Balancing Content and Style with Two-Stream FCNs for Style Transfer

Duc Minh Vo[†] vmduc@nii.ac.jp
Trung-Nghia Le[†] ltnghia@nii.ac.jp
Akihiro Sugimoto[‡] sugimoto@nii.ac.jp

[†] SOKENDAI (The Graduate University for Advanced Studies), Japan

[‡] National Institute of Informatics, Japan

Abstract

Style transfer is to render given image contents in given styles, and it has an important role in both computer vision fundamental research and industrial applications. Following the success of deep learning based approaches, this problem has been re-launched very recently, but still remains a difficult task because of trade-off between preserving contents and faithful rendering of styles. In this paper, we propose an end-to-end two-stream Fully Convolutional Networks (FCNs) aiming at balancing the contributions of the content and the style in rendered images. Our proposed network consists of the encoder and decoder parts. The encoder part utilizes a FCN for content and a FCN for style where the two FCNs are independently trained to preserve the semantic content and to learn the faithful style representation in each. The semantic content feature and the style representation feature are then concatenated adaptively and fed into the decoder to generate style-transferred (stylized) images. In order to train our proposed network, we employ a loss network, the pre-trained VGG-16, to compute content loss and style loss, both of which are efficiently used for the feature concatenation. Our intensive experiments show that our proposed model generates more balanced stylized images in content and style than state-of-the-art methods. Moreover, our proposed network achieves efficiency in speed.

1. Introduction

How New York looks like in "The Starry Night" by Vincent van Gogh is an interesting question and, at the same time, difficult to answer. In practice, re-painting a famous fine-art style takes much time and requires well-trained artists. Answering this question can be stated as the problem of migrating semantic content of one image to different styles, and it is called style transfer.

Style transfer is long-standing and has fallen into image synthesis problem which is a fundamental research in computer vision. Style transfer has its origin from non-photo-realistic rendering [1] and is closely related to texture synthesis and transfer [2, 3]. Along with the impressive progress of various tasks in computer vision using deep neural networks, this topic has very recently been re-launched in both academy and industry. Gatys et al. [4] showed that the image representation derived from a Convolutional Neural Network (CNN) can be used to represent the semantic content of an image and the style, which opened up a new trend of CNN-based style transfer.

Figure 1: Example of stylized results. Left-most column: content image (large) and style image (small). From left to right: the stylized image by our method, Johnson et al. [5], Huang et al. [6], and Gatys et al. [4]. Our results surrounded with red rectangles are more balanced in content and style than the others.

CNN-based approaches in style transfer fall into two categories [7]: Descriptive methods based on Image Iteration (DII) and Generative methods based on Model Iteration (GMI). The key idea of DII is to synthesis a stylized image by directly updating pixels in the image iteratively through the back-propagation. The DII such as [4, 8, 9] starts with a noise image and iteratively update the image by changing the distribution of noise along with the statistics of content and style until the defined loss function is optimally minimized. GMI such as [5, 6, 10, 11], on the other hand, first optimizes a generative model through iter-

978-1-5386-4886-5/18/$31.00 ©2018 IEEE
DOI 10.1109/WACV.2018.00152

ations, and then renders the stylized image using a forward pass. In order to optimize the generative model, GMI trains each feed-forward model for each specific style by using the gradient descent over a large dataset. DII is known to produce better stylized results in quality than GMI [7], while GMI has more efficiency in speed.

Although existing methods [4, 5, 6, 8, 9, 10, 11] show the capability of rendering image contents in different styles, generated stylized images are not always well balanced in content and style. Such methods take care of either the content or the style, but not both, producing unbalanced stylized images. DII is good at faithfully rendering the style while it tends to lose the content. GMI, on the other hand, preserves more semantic content than the style. How to keep the balance between the content and the style in style transfer is a crucial issue to improve the quality of stylized images. Another important issue to address is the computational speed. Although GMI such as [5, 6, 10, 11] are able to produce stylized images fast, they rely on a strong computational power. Therefore, either DII or GMI is hard to apply to real-time applications.

We propose an end-to-end two-steam network for balancing the content and style in stylized images where contributions of the content and the style are adaptively taken into account. The encoder part of our network consists of the content stream and the style stream where the streams have different architectures. The two streams are independently trained to learn the semantic content or the style representation. The content features and the style features are then combined in our proposed adaptive concatenation layer to ensure the balanced contribution of each stream. As the decoder part of our network, we use the feed-forward model to reduce the rendering time while we spend much time on learning like [5, 6]. We employ the instance incremental learning strategy [12] in the network training, which allows our network not only to accommodate fast training but also to easily adapt new styles. Our experiments demonstrate that our method produces more balanced stylized images in both content and style than the state-of-the-art methods (Fig. 1). They also show that our method runs about 10× faster than the state-of-the-art methods. We remark that our proposed model is trained for one style only, but it is easy to be fine-tuned to other styles incrementally with a low cost.

2. Related work

Early work on style transfer was reported in the context of texture synthesis. Some methods there used histogram matching [13] and/or non-parametric sampling [2, 3]. These methods had limited results because they relied on handcrafted low-level features and often failed in capturing features in semantic levels from the content and the style.

