Decoder Network over Lightweight Reconstructed Feature for Fast Semantic Style Transfer

Ming Lu *[1], Hao Zhao[1], Anbang Yao[2], Feng Xu[3], Yurong Chen[2], and Li Zhang[1]

[1]Department of Electronic Engineering, Tsinghua University
[2]Cognitive Computing Laboratory, Intel Labs China
[3]School of Software, Tsinghua University
{lu-m13@mails,zhao-h13@mails,feng-xu@mail,chinazhangli@mail}.tsinghua.edu.cn
{anbang.yao, yurong.chen}@intel.com

Abstract

Recently, the community of style transfer is trying to incorporate semantic information into traditional system. This practice achieves better perceptual results by transferring the style between semantically-corresponding regions. Yet, few efforts are invested to address the computation bottleneck of back-propagation. In this paper, we propose a new framework for fast semantic style transfer. Our method decomposes the semantic style transfer problem into feature reconstruction part and feature decoder part. The reconstruction part tactfully solves the optimization problem of content loss and style loss in feature space by particularly reconstructed feature. This significantly reduces the computation of propagating the loss through the whole network. The decoder part transforms the reconstructed feature into the stylized image. Through a careful bridging of the two modules, the proposed approach not only achieves competitive results as backward optimization methods but also is about two orders of magnitude faster.

1. Introduction

Style transfer with Convolution Neural Network (CNN) is becoming prevalent since the advent of the seminal work [9]. It aims to generate a stylized image I_o which preserves the content of image I_c and the style of image I_s. Both content and style are represented in the feature space of a deep CNN f. The objective of neural style transfer is originally formulated as the minimization of the content loss and style loss:

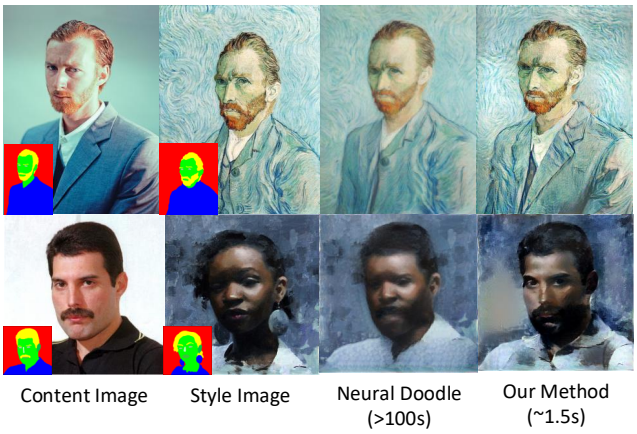

Figure 1. Our method can be applied to portrait style transfer. Compared with backward optimization method [1], our method can achieve competitive results but is about two orders faster.

$$I_o = \arg\min_{I_o} L_{content}(f(I_o), f(I_c)) + \lambda L_{style}(f(I_o), f(I_s)) \quad (1)$$

The content loss is usually defined as the distance between CNN features. Comparatively, the community makes great efforts to explore the representation of image style in order to define the style loss. The image style can be represented as the gram matrix [9] of CNN feature. Then the style loss can be naturally formulated by matching the gram matrices between the stylized image and style counterpart. Other methods [18, 19] based on neural patch characterize the style loss as neural patch matching. These methods are more suitable for the realistic style transfer than artistic style transfer since local patch preserves more structural information. The optimization of Eq. 1 can be solved by backward

*This work was done when Ming Lu was an intern at Intel Labs China supervised by Anbang Yao who is responsible for correspondence.

propagation.

However, these methods ignore meaningful spatial control over the transfer process, resulting in unsatisfying results. Recently, the community resorts to introduce semantic information into the system. Some methods [1, 10] are proposed to solve this semantic style transfer problem which are still based on time-consuming backward propagation. In [10], the authors present a possible solution to the computation bottleneck by extending their spatial control to fast neural style transfer [15]. Nevertheless, it is necessary to enforce further factorisation during the network training as discussed by the authors.

