Conditional Fast Style Transfer Network

Keiji Yanai and Ryosuke Tanno
The University of Electro-Communications, Tokyo
1-5-1 Chofugaoka
Chofu-shi, Tokyo 182-8585
yanai@cs.uec.ac.jp,tanno-r@mm.inf.uec.ac.jp

ABSTRACT

In this paper, we propose a conditional fast neural style transfer network. We extend the network proposed as a fast neural style transfer network by Johnson et al. [8] so that the network can learn multiple styles at the same time. To do that, we add a conditional input which selects a style to be transferred out of the trained styles. In addition, we show that the proposed network can mix multiple styles, although the network is trained with each of the training styles independently. The proposed network can also transfer different styles to the different parts of a given image at the same time, which we call "spatial style transfer". In the experiments, we confirmed that no quality degradation occurred in the multi-style network compared to the single network, and linear-weighted multi-style fusion enabled us to generate various kinds of new styles which are different from the trained single styles. In addition, we also introduce a mobile implementation of the proposed network which runs in about 5 fps on an iPhone 7 Plus.

KEYWORDS

neural style transfer, CNN, conv-deconv network

ACM Reference format:
Keiji Yanai and Ryosuke Tanno. 2017. Conditional Fast Style Transfer Network. In *Proceedings of ICMR '17, Bucharest, Romania, June 6–9, 2017*, 4 pages.
https://doi.org/10.1145/3078971.3079037

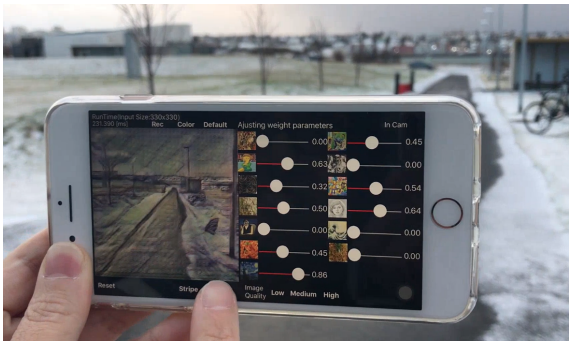

Figure 1: The world's first real-time multi-style-mixing transfer app, *"DeepStyleCam"*, running in about 5 fps on an iPhone 7 Plus. See the demo video in the supplementary material. This app is available on the iOS App Store. Please search for *"DeepStyleCam"*.

1 INTRODUCTION

The neural style transfer method was proposed by Gatys et al. [4, 5] in 2015. This method synthesizes an image which has the style of a given style image and the contents of a given content image using Convolutional Neural Network (CNN). For example, by integrating the style of Gogh's starry night and the content of a lion photo, we obtain a lion painting in the style of Gogh. This method enables us to modify the style of an image keeping the content of the image easily. It replaces the information which are degraded while the signal of the content image goes forward through the CNN layers with style information extracted from the style image, and reconstructs a new image which has the same content as a given content images and the same style as a given style image.

However, since the method proposed by Gatys et al. required forward and backward computation iteratively to synthesize a composite image (in general several hundreds times), the processing time tends to become longer (several tens of seconds) even using GPU. Then, several methods using only feed-forward computation of CNN to realize style transfer have been proposed so far [8, 11]. One of them is the method proposed by Johnson et al. [8]. They proposed perceptual loss functions to train a ConvDeconv-style network as a feed-forward style transfer network. Their network consists of down-sampling layers, convolutional layers and up-sampling layers, which accepts an content image and outputs an synthesized image integrated with a fixed pre-trained style. ConvDeconv-style networks are commonly used for image transformation tasks such as super-resolution [1] and coloring of gray-scale images [6]. In their method, they train a ConvDeconv-style network so that the style matrix of its output image becomes closer to the style matrix of the given fixed style image and the CNN features of the input image leaves unchanged by using the proposed perceptual losses. By using the perceptual loss functions, we do not need to prepare transferred target images as ground-truth images.