Gatys et al. [4] for the first time proposed a method using CNNs and showed remarkable results. Their method trains CNNs to learn the semantic information from content images and matched it with the distribution of the style. It starts from a randomly distributed noise image and iteratively updates the image to produce an image satisfying the semantic distribution of the content image and appearance statistics of the style. During the iteration, the weighted sum of style loss and content loss is minimized. As follow-up work of Gatys et al. [4], Luan et al. [8] proposed a structure preservation method using Matting Laplacian for photo-realistic style transfer. Mechrez et al. [9] utilized the screened Poisson equation to make a stylized image more photo-realistic. These approaches fall into the DII category, and all face with the computational speed problem.

Johnson et al. [5], on the other hand, took GMI, proposing a feed-forward CNN and used the perceptual loss function for gradient-based optimization. The perceptual loss used there is similar to content and style loss in [4]. Their model has only to pass the content image to a single forward network to produce a stylized image, which is fast. Wang et al. [10] also utilized the feed-forward network, and they used multiple-generator to improve the quality of results. These methods are fast in generating stylized images, but they are capable of dealing with a single style only.

Dumoulin et al. [14] proposed a multi-style network that introduces shared-computation in many style images where they used instance normalization (IN) [15] for balancing features from the content and from the style. They also proposed an improvement of IN to learn a different set of affine parameters for multi-styles in the batch way. However, their model can train a limited number of styles because the network capability is limited, meaning that the number of styles to handle is limited. Chen et al. [11] proposed a method that overcomes the limitation of the number of styles by using a patch-based method. Their method first extracts a set of patches from the content and style each, and then, for each content patch, the method finds its closest style patch and swaps their activation. In this way, their method transfers an unlimited number of styles; however, the cost for patch extraction and swapping increases the computational time significantly.

Very recently, Huang et al. [6] proposed a new multi-style transfer model that consists of two CNN streams for content and style. They employed pre-trained VGG-16 to extract content and style features and introduced Adaptive Instance Normalization (AIN) that aligns the mean and the variance of content features in accordance with those of style features. Their method, however, used the same architecture for the content and for the style, causing unavoidable unbalance between the content and the style because semantic levels extracted from the content and the style should not be the same in style transfer. Furthermore, AIN assumes the standard distribution on pixel values of images, which is not always ensured in styles when normalizing data. AIN

requires an expensive computational cost as well.

Differently from the methods above, we take into account the contributions of the content and the style through a two-stream feed-forward network to balance the content and the style in stylized images. In particular, our proposed two-stream network is different from [6] in that our network has different depths in layer for the content and the style encoders to extract different semantic levels of the content and the style. In addition, our method employs the instance incremental learning strategy [12] to make our network easy to fine-tune to other styles with a cheap computational cost, enabling our method to deal with multi-styles. This strategy also helps our method to speed up rendering time.

3. Proposed method

3.1. Network design

Our model follows the encoder-decoder architecture for the end-to-end rendering of the content in a given style [5, 10, 11]. The network in [5, 10, 11] possesses only one encoder to extract the semantic content and style. This means that the extracted semantic level of the content and that of the style are the same. When we stylize images, the role of the content should be different from that of the style because the content gives us what exist (object shapes and locations) in the rendered image and the style gives us the impression of the rendered image. Accordingly, the semantic level used for the rending should be different depending on the content or the style. Otherwise, unbalance between the content and style remains in stylized images. We thus design a network having two encoders in which their architectures are different from each other to extract different semantic levels of the content and the style. With two encoders, our model treats the content and the style in different ways, allowing the network to be able to balance the roles of the content and the style better than the model having only one encoder.

Ideally, the network should be able to retain the semantics of the content as well as the statistics of the style as much as possible. The semantic content and style of an image are captured at different layers in the network [4]: the network obtains the style at low-level layers in depth while high-level layers become more sensitive to the actual content of the image. We thus design the encoders with different depths to retain useful information from both the content and the style. Namely, we design a deep encoder for the content and a shallow encoder for the style. Because the content feature and the style feature are extracted at different levels in the network, they have different characteristics. We thus introduce an effective concatenation to enhance the contribution of these features for good performances instead of implementing their simple concatenation.

3.2. Network architecture

Figure 2 illustrates our proposed two-stream network architecture, which consists of three Fully Convolutional Networks (FCNs): two encoders and one decoder. We develop a deep network, called the content subnet (the first encoder), to extract content feature ϕ_c from a content image, and a shallow network, called the style subnet (the second encoder), to extract style feature ϕ_s from a style image. These features are adaptively concatenated using the balance weight and then fed into a deep network, called the generator subnet, to produce a stylized image. We employ the VGG-16 model [16] as the loss network in the training phase.

In the training phase, content images and style images are resized to the size of 256×256 and then fed into the proposed network to produce stylized images with the same size. Although we train the network on images with the size of 256×256, the network can accept any size of images in testing. We remark that the size of the content image and that of the style image have to be the same.