In this paper, we propose a fast semantic style transfer method. Our motivation is to re-formulate the optimization of $L = L_{content} + \lambda L_{style}$ as below.

$$\frac{dL(f(I_o))}{dI_o} = \frac{dL(f(I_o))}{df(I_o)} \frac{df(I_o)}{dI_o} \tag{2}$$

The first part $\frac{dL(f(I_o))}{df(I_o)}$ is self-contained within the feature map. The second part $\frac{df(I_o)}{dI_o}$ requires iterative back-propagation through the deep neural network which is computationally intensive. We try to shed some new insights on these two parts in our framework. Specifically, we solve the first part by our lightweight feature reconstruction method. Regarding the second part, we train an decoder network as a substitution which can reduce most of the computation.

The appealing merit of our method is that we solve the semantic style transfer problem in feature space instead of image space [9]. This avoids propagating the loss through a deep neural network which demands heavy computation. Extensive experiments demonstrate that our method can obtain comparable results as backward optimization methods. What's more, it is about two orders of magnitude faster.

Our contributions can be concluded as follows:

1. We present a lightweight feature reconstruction method to approximately solve the optimization of the content loss and style loss in feature space.

2. We train an decoder network to transform the reconstructed feature into the stylized image.

3. We propose a fast and compelling framework for fast semantic style transfer by associating the above modules.

2. Related Work

Backward Style Transfer Traditional style transfer methods [13, 26] formulate the problem as matching the local statistics of style image and content image. Based on the recent development of deep neural network [28, 27], [7, 9] propose a new framework for both texture synthesis and style transfer. Their methods consider the problem as the optimization of content loss and style loss. The style loss can be expressed as matching the gram matrix or local patch [18]. The patch based method [18] is more suitable for realistic image transfer. Later, [10, 8] propose to control the perceptual factors during style transfer. Other methods focus on improving the results by incorporating additional loss [32, 30] or giving an explanation of the gram matrix for style transfer [20]. Besides, some methods extend the applications of style transfer to video [23] or painting style transfer for portrait image [24]. All the above methods solve the optimization by backward optimization which is intrinsically time-consuming.

Fast Style Transfer In order to improve the speed of backward style transfer, some methods [15, 19] propose to train a feed-forward generation network to approximate the optimization process. [15] uses perceptual loss defined over deep convolution layers to train a transfer net. Regarding texture synthesis and style transfer, a pre-trained texture synthesis network can be applied to stylize a content image. [19, 29, 31] train the texture network based on image or neural patch. [2] presents a new method based on patch swap to express style transfer in feature space. First, the content and style images are forwarded through deep neural network to extract features. Then the style transfer is formulated as neural patch swap to get a reconstructed feature map. This feature map is finally decoded by an inverse network to image space. Methods like [6] propose to learn a network which can combine several styles with a single network. The above methods all ignore the semantic information for better results. Recently, Generative Adversarial Net (GAN) [11, 22, 14] has drawn a lot of attentions for image generation. However, applying GAN to style transfer is not a trivial problem because the content and style images are variant.

Semantic Style Transfer Unlike global style transfer, transferring the style between the corresponding regions of the style and content images usually shows better perceptual results than global style transfer [1]. In [1], the authors extend the patch based method [18] to incorporate the semantic masks into neural patch swap. This method is a straightforward extension to semantic style transfer. Combining semantic style transfer with modern segmentation methods [25] is partially explored in portrait style transfer by [24].

3. Proposed Method

The pipeline of our method is summarized as Figure 2. We first forward the content image I_c and style image I_s through a VGG network [27] to get respective features F_c and F_s. We simultaneously forward the semantic net to retrieve the corresponding masks in each layer. For K semantic regions, we denote the features in the k-th region as F_c^k and F_s^k. Our feature reconstruction aims to optimize the target feature F_o^k which minimizes the content loss and style loss [9] in individual regions. The features regarding K regions are collected to reconstruct the stylized feature

