An Improved Image Transformation Network for Neural Style Transfer

Qiu Hong-Xu and Huang Xiao-Xia*

Dept. of Computer Sciences, College of Information & Engineering, Shanghai Maritime University, Shanghai 201306, China
{qiuhongxu93, lhuangxiaoxia}@163.com

Abstract. By using Convolutional Neural Networks(CNNs), the semantics and styles of images can be separated and recombined to create fascinating images. In this paper, an image transformation network for style transfer is proposed, which consists of convolution layers, deconvolution layers and Fusion modules composed of two 1x1 convolution layers and a residual block. The output of each layer in the network is normalized using batch normalization to speed up the training process. Compared with other networks, our network has fewer parameters and better real-time performance while generating similar quality images..

Keywords: Style Transfer, Batch Normalization, Residual Block, Convolutional Neural Networks

* Fund project: The 48th Project Sponsored by the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry; Shanghai Science and Technology Commission Innovation Project (16DZ1201402).

1 Introduction

For centuries, painting has been a popular form of art, having produced plenty of valuable masterpieces which attract people's attention. But in the past, it would take a long time for a well-trained artist to draw a painting of a particular style.

Recently, Gatys et al. first studied how to use CNN to reproduce famous painting styles in the natural picture. They obtained the representations of the image from the CNN and found that the content of image and the style of image were separable. Based on the above findings, Gatys et al. proposed a Neural Style Transfer algorithm [1] to recombine the contents of a given image and the style of famous artworks. However, the efficiency of his algorithm can't meet the real-time requirements. Johnson et al. introduced a fast method based on the algorithm proposed by Gatys et al. Firstly, they trained an equivalent feed-forward generator network by using the perceptual loss function [2] they proposed for each particular style. The perceptual loss function calculates the loss by using high-level features extracted from the images using the 16-layer VGG network [3] pretrained on ImageNet dataset [4]. When there is a content image to be stylized, only a single forward transfer is required to produce the result.

Due to the amazing stylized results, the study of Neural Style Transfer has led to many successful industrial applications. The mobile application Prisma [5] is one of the first industrial applications that offer the Neural Style Transfer algorithm as a service. Before Prisma appeared, people never thought that one day they could turn their images into artworks in just a few minutes. In order to meet the growing needs of the mobile end, a smaller and faster network is urgently needed.

In this paper, a new module which can be used to construct an image transformation network is proposed. In order to train the network, the pretrained 16-layer VGG network is used to extract the advanced features of the image, then train our network by minimizing the perceptual loss function. At test-time, compared with the network of Johnson et al., our transformation networks reduce network parameters by 62.3% and the running time reduced by 12%.

2 Related Work

2.1 Feed-forward Image Transformation

In recent years, training a deep convolutional neural network can solve many image transformation problems. Since the purpose of our image transformation network is to convert an image into a stylized image, we referred to the architecture of the Fully Convolutional Networks [6] and the Deconvolution Network [7]. In the architecture of our image transformation network, instead of using the pooling layer, the convolution layer and the deconvolution layer are used to perform the down-sampling and up-sampling operations.

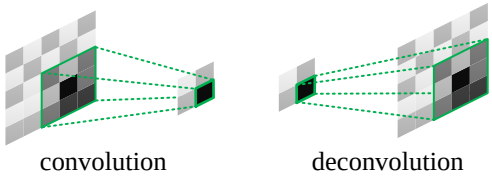

Fig. 1. Convolution and deconvolution operations for downsampling and upsampling.

2.2 Neural Style Transfer

The method proposed by Gatys et al. starts with random noise and updating the stylized image constantly by back propagation. The method of Johnson et al. is to train a feedforward network on a large image dataset for each particular style of image. Gradient descent is used to optimize the network by iteratively updating the network. Those two methods use similar objective function.

The perceptual loss function is improved by Johnson et al. on the basis of Gatys et al. Given the style and content images y_s and y_c, and the layer *j* and *J* used in the network ϕ for feature and style reconstruction, the stylized images can be generated by minimizing the total loss.

$$\text{loss} = \lambda_c l_{feat}^{\phi,j}(y, y_c) + \lambda_s l_{style}^{\phi,J}(\text{y}, y_s) + \lambda_{TV} l_{TV}(y) \quad (1)$$

λ_c, λ_s and λ_{TV} are scalars representing the weights of feature reconstruction loss, style reconstruction loss and total variation regularizer in the total loss. In this paper, the network ϕ is the pretrained 16-layer VGG network.

2.3 Batch Normalization

Because of change in the parameters of the previous layer, the distribution of each layer's inputs changes during training. It makes the depth neural network difficult to train.