3.2.1 Content subnet

The content subnet is a stack of six convolution layers with the filter size of 3×3, and the padding size of 1×1. We use the stride of 2×2 at the third, the fifth, and the sixth layers to reduce the size of feature maps and the stride of 1×1 at the other layers. The numbers of the output channels are 32, 48, 64, 80, 96, and 128, respectively. Each convolution layer is followed by a spatial instance normalization layer [15] (because of its effectiveness in style transfer [15]) and a Rectified Linear Unit (ReLU) layer [17]. In order to avoid the border artifacts caused by convolution, the reflection-padding is used instead of the zero-padding similarly to [14].

3.2.2 Style subnet

The style subnet, which has four convolution layers, is shallower than the content subnet. All convolution layers have the filter size of 3×3, the reflection-padding of 1×1, and the stride of 2×2, except for the first layer that employs the stride of 1×1. The numbers of the output channels are 32, 64, 96, and 128, respectively. Similarly to the content subnet, each convolution layer is also followed by a spatial instance normalization layer [15] and a ReLU layer [17].

Feature maps ϕ_c and ϕ_s extracted from the content subnet and the style subnet respectively are concatenated before being fed into the generator subnet. To take into account the contributions of ϕ_c and ϕ_s, we introduce the adaptive concatenation layer with the balance weight γ (cf. Section 3.4).

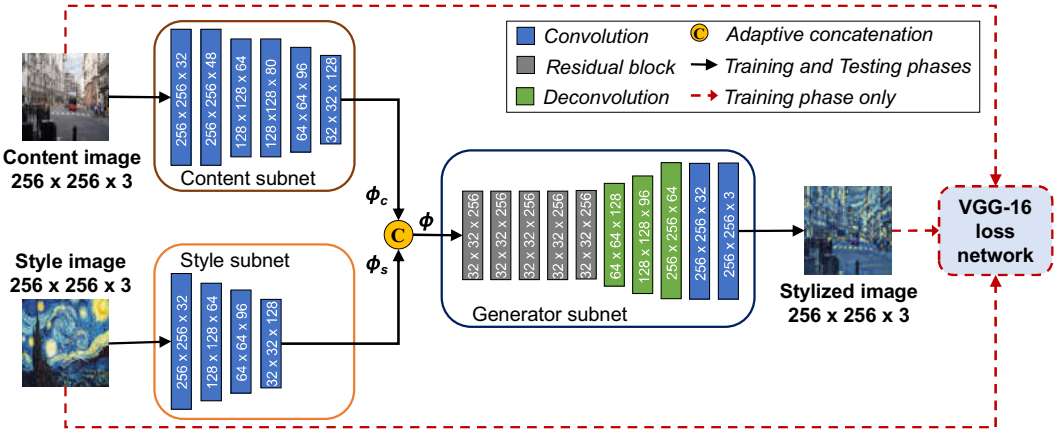


Figure 2: Framework of our proposed method. Our network consists of two encoders having different architectures and one decoder. The loss network is used to train the encoders and the decoder.

3.2.3 Generator subnet

The generator subnet consists of five residual blocks, three deconvolution layers, and two convolution layers in this order.

Johnson et al. [5] argues that the residual block can enrich the information involved in the input feature. We, therefore, use residual blocks to increase the impact of the balance weight γ in the concatenated feature. Similarly to [5], we use five residual blocks outputting 256 channels, where each of them has two convolution layers with the filter size of 3×3, the reflection-padding of 1×1, the stride of 1×1, and a summation layer as in [18]. All convolution layers are followed by a spatial instance normalization layer [15] (we use it to replace the batch normalization [19] in the original architecture [18]) and a ReLU layer [17].

To upscale the feature map, we employ three deconvolution layers with the same filter-size of 3×3, the reflection-padding of 1×1, and the stride of 2×2, outputting 128, 96, and 64 channels, respectively.

In order to eliminate the affect of the convolution stride, we use two convolution layers which have the filter-size of 1×1, the padding of 0×0, and the stride of 1×1, outputting 32 and 3 channels. All deconvolution layers and convolution layers are followed by a spatial instance normalization layer [15] and a ReLU layer [17], except for the last convolution layer that uses the $\tanh$ function to guarantee that the range of the output is $[0, 255]$.

3.3. Loss function

We employ two loss functions for content loss and style loss, which are computed from layers of the loss network. The content loss $\mathcal{L}_\mathrm{c}$ computes the similarity of high-level features between the content image and the stylized image. The style loss $\mathcal{L}_\mathrm{s}$, on the other hand, computes the similarity of low-level features between the style image and the stylized image.

The overall loss is a weighted sum of the content loss and the style loss:

$$\mathcal{L}(\hat{y}, y_\mathrm{c}, y_\mathrm{s}) = \alpha \mathcal{L}_\mathrm{c}(\hat{y}, y_\mathrm{c}) + (1 - \alpha)\mathcal{L}_\mathrm{s}(\hat{y}, y_\mathrm{s}), \quad (1)$$

where $y_\mathrm{c}, y_\mathrm{s}$, and $\hat{y}$ denote the content image, the style, and the stylized image, respectively. α is the combination weight (we set $\alpha = 0.5$ in our experiments to equally weight these two loss functions).