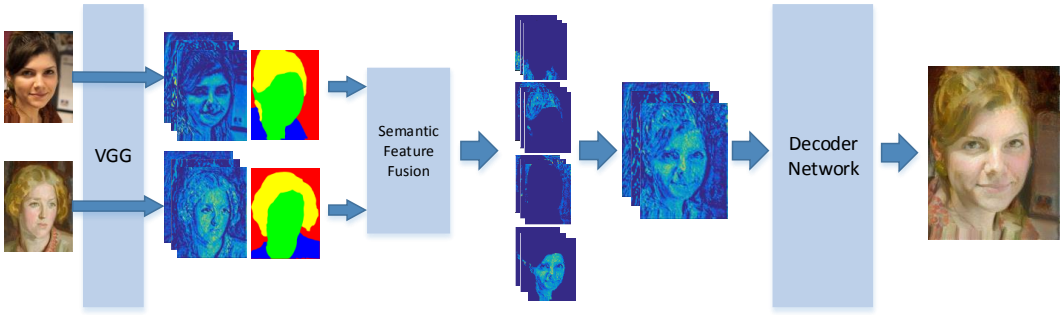

Figure 2. The pipeline of our method. The style and content images are first forwarded through a feature extraction network. The semantic segmentation masks are also forwarded and overlayed on the feature maps. Then our method reconstructs the feature map within each region. This feature map approximately solves the optimization [9] in a single layer. The feature map is finally decoded into a stylized image.

F_o. This reconstructed feature will be decoded into the final stylized image by a decoder network.

3.1. Lightweight Feature Reconstruction

We first describe our lightweight feature reconstruction within a semantic region in a certain feature layer. For the k-th region, we denote the content feature and style feature as $F_c^k \in R^{C\times N_k}$ and $F_s^k \in R^{C\times M_k}$, where, C is the number of channels in a certain layer, N_k and M_k are the number of activations in the k-th region of content image and style image respectively. Now semantic style transfer can be defined as optimizing the loss:

$$L_k(F_o^k) = L_{content}(F_o^k, F_c^k) + \lambda L_{style}(F_o^k, F_s^k) \quad (3)$$

Similar to the formulation of [9], the content loss is defined as the square Euclidean distance of F_o^k and F_c^k. The style loss is defined as the square distance between the gram matrices of F_o^k and F_s^k. The goal of semantic style transfer is to obtain the stylized image I_o by minimizing the above loss. Optimization by back-propagating the gradient to image space I_o is adopted to minimize Eq. 3. As described in [20], matching the Gram matrices of features is equivalent to minimizing the Maximum Mean Discrepancy (MMD) with the second order polynomial kernel. [20] achieves appealing results with several other kernels, validating their claims. Inspired by their demonstration, we develop our reconstruction method based on the linear approximation. Here we provide the formulation under adopting linear kernel $k(x, y) = x^T y$ instead of $k(x, y) = (x^T y)^2$. The new style and content loss can be formulated as follows.

$$\begin{aligned} L_{content} &= \tfrac{1}{2}\sum_{i=1}^{C}\sum_{k=1}^{N}(F_o^{ik} - F_c^{ik})^2 \\ L_{style} &= \frac{\lambda}{4C^2}\frac{1}{N^2}\sum_{k_1=1}^{N}\sum_{k_2=1}^{N}(\sum_{i=1}^{C}F_o^{ik_1}F_o^{ik_2}) \\ &-\frac{\lambda}{4C^2}\frac{2}{MN}\sum_{k_1=1}^{N}\sum_{k_2=1}^{M}(\sum_{i=1}^{C}F_o^{ik_1}F_s^{ik_2}) + C_s \end{aligned} \quad (4)$$

C_s denotes the self-product term of style feature which is a constant. The above formulation is within a single region so we omit the sub-script k. The goal of iterative optimization is to find the optimal F_o minimizing Eq. 4. We deduce the derivative of content loss and style loss respectively. The local minimum can be obtained by solving $\frac{dL}{dF_o^{ik}} = 0$. In this way, we obtain a rule to update the optimal feature F_o^{ik}.

$$F_o^{ik} = F_c^{ik} + \frac{\lambda}{2MNC^2}(\sum_{k=1}^{M}F_s^{ik}) - \frac{\lambda}{4NC^2}(\sum_{k=1}^{N}F_o^{ik}) \quad (5)$$

Similar to the iterative optimization method [9], we can update F_o^{ik} initialized by the feature of content image F_c^{ik}. Although our formulation still minimizes the style loss and content loss, different from optimizing in image space, we optimize the loss directly in feature space. This is important for improving computation performance since propagating the loss through a deep convolution neural network is much slower than optimizing the losses within a single layer. We name our formulation as a lightweight feature fusion to reconstruct the target feature F_o^{ik}. Furthermore, for semantic style transfer, this lightweight feature fusion shall reconstruct feature within corresponding regions. Note that our formulation is directly based on a single layer to characterize the content loss and style loss. A discussion on multiple layers will be given in experimental section.

