

PHOTO STYLE TRANSFER WITH CONSISTENCY LOSSES

Xu Yao^{1,2*} Gilles Puy¹ Patrick Pérez³

¹Technicolor R&I, 975 avenue des Champs Blancs, 35576 Cesson-Sévigné, France

²Telecom ParisTech, 46 Rue Barrault, 75013 Paris, France

³Valeo.ai, 15 rue de la Baume, 75008 Paris, France

ABSTRACT

We address the problem of style transfer between two photos and propose a new way to preserve photorealism. Using the single pair of photos available as input, we train a pair of deep convolution networks (convnets), each of which transfers the style of one photo to the other. To enforce photorealism, we introduce a content preserving mechanism by combining a cycle-consistency loss with a self-consistency loss. Experimental results show that this method does not suffer from typical artifacts observed in methods working in the same settings [1, 2]. We then further analyze some properties of these trained convnets. First, we notice that they can be used to stylize other unseen images with same known style. Second, we show that retraining only a small subset of the network parameters can be sufficient to adapt these convnets to new styles.

Index Terms— style transfer, cycle-consistency loss, self-consistency loss.

1. INTRODUCTION

Image style transfer has been investigated for many years [3, 4, 5]. In the case where one wants to make a photo look like a painting, several innovative approaches using deep convnets have been proposed recently [6, 7, 8, 9, 10, 11, 12]. However, these methods fail for style transfer between photos as they generate strong artifacts and the results lack photorealism. This issue can be partly solved by applying a structure-preserving filter / regularization [1, 2, 13] based on the “matting Laplacian” [14]. But since this filter tries to generate an eye-pleasing image from an intermediate result with heavy distortions, it sometimes fails to recover all the structures visible in the input photo or to suppress stylization artifacts. In this work, we propose an alternative method for photo style transfer.

The recent work of Ulyanov *et al.* [15] shows that the sole structure of a deep convnet is able to capture low-level image statistics before any learning and, thus, can be used as an image prior. Given a deteriorated image, they show that optimizing the parameters of a randomly-initialized convnet, using

only this image for “training”, is sufficient to improve its quality. The authors demonstrate the efficiency of this technique for, *e.g.*, denoising, up-sampling and inpainting. Inspired by this work, we propose to train two networks for photo stylization using only the available pair of photos. All along the training, the networks are forced to preserve the structure of the input photos via cycle- and self-consistency mechanisms [16]. This approach successfully prevents structure distortions and generates satisfying photo stylization. This constitutes our main contribution. We compare our results to those obtained with state-of-the-art methods in Section 3. We then explore some properties of the trained convnets. First, even though these networks are trained using only a single pair of photos, we show that each network can transfer the learnt style to natural images not viewed at training time. The results are comparable to those obtained by training two new convnets using each new image and the original photo of the style. We note however that the images to be stylized need to have a similar semantic content to the original photo of the style. Second, to apply a new style, we show that only a small subset of the network parameters needs to be retrained (with however a good choice of the image pair used to pre-train the other parameters).

Related works. Optimization-based Neural Style Transfer [6] has attracted a wide interest as this is the first method to perform artistic style transfer using features extracted from a deep convnet. This method being slow, follow-up works [8, 7, 9, 10, 11, 12] proposed fast artistic stylization by training feed-forward networks to estimate the solution of [6] on a large collection of images. An improvement of the neural style transfer method was proposed by [1] to address photo stylization, with faster methods detailed in [2, 13]. In this work, we first concentrate on improving [1] with the construction of a new photo prior. This prior is constructed using two deep convnets that are trained from a random initialization using two images as inputs. Once trained each network can be used to stylize rapidly new images towards the style of one of the images used for training. Applying a new style needs however a partial retraining. Another related image transformation task is domain adaptation. Each style could be considered as a domain and deep convnets can be trained to transform images from a source domain to a target domain

*Work done when Xu Yao was a last-year student at Mines ParisTech.

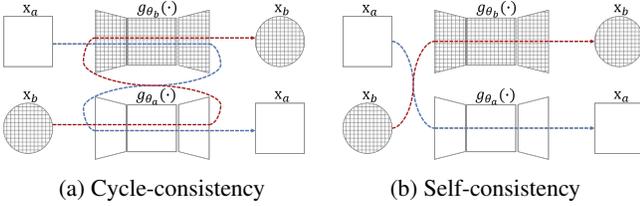


Fig. 1: The network $g_{\theta_a}(\cdot)$ transfers the style of the image \mathbf{x}_a to its input, while $g_{\theta_b}(\cdot)$ transfers the style of \mathbf{x}_b . Cycle-consistency is the fact that $g_{\theta_b}(g_{\theta_a}(\mathbf{x}_b)) \approx \mathbf{x}_b$ and $g_{\theta_a}(g_{\theta_b}(\mathbf{x}_a)) \approx \mathbf{x}_a$, while self-consistency is the fact that $g_{\theta_b}(\mathbf{x}_b) \approx \mathbf{x}_b$ and $g_{\theta_a}(\mathbf{x}_a) \approx \mathbf{x}_a$.