However, their feed-forward style transfer network can treat only one fixed style. If transferring ten kinds of styles, we have to train different ConvDeconv networks independently and have to hold all the network parameters. This is inconvenient for mobile implementation in terms of required memory size. Then, in this paper, we modified Johnson et al.'s method so that a single ConvDeconv network can train multiple styles at the same time by adding an conditional input. We call our network as "a conditional fast style transfer network".

By adding an conditional input which selects a style to be transferred to a given input image among the trained styles, we have enabled a ConvDeconv style fast style transfer network to train multiple styles with one network. This network cannot only select

ICMR '17, June 6–9, 2017, Bucharest, Romania

ACM ISBN 978-1-4503-4701-3/17/06…$15.00.
https://doi.org/10.1145/3078971.3079037

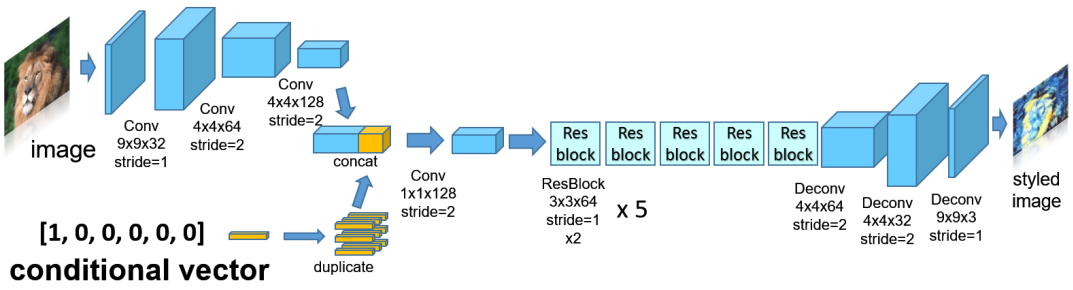

Figure 2: A conditional fast style transfer network.

one of the style out of the pre-trained styles but also mix multiple styles with arbitrary mixing weights. In addition, this network has ability to perform spatial style transfer which means that it can transfer different styles to the different parts of a given image through one-time feed-forward computation of the network according to the conditional signal.

In this paper, we introduce our conditional fast style transfer network and show its effectiveness regarding multiple style transfer, linear weighting style mixing, and spatial style transfer.

Note that recently Dumoulin et al. [2] proposed a method to learn multiple styles with the ConvDeconv style fast style transfer network proposed by Johnson et al. They used Instance Normalization [12] instead of Batch Normalization [7] for normalization of activation signals, and they proposed to replace scale and bias parameters of instance normalization layers depending on the styles. They call this as "conditional instance normalization". The idea on introducing conditions for style selection is similar to ours. However, they introduced a new special layer, "conditional instance normalization" layer, which is needed to be newly implemented for the existing deep learning frameworks such as Caffe, Tensorflow and Chainer, while in our method we added an additional input signal which is concatenated with an internal activation signal and we introduced one additional 1×1 convolution layer to integrate an internal signal and a style signal. We use only common layers in the proposed networks. In addition, in their method, to mix multiple styles, they have to calculate linear-weighted scale and bias parameters in advance, while we just assign weight values to the multiple elements of the conditional input vector such as $(0.2, 0.3, 0.1, 0.4)$.

2 CONDITIONAL FAST STYLE NETWORK

We modified the ConvDeconvNetwork used in [8] by adding a style condition input and an additional 1×1 convolutional layer for fusion of feature map activations and a conditional signal as shown in Figure 2. This network is inspired by Iizuka et al's CNN-based coloring work [6]. They proposed adding a scene contextual stream to the ConvDeconv-style colorization network. They transform a scene vector, which is an output vector of scene recognition network, to a scene feature map by making the same size of copies as the activation feature map, and concatenate it with the intermediate feature map of the activations of the ConvDeconv colorization network. With this improvement, they achieved coloring depending on the scene content of a target image.