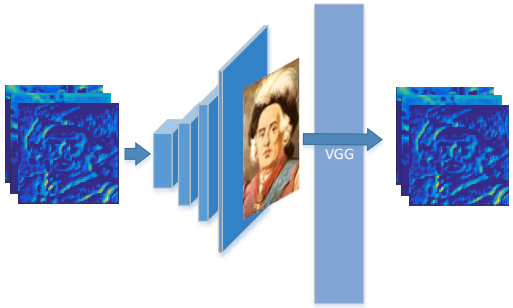

Figure 3. The pipeline to train our decoder network. Our decoder network is trained in an auto-encoder manner. We mix the real images' features and artistic images' features in order to train the decoder network. The extracted and mixed features are used to train our inverse net. In this manner, the network is trained to decode the features into images.

3.2. Neural Feature Decoder

Now, we detail the design and training of our decoder network. Since the ground-truth stylized results are not available, we train the decoder network in an auto-encoder manner. The pipeline is shown in Figure 3.

We collect the content images and the style images separately from COCO [21] and PainterByNumber [5] hosted by Kaggle. There are about 80000 natural content images and 80000 artistic style images. In order to get the fused features of content images and style images, we randomly select pairs of content and style images. Pairs of content and style images are passed through the VGG network [27] to obtain the features F_c and F_s. The stylized images shall keep the appearance of the content images while incorporating local statistics of the style images. We adopt the patch-based method [18] to get the mixed feature F_o of F_c and F_s. As shown in Figure 4, [18] reconstructs the content neural patch with the nearest neighbour of style neural patch. In this manner, the reconstructed feature keeps the appearance of the content image while containing the local statistics of the style image. By this way, we get the input feature samples to train the decoder network. We visualize some of these samples in Figure 5. Although our feature reconstruction method can be used to generate training samples, here we aims to efficiently generate a variety of feature maps for training the decoder. Neural patch swap is faster than our method which makes it more suitable for generating training samples.

We collect the training samples F_o (Figure 3) in a certain layer as mentioned above. Here the decoder network is denoted as $g(F_o;\theta)$ and the network to extract CNN feature is denoted as f. We use VGG [27] as f. Now the optimum of our decoder network can be defined as:

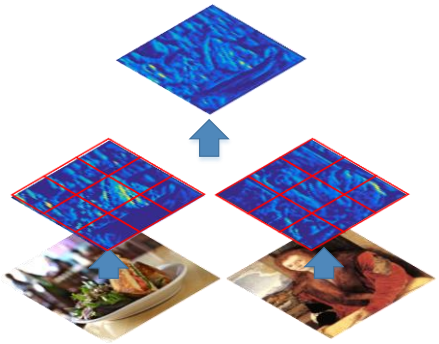

Figure 4. Illustration of sample generation. We use patch based method [18] to mix the real images with artistic images. In this manner, the reconstructed feature map keeps the appearance of the content image while the local distribution is similar as the style image. This is consistent with our goal, training a decoder network to inverse the fused feature.

Figure 5. Some training samples. The top row is the style images. The second row is the content images. Apart from using the original images, we use mixed feature maps based on neural patch swap [18] as shown in the last two rows. In this manner, our inverse net is trained on samples combining real and artistic images.

$$\theta = \arg\min_{\theta} \frac{1}{N}\sum_{i=1}^{N} \left\| f\left(g(F_o^i;\theta)\right) - F_o^i \right\|_F^2 + \lambda L_{TV}\left(g(F_o^i;\theta)\right) \tag{6}$$

θ is the parameter of the decoder network while N is the number of generated samples. We regularize the output image by a total variation loss L_{TV}. The network is trained in an auto-encoder manner. The loss enforces that the feature of decoded image is as same as possible to that of the input feature. This decoder network training is similar with the training of transfer net as in [15]. However, the purpose of our network is to inverse the neural feature into an image.