We obtain the content loss at M layers as follows:

$$\mathcal{L}_\mathrm{c}(\hat{y}, y_\mathrm{c}) = \frac{1}{M} \sum_{k \in M} \frac{1}{C_k \times H_k \times W_k} \|\Phi_k(\hat{y}) - \Phi_k(y_\mathrm{c})\|_2,$$

where $\Phi_k(\cdot)$ denotes the normalized feature map at the k-th layer, which has $C_k \times H_k \times W_k$ elements.

The style loss is computed at N layers as follows:

$$\mathcal{L}_\mathrm{s}(\hat{y}, y_\mathrm{s}) = \frac{1}{N} \sum_{k \in N} \|G(\Phi_k(\hat{y})) - G(\Phi_k(y_\mathrm{s}))\|_F,$$

where $\|\cdot\|_F$ denotes the Frobenius norm [20]. $G(\Phi_k(\cdot))$ is the Gram matrix [20] of the normalized feature map at the k-th layer. The Gram matrix $G_{C_k \times C_k}$ has elements $G_{ij} = \langle v_i, v_j \rangle$ where v_i, v_j are features at the i-th and the j-th channels respectively of the feature map $\Phi_k(\cdot)$.

3.4. Adaptive concatenation layer

In our network, features extracted from the content subnet and the style subnet are concatenated before being fed into the generator subnet. In order to weight the contributions of the content and the style in the final result, we introduce the balance weight using $\mathcal{L}_c(t)$ and $\mathcal{L}_s(t)$ at the t-th iteration as follows:

$$\gamma_t = \frac{\mathcal{L}_\mathrm{s}(t)}{\mathcal{L}_\mathrm{s}(t) + \mathcal{L}_\mathrm{c}(t)},$$

where γ_t is the balance weight at the t-th iteration in the training phase. To restrict the fluctuation of the balance weight, we compute γ at every T iterations and use it for the next T iterations:

$$\gamma = \frac{1}{T}\sum_{t=1}^{T}\gamma_t. \tag{2}$$

Using γ, we concatenate content features and style features in the adaptive concatenation layer as follows:

$$\phi = (\gamma \times \phi_{\mathrm{c}}) \oplus ((1-\gamma) \times \phi_{\mathrm{s}}),$$

where ϕ, ϕ_{c}, and ϕ_{s} denote the concatenated feature, the content feature, and the style feature, respectively. The learned balance weight γ ensures the balance of the contributions of the content feature and the style feature. For example, when $\mathcal{L}_{\mathrm{s}}$ is smaller than $\mathcal{L}_{\mathrm{c}}$ (meaning $\gamma \leq 0.5$ in Eq. (2)), the contribution of style feature is increased in the next iterations, and vice verse. Moreover, the learned balance weight γ is more advantageous than the fixed balance weight that does not concern the balance of losses.

4. Experiments

4.1. Experiment setup

We used in our experiments, images in the MS-COCO 2014 dataset [21] as our content images, and six famous paintings widely used in style transfer [4, 5, 6], as our style images (cf. Fig. 3).

We used the MS-COCO 2014 training set for our training, and we randomly selected 20 images from the MS-COCO 2014 validation set for our validation. In the testing phase, on the other hand, we randomly selected 50 images from MS-COCO 2014 validation (different ones from the 20 images used in our validation).

We compared our method with state-of-the-art methods: Gatys+[4], Johnson+[5], and Huang+[6]. We note that Gatys+[4] is based on DII and the others are on GMI. For [4], we used the re-implementation version by J. Johnson[1]. For the others, we used publicly available source codes with parameters recommended by the authors. We remark that we set 1000 iterations for Gatys+[4].

The proposed network was implemented in Torch [22]. Our method used the instance incremental learning strategy for dealing with multiple styles. We conducted all experiments using a PC with CPU core i7 3.7 GHz, 12 GB of RAM, and GTX 770 GPU (4 GB of VRAM).

We used the VGG-16 model [16] pre-trained on the ImageNet [23] as the loss network. All layers after $relu4\_3$ layer were dropped. We obtained the content loss at $M = 1$ layer, e.g., $relu3\_3$, and the style loss at $N = 4$ layers, e.g., $relu1\_2$, $relu2\_2$, $relu3\_3$, and $relu4\_3$ (M and N are defined in Section 3.3).

[1] https://github.com/jcjohnson/neural-style

Figure 3: Styles used in experiments. From left to right: Starry Night, Mosaic, Composition VII, La Muse, The Wave, and Feathers.

4.2. Evaluation metric

We introduce a metric to evaluate how the stylized image is balanced in content and style.

For each pair of content image c and style s, we compute content loss $\mathcal{L}_{\mathrm{c}}$ and style loss $\mathcal{L}_{\mathrm{s}}$. When $\mathcal{L}_{\mathrm{c}}$ and $\mathcal{L}_{\mathrm{s}}$ are almost same, we may say that the stylized image is well balanced in content and style. This means that in the 2D plane whose coordinate system is defined by content loss and style loss, how close $(\mathcal{L}_{\mathrm{c}}, \mathcal{L}_{\mathrm{s}})$ is to the line of "content loss"="style loss" (called the balanced axis hereafter) can be a criterion to evaluate how the stylized image is balanced in content and style. The distance between the origin and $(\mathcal{L}_{\mathrm{c}}, \mathcal{L}_{\mathrm{s}})$ is, of course, a criterion for evaluating the quality of the stylized image.