[17, 16, 18]. In this work, we use the cycle-consistency mechanism widely used for unpaired domain adaptation. Note however that we do not use any adversarial loss and that a domain/style in our case is not made of a collection but of a single image.

2. PHOTO STYLE TRANSFER NETWORK

Our goal is to transfer the style of an image \mathbf{x}_a to another image \mathbf{x}_b . We denote this stylized image by $\mathbf{x}_{b \rightarrow a}$. Following [1], $\mathbf{x}_{b \rightarrow a}$ is defined as a solution of

$$\arg \min_{\mathbf{x}} \lambda_c \mathcal{L}_c(\mathbf{x}, \mathbf{x}_b) + \lambda_s \mathcal{L}_s(\mathbf{x}, \mathbf{x}_a) + \lambda_L \mathcal{L}_L(\mathbf{x}), \quad (1)$$

where $\lambda_c, \lambda_s, \lambda_L > 0$. The content loss \mathcal{L}_c permits us to retain the content of \mathbf{x}_b in $\mathbf{x}_{b \rightarrow a}$, while the style loss \mathcal{L}_s aims at transferring the style of \mathbf{x}_a . The content loss is defined as the Euclidean distance between VGG-19 features [19] of \mathbf{x}_b and \mathbf{x} . The style loss is defined as the Euclidean distance between Gram matrices of VGG-19 features of \mathbf{x}_a and \mathbf{x} . Finally, \mathcal{L}_L is a regularization term favoring photorealism and built using the matting Laplacian L [14]. Due to space constraint, we let the reader refer to [1] for more details about these losses.

Although visually pleasing at the first glance, the photo stylization results obtained with the use of the matting Laplacian L present several undesirable artifacts (see Section 3). To avoid these disadvantages, we exploit the fact that both \mathbf{x}_a and \mathbf{x}_b are photorealistic to build a new content loss that preserves photorealism as much as possible.

2.1. Content preserving mechanism

First, we remove the loss \mathcal{L}_L involving the matting Laplacian in (1), or, equivalently, set $\lambda_L = 0$.

Second, taking inspiration from [15], we propose to use the constraint $\mathbf{x} = g_{\theta_a}(\mathbf{x}_b)$ in (1), where $g_{\theta_a}(\cdot)$ is a deep network with parameters θ_a that performs stylization toward the style of \mathbf{x}_a . Therefore, instead of directly minimizing the loss on \mathbf{x} , the minimization is conducted over θ_a .

Third, we borrow from [16] the idea of cycle consistency to construct a new content loss preserving input structures. Let $g_{\theta_b}(\cdot)$ be a deep network with parameters θ_b that performs

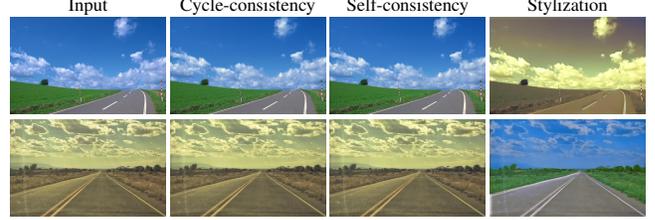


Fig. 2: Given a pair of input photos, the proposed cycle/self-consistencies yield images that are almost identical to the inputs while the stylization renders each photo in the style of the other.

stylization toward the style of \mathbf{x}_b . If we feed the stylized image $\mathbf{x}_{b \rightarrow a} = g_{\theta_a}(\mathbf{x}_b)$ into $g_{\theta_b}(\cdot)$, we expect to recover the original input image \mathbf{x}_b . Similarly, we expect $g_{\theta_a}(g_{\theta_b}(\mathbf{x}_a)) = \mathbf{x}_a$ (Fig. 1a). The first part of our new content loss reads

$$\tilde{\mathcal{L}}_1(\theta_a, \theta_b) = \mathcal{L}_c(g_{\theta_b}(g_{\theta_a}(\mathbf{x}_b)), \mathbf{x}_b) + \mathcal{L}_c(g_{\theta_a}(g_{\theta_b}(\mathbf{x}_a)), \mathbf{x}_a), \quad (2)$$

where we recall that \mathcal{L}_c is a perceptual loss constructed using VGG-19 features. As in the recent work on image inpainting [20], we use layers `conv1_1`, `conv2_1` and `conv3_1` of VGG-19.