Instead, [15] aims to train an image to image translation network.

4. Experimental Details

In the feature fusion process, the iteration is repeated by N times. We fix N = 500 as our default setup and the average reconstruction process takes about 1 second.

The neural decoder network is trained on COCO dataset [21] and PainterByNumber from Kaggle [5]. The training data we use consists of 80000 content images from COCO and 80000 artistic images from PainterByNumber. We resize each pair of content and style images to 256×256 and train our inverse networks with a batch size of 2 for 80000 iterations, giving roughly two epochs over the training data as [15]. We use Adam [16] with a learning rate of 1e-3. The output images are regularized with total variation loss with a strength of 1e-6. We train our decoder networks separately at layer $relu3\_1$, $relu4\_1$ and $relu5\_1$ of the VGG network. Our implementation uses Torch [4] and cuDNN [3]. The training takes roughly two days on a single GTX Titan X GPU.

5. Results

5.1. Qualitative Analysis

Our Method versus Backward Optimization: Recall that we provide illustrative results in Figure 1 to compare the performance of our method and neural doodle [1]. Here, we further compare our method against the latest method Gatys2016 [10]. Both [1] and [10] solve the style loss and content loss by back-propagation. Regarding our results, we define the style loss and content loss both in $relu3\_1$. The reconstructed feature is updated with 500 iterations. As for Gatys2016, we use the publicly available implementation of [10]. As shown in the last two columns of Figure 6, our method achieves competitive results compared with [10].

Although our feature reconstruction is lightweight, it is still iterative. We compare the iteration number with the transfer results in Figure 7. As optimization in image space, optimization in feature space will also obtain a converged stylized result. In feature space, the reconstructed feature map maintains content image's appearance. The local statistics of the content image iteratively is transformed into the statistics of the style image. Both backward optimization and our method converge in the local minimum within a few iterations, however, iteration within a single feature layer is obviously faster than iteration over the network. As shown in Figure 7, after about 500 iterations, both the stylized image and feature map converge. This is the reason why we set 500 iterations as a default setup.

Our Method in Different Layers: In this experiment, we perform a set of ablative experiments to analyze the effect of different layers to our method. In figure 8, we show

Figure 6. We show the results of our method compared with Gatys2016 [10]. Although our method is the linear approximation of gram matrix, the decoded stylized images are perceptually similar as time-consuming backward optimization [10].

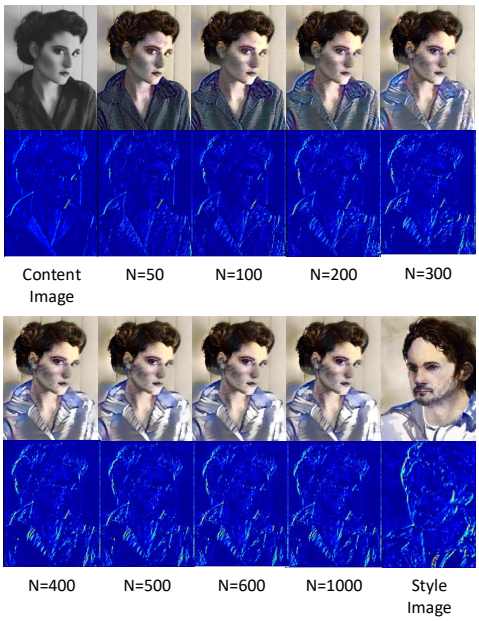

Figure 7. We show the transformation of both the feature map and the output image using our method. The stylized image will converge in a hundred iterations by our feature reconstruction. The result feature keeps the content image's appearance and incorporates the local statistics of style image.

our method in $relu3\_1$ and $relu4\_1$. As we can see, $relu3\_1$ and $relu4\_1$ obtain similar results, which proves the generalization ability of our framework to different layers. However, compared with $relu3\_1$, the results of $relu4\_1$ are darker. Although the texture is similar, the color distribution differs from the original content image. This might because the deeper layer's feature consists more abstract in-