To solve this problem, Ioffe et al. proposed batch normalization [8]. By using batch normalization, the internal covariate shift is reduced and the training process of network is shortened greatly. Batch normalization is implemented by normalizing each feature map so that the mean is zero and the variance is one. For a layer with d-dimensional input $\text{x} = (x^{(1)} \ldots x^{(d)})$, each dimensional will be normalized

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}} \quad (2)$$

where the expectation and variance are computed over the training data set.

In this paper, batch normalization is added after the output of each convolutional and deconvolutional layer.

2.4 Residual Connection

He et al. found the use of residual connections [9] in the network which made it possible to train deep convolutional neural networks. They used residual connections

on various datasets and proved the effect of residual connections. The residual block is defined as:

$$y = F(x, \{W_i\}) + x \quad (3)$$

Here x and y are the input and output vectors of the layers. The function $F(x, \{W_i\})$ is a residual mapping that needs to be learned. The structure of the residual block is shown in **Fig. 2**.

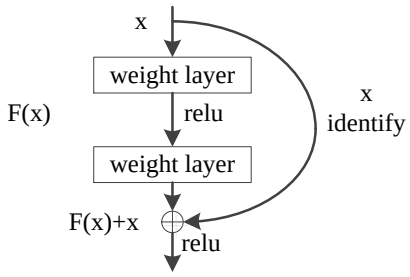

Fig. 2. Residual block.

3 Image Transformation Network

A module we designed with fewer parameters is proposed in this part. First, we introduced design ideas of modules with fewer parameters. Then, we proposed a module called **Fusion** which enabled us to build an image transformation.

3.1 Design Idea

Our objective is to define a CNN module with fewer parameters while keeping the image transformation network capability of generating similar quality images. To achieve it, the following ideas are used in the design of module:

Use 1x1 filters. Given a certain number of filters, the module will use more of the 1x1 filters because the parameters of 1x1 filters are 9 times less than the parameters of 3x3 filters.

Reduce the number of input channels to the 3x3 filters. When the layer contains 3x3 filters, the convolution layer will have (number of input channels) * (number of filters) * (3 * 3) parameters. Obviously, the parameters of the convolution layer can be effectively reduced by reducing the number of filters and the number of input channels. It is crucial to reduce the number of input channels and the number of filters to reduce the convolution layer parameters.

These ideas will be applied to getting a novel module and utilizing it on the main body of the image transformation network. The first layers of the network are the convolution layers used to downsampling, and the last layers of the network are the deconvolution layers for upsampling.

3.2 The Fusion Module

The **Fusion** module is defined as follows.

$$y = \text{joint}\{W^2_{1\times1}(W^1_{1\times1}x), F(W^1_{1\times1}x) + W^1_{1\times1}x\} \quad (4)$$

Here, x and y are module inputs and outputs, the function F is the residual mapping mentioned above. $W^1_{1\times1}$ is the first 1x1 convolution layer used to reduce the number of input channel. Then, the output of the first 1x1 convolution layer is fed into second 1x1 convolution layer $W^2_{1\times1}$ and a residual block respectively. Finally, the output of the second 1x1 convolution layer and the output of the residual block are jointed as output of the module.

In the **Fusion** module, the residual block consists of two 3x3 convolution layers. The output of each convolution layer is normalized by batch normalization and the activation function is ReLU. The **Fusion** module is illustrated in **Fig. 3**.

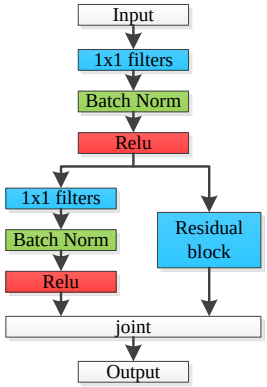

Fig. 3. The architecture of **Fusion** module.

3.3 The Transformation Network Architecture

The transformation network architecture is shown in **Fig. 4**. The image transformation network starts with three convolution layers used for downsampling, followed by 5 **Fusion** modules, and finally ends with three deconvolution layers used for upsampling to generate the final image.

In **Fig. 5**, it is obvious that parameters in our network are much lower than those in Johnson et al. Compared with their network, our network parameters decreased by 62.3%.