When training, we prepare s style images, and make a mini-batch with the combinations of one content training image and all

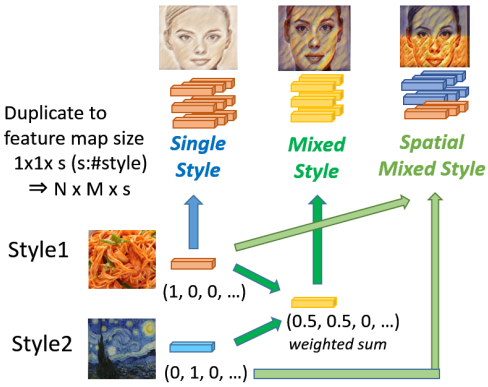

Figure 3: Three different ways to create the conditional input: (1) single style (2) mixed style by linear-weighting (3) spatially mixed style.

the style training images. That is, one mini-batch contains s combinations. To train the network, we provide a content image and one-hot conditional vectors which are s-dimensional binary vectors where the corresponding element to each of the styles is one and the rest elements are all zero. As shown in Figure 2, the conditional vector is $(1, 0, 0, \ldots)$ for the style no.1, while it is $(0, 1, 0, \ldots)$ for the style no.2. As a loss function, we use the perceptual loss proposed by Johnson et al. [8]. In the same way, we use VGG-16 [10] as the loss network, and optimize the weights of the network so that the content feature of the output image extracted from the CONV3_3 layer of VGG-16 gets closer to the one of the input content image and the style features of the output image extracted from the CONV1_2, CONV2_2, CONV3_3 and CONV4_3 layers from VGG-16 get closer to the ones of the correponding style image. Note that content features are feature map activations, while style features are the elements of Gram matrix of feature map activations. The loss functions can be written as follows:

$$
\begin{aligned}
L(w) &= \lambda_s L_s(w) + \lambda_c L_c(w) \\
&= \frac{1}{4} \sum_{l \in \{1\_2, 2\_2, 3\_3, 4\_3\}} ||G(\phi_l(I_{out}(w))) - G(\phi_l(I_s))||^2 \\
&+ \sum_{l \in \{3\_3\}} ||\phi_l(I_{out}(w))) - \phi_l(I_c)||^2
\end{aligned}
$$

where w represents the network parameter set, and $I_{out}(w)$, I_c, and I_s represents the image generated by the network with the parameter set w, a content training image and a style training image, respectively. Please refer the detail on training to [8].

When generating images, we provide a content input image and and a style conditional vector which indicates the styles a user like to transfer to the input image. Although at the time of training we use only one-hot vectors as a style vector, at the time of image generation, we can use the vector each of the elements of which is between 0 and 1 as a style-mixing weight vector. By using the vector which have two or more non-zero elements, we can mix multiple weights as shown in Figure 3(2). This is an interesting characteristic. We guess this comes from the linearity of neural networks. As a matter of course, by providing an one-hot vector as a conditional vector, we can select one single style among the pre-trained styles as shown in Figure 3(1).

In addition, this network has ability to perform spatial style transfer. As shown in Figure 3(3), if we combine multiple style vectors spatially, we can achieve spatial style transfer easily. In this case, we can use mixed style weight vectors as base style vectors.

3 EXPERIMENTAL RESULTS

In the experiment, we trained the network with 13 style images shown in the top row of Figure 4. From the second column to the fifth in the figure, we show the output images generated by single style transfer with the proposed multiple style transfer network using one-hot vectors as conditional vectors. Compared to the same transformation results shown in Figure 5 by single style transfer networks, the quality of the generated images are comparable, which shows the effectiveness of the proposed network. Regarding the processing speed, both are almost the same, since only one convolution layer is added to the original one.

In Figure 4 from the sixth column to the most left, we show the mixed-style transferred images. By adding the style no.10 to the style no.4, the result image became brighter. By mixing multiple styles, we can create new styles as we like. Random weight mixing is also interesting as shown in the left of the figure. Figure 6 shows results when changing the weights gradually between the style no.7 to no.8.