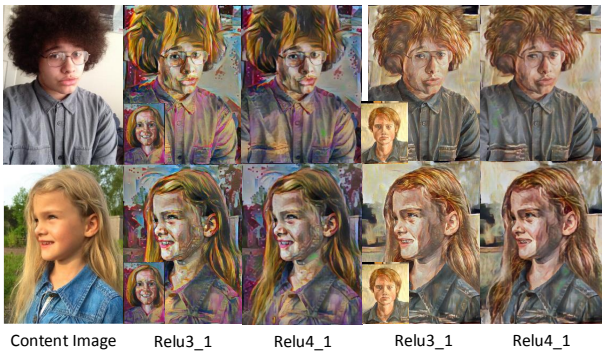

Figure 8. Our method in different layers. We present the results in $Relu3\_1$ and $Relu4\_1$. The texture is successfully transferred in both layers.

formation which fails to maintain the local color distribution. Besides, the deeper the layer, the larger receptive field each activation can cover.

Our Method versus Fast-Gatys2016: The most related work on fast semantic style transfer is the potential solution mentioned by [10]. In [10], the author proposes an approach to train a transfer net with additional guide channels. Here, we first compare our method with their paper results. This approach trains the transfer net to learn the correspondence between semantic style transfer and guide masks. Our results in Figure 9 are based on $relu3\_1$. As can be seen, our method achieves similar results compared with [10]. However, when we adopt their approach to transfer the styles of different regions in one style image, the transfer net doesn't achieve visually pleasing results. This might because transferring styles from different regions of a single image is different from combining two style images. The training of transfer net needs more regularization to enforce the connection between guide channels and styles. In sharp contrast to their strategy, our method still obtains satisfying results. All the results of [10] are from the open-source code. We train the transfer net with default settings.

We provide more results of our method for portrait style transfer in Figure 13. The purpose of our additional results is to illustrate the generalization ability of our method. The portrait images are from [25], we manually label the semantic segmentations of some images. Furthermore, we also label some head portrait paintings as our style images.

Single Layer versus Multi-Layer: In this part, we will show the comparison results of defining the style loss in a single layer versus multi-layer. [9] defines the content loss in a CNN layer and formulates the style loss in multi-layer. In contrast, our method defines both losses in a single layer. In what follows, we will show the comparison of defining style loss in one layer versus multi-layer. As shown in Figure 10, single layer causes some texture loss in the stylized result compared with multi-layer. The overall appearance

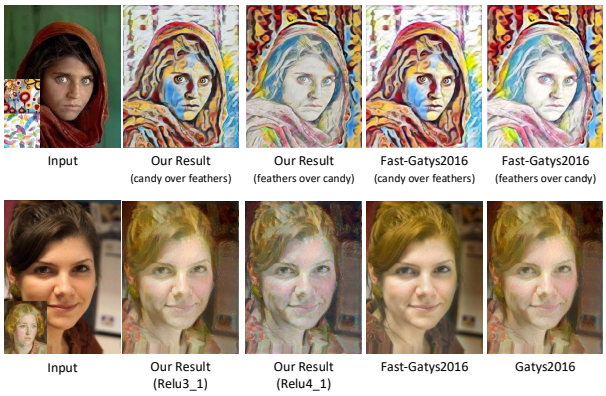

Figure 9. The comparison of our method with Fast-Gatys [10]. The first row shows our results against [10]'s results in combining two style images. Both methods obtain satisfying results. The second row is the results when we try to train a guided transfer net of semantic style transfer. Our results still achieve consistently pleasing results.

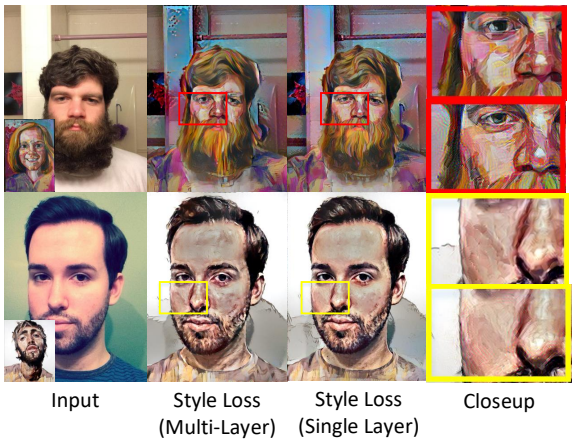

Figure 10. We show the comparison of single layer with multi-layer. The overall artistic effect is sufficiently similar. However, optimization on multi-layer transfers more texture as showed in the closeups.

of single layer is almost the same as multi-layer. However, for smooth regions, as shown in the closeups of Figure 10, the results from multi-layer transfer more texture. To summarize, single layer preserves less texture compared with multi-layer. This is one limitation of our method. However, even defining the style loss in a single layer, our method can still achieve perceptually amazing results.