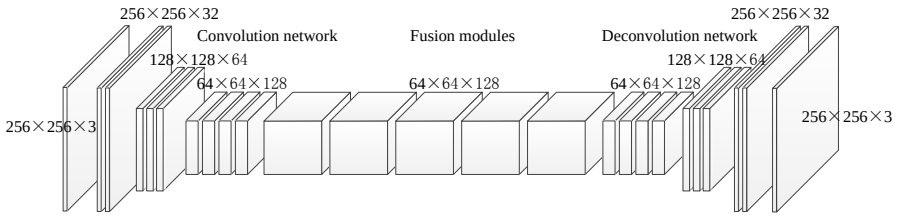

Fig. 4. The architecture of transformation network.

Layer name	output size	filter size/stride	depth	parameter
input image	256×256×3			
conv1	256×256×32	9×9/1	1	7,808
conv2	128×128×64	3×3/2	1	18,496
conv3	64×64×128	3×3/2	1	73,856
resid1	64×64×128		2	295,168
resid2	64×64×128		2	295,168
resid3	64×64×128		2	295,168
resid4	64×64×128		2	295,168
resid5	64×64×128		2	295,168
conv5	128×128×64	3×3/(1/2)	1	73,792
conv6	256×256×32	3×3/(1/2)	1	18,464
conv7	256×256×3	9×9/1	1	7,779
output image	256×256×3			
			16 (total)	1,676,035 (total)

(a)

Layer name	output size	filter size/stride	depth	parameter
input image	256×256×3			
conv1	256×256×32	9×9/1	1	7,808
conv2	128×128×64	3×3/2	1	18,496
conv3	64×64×128	3×3/2	1	73,856
fusion1	64×64×128		3	86,272
fusion2	64×64×128		3	86,272
fusion3	64×64×128		3	86,272
fusion4	64×64×128		3	86,272
fusion5	64×64×128		3	86,272
conv5	128×128×64	3×3/(1/2)	1	73,792
conv6	256×256×32	3×3/(1/2)	1	18,464
conv7	256×256×3	9×9/1	1	7,779
output image	256×256×3			
			21 (total)	631,555 (total)

(b)

Fig. 5. Parameters for each layer of the network. (a)The network of Johnson et al. (b) Our network.

4 Experiments

We use the perceptual loss function as the objective function to train the image transform network. The Microsoft COCO dataset [10] is used to train our image transformation network. Our implementation uses Tensorflow [11] and cuDNN [12]. The whole training process is accomplished on a single GTX 950M GPU. At the same time, we also have reimplemented the work of Johnson et al. In **Fig. 6**, our image transformation network has generated images with similar quality to Johnson et al.

In **Table 1**, we compared the runtime of two networks on different sizes of images. In different sizes of images, our network has a faster speed. Compare to the network of Johnson et al., our network is 12% faster on average. Experiments show that our network can meet the higher real-time requirements.

Since the image transformation network is a fully convolutional neural network, it can be applied to any resolution image as long as the machine memory is enough.

Table 1. For different sizes of images, the table lists the speed of our network and the speed of the network of Johnson et al. Our network gets stylized images of similar quality but is faster than the network of Johnson *et al.* Both methods are implemented on GTX 950M GPU.

Image size	Johnson et al	Ours	Speedup
256×256	0.098s	0.084s	14.3%
512×512	0.335s	0.296s	11.6%
1024×1024	1.244s	1.1s	11.6%

(a) (b) (c) (d)

Fig. 6. (a) The content images. (b) The style images. (c) Results of Johnson et al. (d) Our results.

5 Conclusion

In this paper, **Fusion** module is proposed based on some design ideas, and an image transformation network is constructed with the module to obtain a network with fewer parameters and better real-time performance. Experimental results show that, compared with the network of Johnson et al., the number of parameters in our network is greatly reduced and the runtime is shortened when similar quality images are generated.

In the future, we would like to explore a better network architecture in image transformation tasks. We hope that an image transformation network with smaller and higher real-time performance can be applied to smart phones, which enable them to partly break away from the constraints of mobile phone hardware conditions.

References

1. Gatys, L.A., Ecker, A.S., Bethge, M.: A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576 (2015)
2. 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
3. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015)
4. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet large scale visual recognition challenge. Int. J. Comput. Vis. (IJCV) **115**(3), 211–252 (2015)
5. I. Prisma Labs. Prisma: Turn memories into art using artificial intelligence. (2016)
6. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015)
7. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: ICCV (2015)
8. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: ICML (2015)

9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
10. 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: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014, Part V. LNCS, vol. 8693, pp. 740–755. Springer, Heidelberg (2014)
11. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: A system for large-scale machine learnin. arXiv preprint arXiv:1605.08695(2016)
12. Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., Shelhamer, E.: cuDNN: efficient primitives for deep learning. arXiv preprint arXiv:1410.0759(2014)