Figure 7 shows the results of spatial style transfer with the proposed network. The images in the left and the right were transformed with three different random weights regarding three vertically-divided regions. The images in the middle were transformed with three different random weights regarding three horizontally-divided regions. For future work, it is an interesting application to combine it with semantic segmentation.

4 IMPLEMENTATION AS IOS APP

We have implemented the proposed method. We modified the CNN network so that the amount of computation is reduced one tenth compared to the original multi-style network. Regarding mobile implementation, we followed the work on Efficient Mobile CNN Implementation [13]. In the method, CNN networks are directly converted to a C source code which utilized multi-threading and iOS Accelerate Framework for CNN computation. We have realized a real-time multi-style-mixing transfer app running in about 5 fps on an iPhone 7 Plus as show in Figure 1. The demo video of this app is attached as the supplementary material.

We shrunk the ConvDeconvNetwork compared to [8] to save computation costs. We added one down-sampling layer and up-sampling layer, replaced 9x9 kernels with smaller 5x5 kernels in the first and last convolutional layers, and reduced five Residual Elements into three. The detail of the network architecture is shown in Figure 8. We confirmed that these network shrinking did not harm the quality of outputs significantly. To accelerate computation of deconvolution layers, we use the sub-pixel convolutional layer proposed by Shi et al. [9], which realizes a deconvolution layer with n kernels and stride 2 using a convolution layer with $4n$ kernels and stride 1.

In addition, we implemented color preserving mode [3] which transfers selected styles only into gray-scale elements of an input image. It can keep the color of the content image while changing only the intensity of pixels, which is especially suitable for food images.

5 CONCLUSIONS

We proposed a conditional fast style network which we added a conditional vector input to the standard ConvDeconv network. Although at the time of training we use only one-hot vectors as a style vector, at the time of image generation, we can use the vector each of the elements of which is between 0 and 1 as a style-mixing weight vector. By only changing a style input vector, we can mix multiple styles out of the pre-trained styles with arbitrary weights. In addition, we implemented the world's first real-time multi-style-mixing transfer app running in about 5 fps on an iPhone 7 Plus. The app for iOS can be downloaded from http://bit.ly/deepstylecam.

For future work, we will extend the proposed method for unseen style transfer and an end-to-end network combining the proposed network with semantic segmentation.

Acknowledgment This work was supported by JSPS KAKENHI Grant Numbers 15H05915, 17H05972 and 17H06026.

REFERENCES

[1] C. Dong, C. C. Loy, K. He, and X. Tang. 2014. Learning a deep convolutional network for image super-resolution. In *Proc. of European Conference on Computer Vision*. 184–199.
[2] V. Dumoulin, J. Shlens, and M. Kudlur. 2016. A Learned Representation For Artistic Style. In *arXiv:1610.07629*.
[3] L. A. Gatys, M. Bethge, A. Hertzmann, and E. Shechtman. 2016. Preserving Color in Neural Artistic Style Transfer. In *ArXiv:1606.05897*.
[4] L. A. Gatys, A. S. Ecker, and M. Bethge. 2015. A Neural Algorithm of Artistic Style. In *arXiv:1508.06576*.
[5] L. A. Gatys, A. S. Ecker, and M. Bethge. 2016. Image Style Transfer Using Convolutional Neural Networks. In *Proc. of IEEE Computer Vision and Pattern Recognition*.
[6] S. Iizuka, E. Simo-Serra, and H. Ishikawa. 2016. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2016)* 35, 4 (2016).
[7] S. Ioffe and C. Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. of International Conference on Machine Learning*. 448–456.
[8] J. Johnson, A. Alahi, and L. F. Fei. 2016. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *Proc. of European Conference on Computer Vision*.
[9] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. 2016. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. In *Proc. of IEEE Computer Vision and Pattern Recognition*.
[10] K. Simonyan, A. Vedaldi, and A. Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.
[11] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. 2016. Texture networks: Feed-forward synthesis of textures and stylized images. In *Internation Conference on Machine Learning*.
[12] D. Ulyanov, A. Vedaldi, and V. Lempitsky. 2016. Instance Normalization: The Missing Ingredient for Fast Stylization. In *arXiv:1607.08022*.
[13] K. Yanai, R. Tanno, and K. Okamoto. 2016. Efficient Mobile Implementation of A CNN-based Object Recognition System. In *Proc. of ACM International Conference Multimedia*.