We assume that we have K stylized images. We normalize content loss and style loss for each stylized image over K images:

$$\widetilde{\mathcal{L}_{\mathrm{c}}} = \frac{1}{1+\exp\left(\frac{\mathcal{L}_{\mathrm{c}}-\overline{\mathcal{L}_{\mathrm{c}}}}{\sigma_{\mathrm{c}}}\right)}, \quad \widetilde{\mathcal{L}_{\mathrm{s}}} = \frac{1}{1+\exp\left(\frac{\mathcal{L}_{\mathrm{s}}-\overline{\mathcal{L}_{\mathrm{s}}}}{\sigma_{\mathrm{s}}}\right)},$$

where $\overline{\mathcal{L}_{\mathrm{c}}}$, σ_{c}, $\overline{\mathcal{L}_{\mathrm{s}}}$, and σ_{s} are the mean and the standard deviation of content loss and style loss over K stylized images, respectively.

Let ω ($\in [0, \frac{\pi}{4}]$) denote the angle between the line going through the origin and $(\widetilde{\mathcal{L}_{\mathrm{c}}}, \widetilde{\mathcal{L}_{\mathrm{s}}})$ and the content loss axis or the style loss axis (the smaller angle is selected):

$$\omega = \begin{cases} \tan^{-1}\frac{\widetilde{\mathcal{L}_{\mathrm{s}}}}{\widetilde{\mathcal{L}_{\mathrm{c}}}} & \text{if } \widetilde{\mathcal{L}_{\mathrm{c}}} \geq \widetilde{\mathcal{L}_{\mathrm{s}}} \\ \pi/2 - \tan^{-1}\frac{\widetilde{\mathcal{L}_{\mathrm{s}}}}{\widetilde{\mathcal{L}_{\mathrm{c}}}} & \text{otherwise} \end{cases}.$$

Lager ω indicates that $(\widetilde{\mathcal{L}_{\mathrm{c}}}, \widetilde{\mathcal{L}_{\mathrm{s}}})$ is closer to the balance axis, meaning that the stylized image is more balanced in content and style.

The quality of stylized images is evaluated using

$$length = \sqrt{\widetilde{\mathcal{L}_{\mathrm{c}}}^2 + \widetilde{\mathcal{L}_{\mathrm{s}}}^2}.$$

Using ω and $length$ above, we define our metric $balance$:

$$balance = \frac{\tan(\omega)}{length}.$$

$balance$ concerns both the criterion for balance and the criterion for the quality. Therefore it is a useful metric for

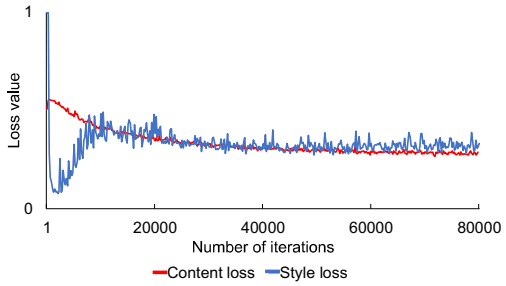


Figure 4: Content loss and style loss in validation set on the Starry Night style.

evaluating stylized images. We note that larger $balance$ is better because $\tan(\omega)$ should be larger and $length$ should be smaller for better stylized images.

4.3. Training the model

Our method addresses a one-style model to reduce computational time. To deal with multiple styles, we follow instance incremental learning when training a new yet unknown style. For the new style, we fine-tune parameters from an existing model. With this learning strategy, our method can easily adapt a new style with a lower cost than existing work [4, 5, 6, 10]. Moreover, instance incremental learning enables our method to cope with an unlimited number of styles fast unlike existing methods such as [11, 14].

We first trained an initial model on the Starry Night style and then incrementally fine-tuned on the other styles one by one. We trained the network on the Starry Night style with a batch size of 2 for 80k iterations corresponding to 2 epochs. The balance weight γ in Eq. (2) is re-computed at every $T = 500$ iterations. All the training and validation images are resized to 256×256. To train the model, we used the Adam optimizer [24] with the learning rate of 10^{-3}, the moments $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and the division from zero parameter $\epsilon = 10^{-8}$. We did not use the learning rate decay and the weight decay.

For the initial model, we trained all subnets simultaneously with independently updating the weight of each subnet. Validation was performed at every 100 iterations during the training process. When observing the content loss and the style loss on the validation set, if any loss function raises the overfitting problem, we stopped updating the weight of the corresponding subnet. Fig. 4 illustrates our validation loss on the Starry Night style (we see content loss and style loss converge to almost the same value). We observed overfitting on style loss after one epoch (about 40k iterations). The training of the style subnet was thus stopped, while the weights of the other subnets were kept updated continuously.

