image Style Transfer algorithm to ensure the comparison process is executed fairly. As a result, Fast Image Style Transfer algorithm is able to produce a high quality result image with faster speed than Normal Image Style Transfer algorithm. For further extensions, the implementation method of Fast Image Style Transfer is also valuable for many related tasks including super-resolution, colorization, semantic segmentation and other relevant fields which may already implemented similar techniques.

References

[1] Leon A. Gatys & Alexander S. Ecker &Matthias Bethge Image Style Transfer Using Convolutional Neural Networks, *CVPR, 2016.*

[2] Leon A. Gatys & Alexander S. Ecker &Matthias Bethge A Neural Algorithm of Artistic Style, *Nature Communications, 2015.*

[3] Johnson, J. & Alahi, A. & Fei-Fei, L. Perceptual Losses for Real-Time Style Transfer and Super-Resolution, *ECCV, 2016.*

[4] Simonyan, K. & Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556 (2014)*

[5] Lin, T.Y. &Maire, M. &Belongie &S., Hays &J. &Perona, P. &Ramanan, D. &Dollár, P. &Zitnick, C.L. Microsoft coco: Common objects in context. *In: Computer Vision–ECCV 2014. Springer (2014) 740–755*

[6] Radford, A. &Metz, L. &Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434 (2015)*

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

[8] Dong, C. &Loy, C.C. &He, K. &Tang, X. Image super-resolution using deep convolutional networks. *(2015)*

[9] Cheng, Z. &Yang, Q. &Sheng, B. Deep colorization. *In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 415–423*

[10] Long, J. &Shelhamer, E. &Darrell, T. Fully convolutional networks for semantic segmentation. *CVPR (2015)*

[11] M. Pedersen *https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/15_Style_Transfer.ipynb*

[12] Watson, G. *https://github.com/ghwatson/faststyle*

[13] Champandard, A. Semantic Style Transfer and Turning Two-Bit Doodles into Fine Artwork. *(2016)*

[14] Huang, H. &Wang, H. &Luo, W. &Ma, L. &Jiang, W. &Zhu, X. &Li, Z. &Liu, W. Real-Time Neural Style Transfer for Videos. *(2016)*

[15] Authors' github link: *https://github.com/shengze/Normal-Style-Transfer-and-Fast-Style-Transfer*