Reconstruction versus Neural Patch Swap: We use neural patch swap [18] to generate the samples for training our decoder network. Neural patch swap also satisfies the two conditions of the target F_o. First, the reconstructed feature map keeps the appearance of content image. Second, [18] also introduces local statistics into the reconstructed feature map. Yet, as we show in Figure 11, the feature map

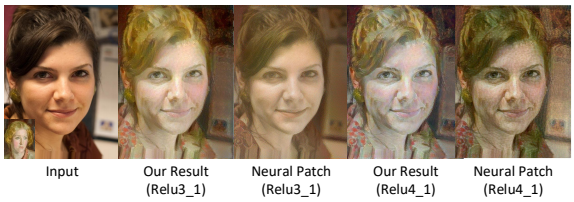

Figure 11. This figure shows the comparison of our results with neural patch swap [18]. Both neural patch swap and our method reconstruct a feature map preserving the appearance of content image and color statistics of style image. However, our method transfers more texture than neural patch swap.

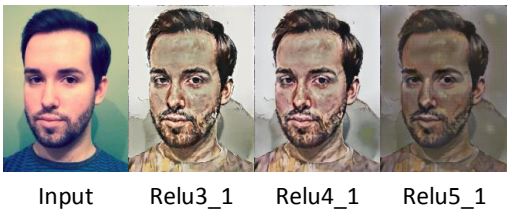

Figure 12. Limitation. Our method obtains low-quality results in $Relu5\_1$ due to the information loss of deeper layer. Only texture is transferred in deeper layer.

reconstructed by neural patch swap is only adaptive to the transfer between realistic images. Less texture is introduced into the stylized image compared with our method. Both methods use the same decoder network over $relu3\_1$ and $relu4\_1$ independently.

5.2. Quantitative Analysis

No Segmentation Masks: Neural doodle, Gatys2016 and our method use segmentation masks to guide the style transfer. Segmentation masks help alleviate common failure cases such as applying ground textures to sky regions. So from the qualitative aspect, our method essentially achieves better results compared with [9, 15]. If the segmentation masks are not used, our method is faster than the backward optimization methods [9] and slower than the feed-forward methods [15]. We calculate the average time of each method in 50 samples, the results are 112s [9], 1.35s (our method) and 0.063s [15]. Both our method and [9] use 500 iterations. Since our method explicitly minimizes the losses proposed by [9], our result is more similar to [9] which transfers more textures.

With Segmentation Masks: Both Neural doodle and Gatys2016 are based on back-propagation, while our method does not require any back-propagation which significantly decreases computational cost. In Gatys2016, the author also proposes a method to train a feed-forward network to apply different styles to different regions. We denote this method as Fast-Gatys2016 and show the results in Figure 9. Our method achieves similar result with its test images (Figure 9, Row 1). However, Fast-Gatys2016 does not achieve consistently pleasing results (Figure 9, Row 2). We train the network using the default setting and the released code of Fast-Gatys2016. We think the network training needs more regularization to enforce the connection between segmentation masks and styles. In contrast, our method directly incorporates segmentation masks in feature space which ensures the consistence of our results.

User Study: This study uses 19 portrait images as content and 12 portrait paintings as style, thus 228 stylized images are generated by each method. We show the content, style and stylized images to testers. We ask the testers to choose a score from 1 (worst) - 5 (best) for the purpose of evaluating the quality of style transfer. We do this user study with 100 testers online. The average scores of our method, neural doodle, Gatys2016 and Fast-Gatys2016 are 3.8, 4.0, 4.1 and 1.5. Each portrait image has the same resolution of 600 x 800. The average time costs of individual methods are 1.48s, 121.5s, 127.3s and 0.064s. This study shows that our method achieves a better tradeoff between quality and computation.