We incrementally fine-tuned the initial model to the other styles one by one. 2000 images in the MS-COCO 2014 training set [21] were randomly selected as content images for training. The network was trained for 1000 iterations with the batch size of 2. The Adam optimizer [24] was also used with the same parameters as the training of the initial model. The balance weight γ in Eq. (2) was re-computed at every $T = 50$ iterations. The loss-based training technique was also applied to avoid overfitting, where the validation was performed at every 50 iterations.

Table 1: Average of rankings. The two best results are given in **red** and **blue**, respectively.

Style	Ours	Johnson+[5]	Huang+[6]	Gatys+[4]
Starry Night	**2.51**	2.88	3.11	**1.50**
Mosaic	**2.25**	2.39	3.58	**1.79**
Composition VII	2.72	2.95	**2.38**	**1.95**
La Muse	**2.23**	3.51	2.44	**1.82**
The Wave	**1.90**	3.00	2.72	**2.38**
Feathers	**2.45**	**1.65**	3.12	2.78
All together	**2.34**	2.73	2.89	**2.04**

4.4. Qualitative evaluation

Figure 5 shows examples of the obtained results, showing that the stylized images obtained by our method are more balanced in content and style. We also see that overall the results obtained by Gatys+[4] reflect the style well, but they mostly lose content (we cannot understand the content of some results). In some styles (Starry Night, Composition VII, and The Wave), we see that Johnson+[5] seems to randomly select a patch in the style and paste it into the content image. Huang+[6] also loses the content and suffers from a so-called checkerboard effect.

To objectively compare the obtained results, we conducted a user study. Namely, we presented 300 sets of images to 23 subjects where each set consists of the content image, the style, and four output images obtained by our method and three comparison methods [4, 5, 6]. We then asked the subjects to rank the four output images at each set (1st is best, and 4th is worst). We note that four output images are aligned in the random order in each set and that each set was displayed for 6 seconds. We also remark that the combination of 50 content images and 6 styles results in 300 stylized images by each method.

Table 1 shows the average of rankings over 300 sets. We see that our method takes the second best ranking among the four methods. As DII is known to perform better in stylized quality than GMI [7], Gatys+[4], which is DII, takes the best ranking. Among the others (GMIs), our method performed best. We remark that the single-style models (ours, Gatys+[4], and Johnson+[5]) performed better than the multi-style model (Huang+[6]).

We also computed the average of rankings in each style, which is also illustrated in Table 1. Like the overall average, our method is ranked in the second place except for the

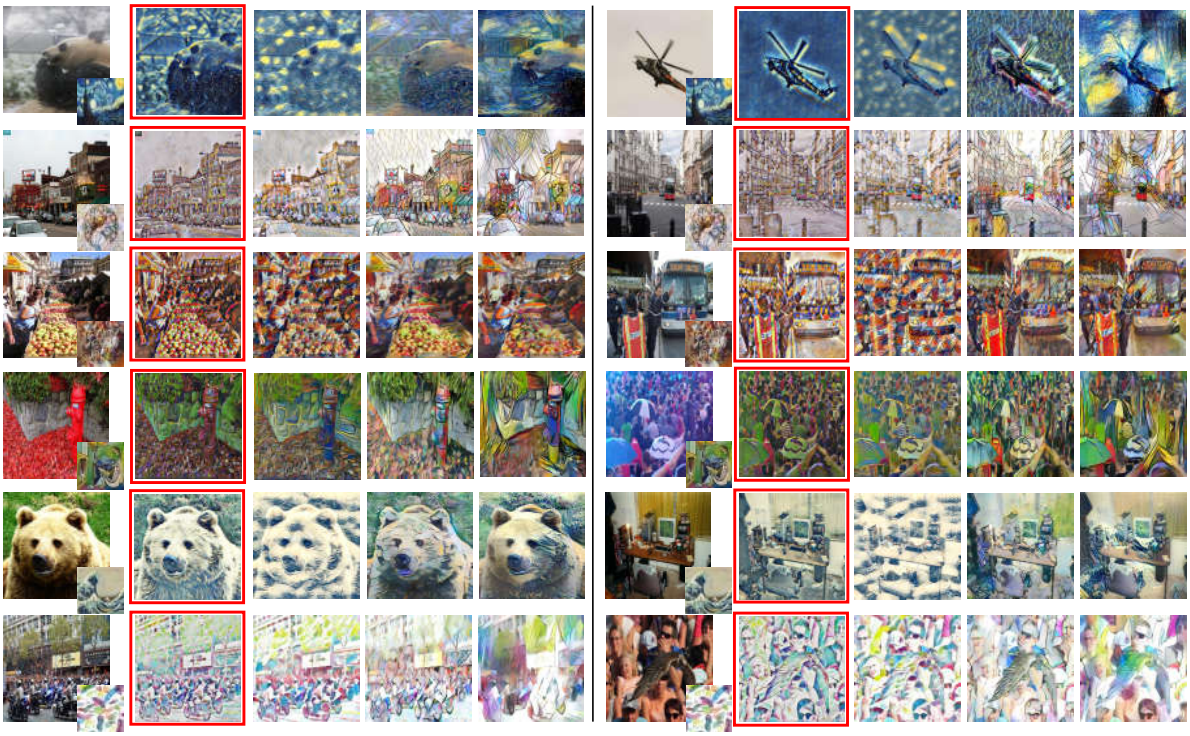
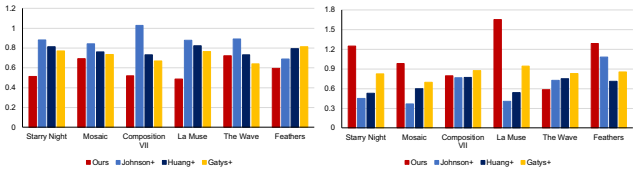