Fourth, given the input image \mathbf{x}_a , the stylization network $g_{\theta_a}(\cdot)$ should be able to preserve \mathbf{x}_a : $\mathbf{x}_a = g_{\theta_a}(\mathbf{x}_a)$. Similarly, $\mathbf{x}_b = g_{\theta_b}(\mathbf{x}_b)$. We name this mechanism self-consistency (Fig. 1b). This yields the second part of our new content loss:

$$\tilde{\mathcal{L}}_2(\theta_a, \theta_b) = \mathcal{L}_c(g_{\theta_a}(\mathbf{x}_a), \mathbf{x}_a) + \mathcal{L}_c(g_{\theta_b}(\mathbf{x}_b), \mathbf{x}_b). \quad (3)$$

We remarked that using the cycle-consistency loss exclusively as content loss was not sufficient to preserve input structures. Using jointly the cycle-consistency and self-consistency losses improved the quality of our results. We show in Fig. 2 that the cycle- and self-consistencies are well respected by the pair of trained networks.

Finally, the stylization is controlled by the style loss \mathcal{L}_s , calculated for both styles:

$$\tilde{\mathcal{L}}_s(\theta_a, \theta_b) = \mathcal{L}_s(g_{\theta_a}(\mathbf{x}_b), \mathbf{x}_a) + \mathcal{L}_s(g_{\theta_b}(\mathbf{x}_a), \mathbf{x}_b), \quad (4)$$

where we recall that \mathcal{L}_s is defined by the distance between Gram matrices of VGG-19 features. We use layers `conv1_1`, `conv2_1` and `conv3_1`.

In total, we replace (1) by minimizing w.r.t. networks' parameters the complete loss:

$$\tilde{\mathcal{L}}(\theta_a, \theta_b) = \lambda_c \left[\tilde{\mathcal{L}}_1(\theta_a, \theta_b) + \tilde{\mathcal{L}}_2(\theta_a, \theta_b) \right] + \lambda_s \tilde{\mathcal{L}}_s(\theta_a, \theta_b). \quad (5)$$

In [16], the cycle-consistency loss is used in combination with an adversarial loss, which also contributes to the photorealism of their result. In our case, no adversarial loss is used. Let us also re-emphasize that, unlike in [16] and related works, $g_{\theta_a}(\cdot)$ and $g_{\theta_b}(\cdot)$ are trained using only two images:

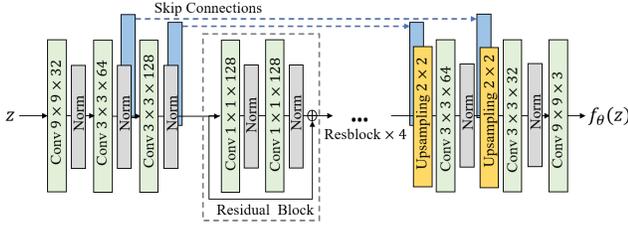


Fig. 3: Our networks consists of 16 convolution layers, of which 10 are in the residual blocks. Instance normalization is added after each layer except the last one. The 2nd and 3rd layers have a stride of 2. Two skip connections are added over the residual blocks.

x_a and x_b . Our method is also inspired by the work of [15], where one network is trained using one image (and a fixed random input) to perform, *e.g.*, denoising, up-sampling and inpainting. However, our trained networks can be applied to images not viewed at training time while the network trained in [15] remains specific to the image used for training.

2.2. Network architecture

We use a network architecture which has been proved effective by earlier works on artistic style transfer [7, 9] and domain adaptation [16]. The structure of the network is presented in Fig. 3. It is similar to that of [9] with two differences. First, inspired by U-net [21], we introduce skip connections between the second and penultimate layers, as well as between the third and ante-penultimate layers, with the aim of better preserving the structure of the input image.

Second, we also reduce the kernel size in the residual blocks [22] from 3×3 to 1×1 as, in our case, reducing the number of parameters had no impact on the quality of generated results while accelerating the training.

2.3. Implementation details

As in [1, 2, 13], we match the style of similar semantic regions (sky, building, lake, *etc.*) between two images by using semantic segmentation masks. We allocate one stylization network for each semantic region to prevent style mixing. Hence, for a pair of photos each with n corresponding semantic regions, we train $2n$ networks. Training is performed jointly for up to 8 semantic regions.

The style given to an image in the networks $g(\cdot)$ can be controlled by the instance normalization parameters as shown in [9, 11]. In order to reduce the number of trainable parameters, the two networks share the same convolutional layers but have different instance normalization layers.