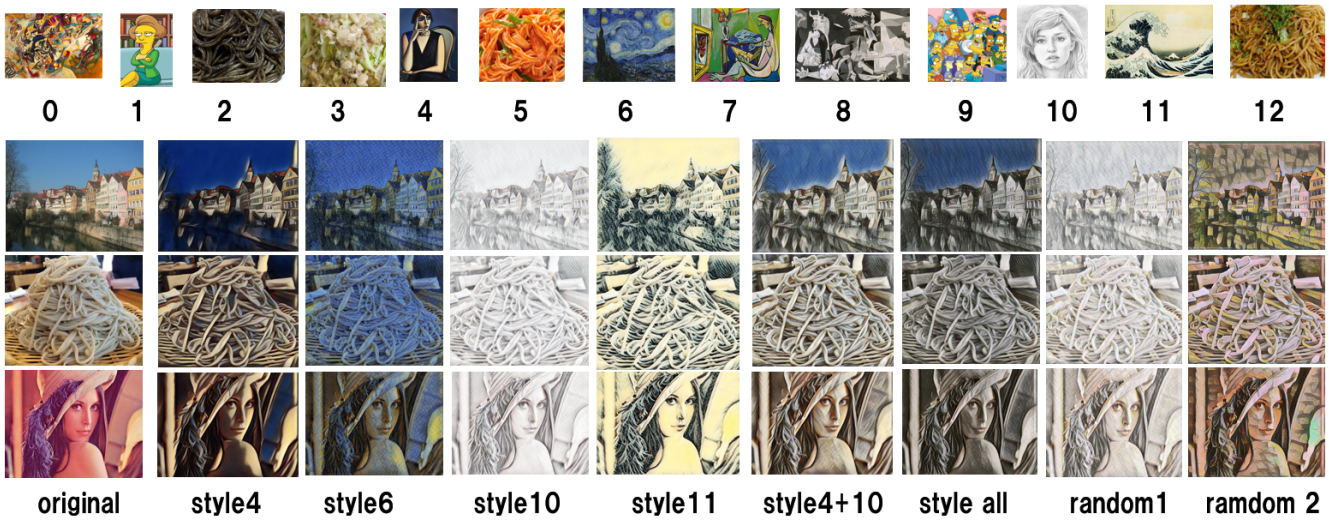

Figure 4: Results by the network trained with 13 styles shown in the top row. From Column 2 to 5, the results with a single style, and from column 6 to 9, the results with mixed styles.

Figure 5: Results by the single network which is equivalent to the one proposed by Johnson et al. [8].

Figure 6: Gradual change of the weights between style no.7 and no.8.

Figure 7: Results of the spatial style transfer.

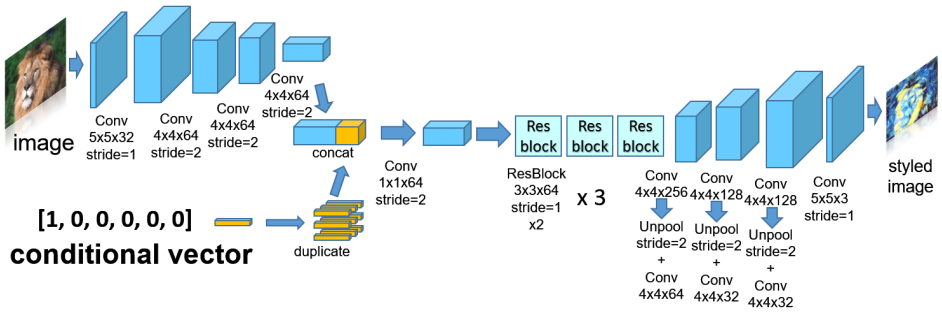

Figure 8: A modified conditional fast style transfer network for mobile implementation.