Figure 5: Visual comparison of our method against the state-of-the-art methods. In each block, from left to right, a content image (large one) with a style (small one) is followed by outputs by our method, Johnson+[5], Huang+[6], and Gatys+[4]. Our results surrounded with red rectangles are more balanced in content and style than the others. Note that all stylized images are with the size of 512×512.

Table 2: Averages of *length* (smaller is better) and *balance* (larger is better).

	Ours	**Johnson+**[5]	**Huang+**[6]	**Gatys+**[4]
length	**0.59**	0.87	0.78	0.73
balance	**1.09**	0.63	0.65	0.84



(a) *length*. (b) *balance*.

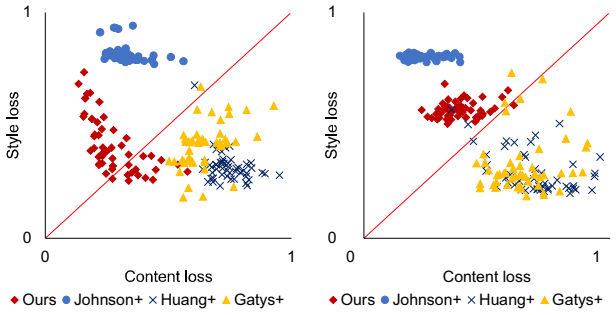
Figure 6: Averages of *length* and *balance* in each style.

Composition VII style and the Wave style. This indicates that our method performs stably independent of styles. We note that the Composition VII style and the Wave style are rather complex (Fig. 3) and, the results for these styles are difficult to evaluate. Indeed, we received the feedback from many subjects that ranking the stylized images for these styles was pretty difficult.

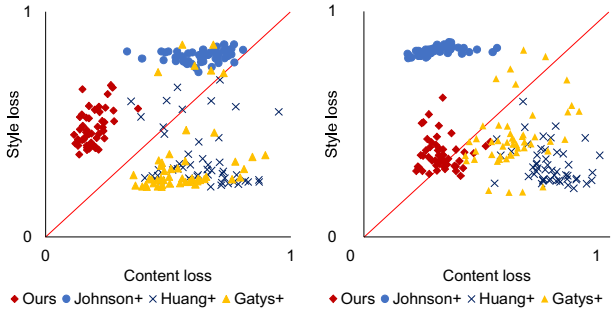
4.5. Quantitative evaluation

In order to quantitatively evaluate the obtained results, we computed the averages of *length*'s and *balance*'s over 300 sets for each method (Table 2). We see that our method performs best both in *length* and *balance*. We also computed the averages of *length*'s and *balance*'s in each style, which is illustrated in Fig. 6. Fig. 6 shows that (1) our method performs best in *length* for all the styles except for the Wave style, and that (2) our method performs best in *balance* for all the style images except for the Composition VII style and the Wave style. The fact that the performances of our method degrade for these particular styles is in good harmony with our qualitative evaluation results. This is because as mentioned above, the stylized images using the Composition VII style or the Wave style are difficult to evaluate according to subjects' impression.

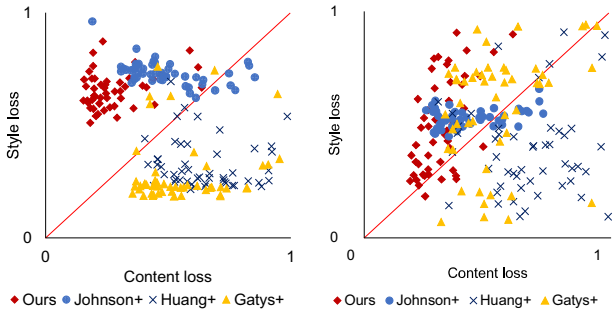
To look into the results in more detail using these styles (the Composition VII and the Wave), we show the loss distribution of 50 stylized images in each style (Fig. 7). In the case of the Composition VII style (Fig. 7c), we see that our method preserves much more content than the style because the loss distribution appears far above the balanced axis. This observation also holds true for the Wave style



(a) Starry Night style. (b) Mosaic style.



(c) Composition VII style. (d) La Muse style.



(e) The Wave style. (f) Feathers style.

Figure 7: Loss distribution in each style. Red lines denote the balanced axis. The distributions by our method exist nearer the balanced axis than those of the other methods.

(Fig. 7e). We see that content loss in our method is smallest among the four methods while style loss is not the case. We can also see in Fig. 7e that our method successfully reduces content loss while keeping style loss similarly to that of Johnson+[5], indicating that our method outperforms Johnson+[5]. We remark that (1) the content loss in our method is smallest for all the styles and that (2) loss distributions in our method appear (densely) near the balanced axis for the other styles while those in the other methods do not (for example, Figs. 7b,7d).