5.3. Limitation

In this section, we will discuss the limitations of our method. In our experiments, we find that using our framework directly in $relu5\_1$ will obtain low-quality results as shown in Figure 12. This might because the deeper layers contains less spatial information. Although the features from deeper layers have increased semantic meaning but they have lower resolution compared with those from the shallow layers. On the other hand, transferring in a single layer results in less texture. This problem can be alleviated by combining the features across layers. The topic of combining multi-layer features has been widely explored in the communities of semantic segmentation [12] and object detection [17]. Our method can also adopt these techniques to reconstruct and decode the concatenated feature map. We consider it as a future extension to our current method.

6. Summary

In this paper, we propose a new framework for fast semantic style transfer. We elucidate that the computation bottleneck of semantic style transfer is the propagation of the loss through the deep neural network. Different from training a transfer net to approximate the backward optimization, our method revisits the two parts of the original formulation [9]. We first solve the optimization of style transfer in feature space. Then the reconstructed feature is decoded into the stylized image. Our method demonstrates competitive results as backward optimization yet is much faster. Our exploration of achieving better style transfer results with slight computational cost may bring a promising

way to real-time applications. Furthermore, we hope our framework will inspire future research on content-aware image processing methods based on deep learning, like deep image harmonization, in-painting, etc.

Acknowledgements. This work was jointly supported by National Natural Science Foundation of China (Grant No.61132007, 61172125, 61601021, and U1533132).

References

[1] A. J. Champandard. Semantic style transfer and turning two-bit doodles into fine artworks. *arXiv preprint arXiv:1603.01768*, 2016.

[2] T. Q. Chen and M. Schmidt. Fast patch-based style transfer of arbitrary style. *arXiv preprint arXiv:1612.04337*, 2016.

[3] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.

[4] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011.

[5] S. Y. Duck. Painter by numbers, wikiart.org, 2016.

[6] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. 2016.

[7] L. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 262–270, 2015.

[8] L. A. Gatys, M. Bethge, A. Hertzmann, and E. Shechtman. Preserving color in neural artistic style transfer. *arXiv preprint arXiv:1606.05897*, 2016.

[9] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016.

[10] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman. Controlling perceptual factors in neural style transfer. *arXiv preprint arXiv:1611.07865*, 2016.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[12] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 447–456, 2015.

[13] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340. ACM, 2001.

[14] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

[15] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[16] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] T. Kong, A. Yao, Y. Chen, and F. Sun. Hypernet: towards accurate region proposal generation and joint object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 845–853, 2016.

[18] C. Li and M. Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2479–2486, 2016.

[19] C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision*, pages 702–716. Springer, 2016.

[20] Y. Li, N. Wang, J. Liu, and X. Hou. Demystifying neural style transfer. *arXiv preprint arXiv:1701.01036*, 2017.

[21] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

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

[23] M. Ruder, A. Dosovitskiy, and T. Brox. Artistic style transfer for videos. In *German Conference on Pattern Recognition*, pages 26–36. Springer, 2016.

[24] A. Selim, M. Elgharib, and L. Doyle. Painting style transfer for head portraits using convolutional neural networks. *ACM Transactions on Graphics (TOG)*, 35(4):129, 2016.

[25] X. Shen, A. Hertzmann, J. Jia, S. Paris, B. Price, E. Shechtman, and I. Sachs. Automatic portrait segmentation for image stylization. In *Computer Graphics Forum*, volume 35, pages 93–102. Wiley Online Library, 2016.

[26] Y. Shih, S. Paris, C. Barnes, W. T. Freeman, and F. Durand. Style transfer for headshot portraits. 2014.

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[28] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[29] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *Int. Conf. on Machine Learning (ICML)*, 2016.

[30] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[31] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. *arXiv preprint arXiv:1701.02096*, 2017.

[32] P. Wilmot, E. Risser, and C. Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. *arXiv preprint arXiv:1701.08893*, 2017.

Figure 13. Additional results on portrait style transfer. Our method shows compelling results with significantly decreased computational cost, thus having the potential to achieve real-time video portrait stylization.