4.6. Computational speed

We measured the running time for generating 300 stylized images with the sizes of 256×256 and 512×512 by each method and compared the average for generating one stylized image by each method.

Table 3: The average wall-clock time for producing one stylized image.

Image size	Time (seconds)			
	Ours	**Johnson+**[5]	**Huang+**[6]	**Gatys+**[4]
256×256	**0.12**	1.12	1.98	74.12
512×512	**0.46**	3.79	6.78	269.6

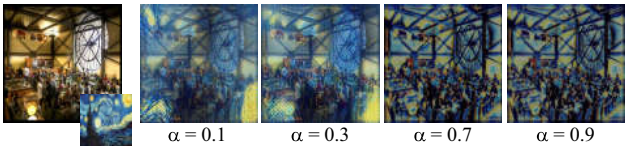


Figure 8: Example of stylized images by changing α from 0.1 to 0.9.

Table 3 illustrates the average of the running time in generating one stylized image. As we see, our method is the fastest and speeds up 10 times for the image size of 256×256 and 8 times for that of 512×512 when compared with the second fastest one (Johnson+[5]). We can thus conclude that our method is promising for real-time applications.

5. Conclusion

We proposed an end-to-end two-stream network for balancing the content and style in stylized images. Our proposed method utilizes a deep FCN to preserve the semantic content and a shallow FCN to faithfully learn the style representation, whose outputs are adaptively concatenated using the balance weight and fed into the decoder to generate stylized images. Our intensive experiments demonstrate the effectiveness of our proposed method against state-of-the-art methods in term of balancing content and style. Furthermore, our proposed method outperforms the state-of-the-art methods in speed.

Finally, we mention the role of α in Eq. (1). Since γ in Eq. (2) is learned through the content and the style losses so that the two losses become well balanced, we need not adaptively change α anymore to balance the two losses. This is the reason why we set $\alpha = 0.5$. In some sense, α plays the role of the indicator for whether content or style is emphasized in obtained stylized images. If we choose smaller α, style is more emphasized and results become more similar to those by Gatys+[4] than the case of $\alpha = 0.5$. For larger α, content is more emphasized and results become more similar to those by Johnson+[5] (see Fig. 8). Investigating this role in more detail is left for future work.

Acknowledgment: This work is in part supported by JST CREST (Grant No. JPMJCR14D1).

References

[1] J. E. Kyprianidis, J. Collomosse, T. Wang, and T. Isenberg, "State of the "art": A taxonomy of artistic stylization techniques for images and video," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 5, pp. 866–885, May 2013. 1

[2] M. Ashikhmin, "Synthesizing natural textures," in *Symposium on Interactive 3D Graphics*, 2001, pp. 217–226. 1, 2

[3] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *SIGGRAPH*, 2001, pp. 341–346. 1, 2

[4] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *CVPR*, June 2016, pp. 2414–2423. 1, 2, 3, 5, 6, 7, 8

[5] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *ECCV*, 2016. 1, 2, 3, 4, 5, 6, 7, 8

[6] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *ICCV*, 2017. 1, 2, 3, 5, 6, 7, 8

[7] Y. Jing, Y. Yang, Z. Feng, J. Ye, and M. Song, "Neural style transfer: A review," *CoRR*, vol. abs/1705.04058, 2017. 1, 2, 6

[8] F. Luan, S. Paris, E. Shechtman, and K. Bala, "Deep photo style transfer," in *CVPR*, July 2017. 1, 2

[9] R. Mechrez, E. Shechtman, and L. Zelnik-Manor, "Photorealistic style transfer with screened poisson equation," in *BMVC*, 2017. 1, 2

[10] X. Wang, G. Oxholm, D. Zhang, and Y.-F. Wang, "Multimodal transfer: A hierarchical deep convolutional neural network for fast artistic style transfer," in *CVPR*, July 2017. 1, 2, 3, 6

[11] T. Q. Chen and M. Schmidt, "Fast patch-based style transfer of arbitrary style," in *NIPS*, 2016. 1, 2, 3, 6

[12] J. Read, A. Bifet, B. Pfahringer, and G. Holmes, "Batch-incremental versus instance-incremental learning in dynamic and evolving data," in *International Symposium on Advances in Intelligent Data Analysis*, 2012, pp. 313–323. 2, 3

[13] D. J. Heeger and J. R. Bergen, "Pyramid-based texture analysis/synthesis," in *SIGGRAPH*, 1995, pp. 229–238. 2

[14] V. Dumoulin, J. Shlens, and M. Kudlur, "A learned representation for artistic style," *ICLR*, 2017. 2, 3, 6

[15] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," in *ICML*, 2016. 2, 3, 4

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015. 3, 5

[17] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010, pp. 807–814. 3, 4

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016. 4

[19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015, pp. 448–456. 4

[20] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. New York, NY, USA: Cambridge University Press, 2012. 4

[21] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollr, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*, Zurich, 2014. 5, 6

[22] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *NIPS Workshop on BigLearn*, 2011. 5

[23] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015. 5

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015. 6