The full objective function (5) contains six sub-losses¹, each involving the VGG-19 network. To reduce memory footprint, we randomly draw one of the sub-losses at each itera-

¹Two in (2), two in (3), two in (4).

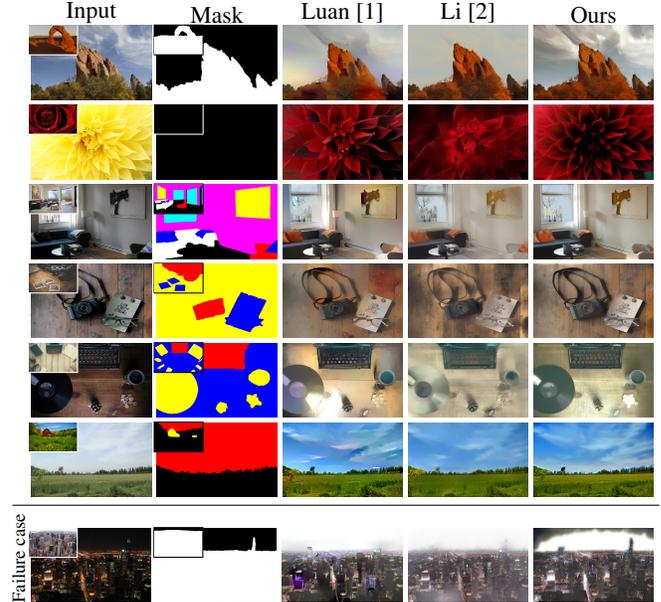


Fig. 4: Photo style transfer with the methods of [1], [2] and ours.

tion and apply a gradient step using this sub-loss only. The sub-losses are drawn using a uniform distribution and without replacement. This ensures that all the sub-losses are selected once every six iterations. For a pair of photos of size 700×400 , each with a single semantic region, the training process takes about 15 minutes on a NVIDIA Tesla P100 GPU. Our method was implemented using PyTorch [23].

For a fair comparison with existing methods, we post-process the obtained images using the same filter as in [1, 2, 13]. This post-processing permits ones to enhance fine details and global photorealism.

3. RESULTS

3.1. Comparison with state-of-the-art methods

We compare our results with those of [1, 2] in Fig. 4. The method of [1] has been described above. Li *et al.* [2] proposed a closed-form solution to the photo style transfer problem by using a feed-forward network for stylization followed by a smoothing step that favors photorealism.

Qualitatively, the results of [1] seem satisfying at first glance, but some of them present watercolor painting-like artifacts. The method [2] is fast and preserves well the content structure. Nevertheless, the smoothing step tends to weaken the stylization, with a lack of color saturation in the output image compared to the style image, and sometimes yields an undesired haze effect. In comparison, we find that our results respect better the original style while looking closer to real photos. They do not suffer from watercolor painting-like or hazy artifacts. However, we remarked that our approach sometimes generates inconsistent stylizations at the bound-

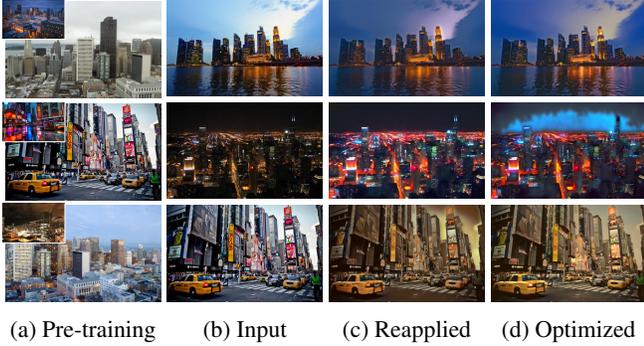


Fig. 5: Style transfer networks pre-trained on (a) have been reapplied on (b), getting stylized images (c). (d) are outputs generated by our basic approach using (b) and the small top image of (a) as reference image.

aries of different semantic regions. We also noticed that it has difficulties transferring city landscape images from night to day. We show such a failure case in the last row of Fig. 4. Note that the results obtained by [1] or [2] are also not entirely satisfying for this example. Finally, we remarked that the method of [2] has more difficulties to transfer the color saturation of the style image than the method of [1] or ours. A measure of the distance between the histograms (computed in the channel S of the HSV color model) of the style images and stylized images shows that, on average, [1] preserves the best color saturation followed by our method.

3.2. Generalization to unseen images

Even though trained on merely two images, our network can stylize images not viewed at training time. As an example, we first train style transfer networks on the pair of images in Fig. 5a. We then use these trained networks to transfer the styles of the small top images in Fig. 5a to the images in Fig. 5b. We obtained the stylized images in Fig. 5c, which preserve well the original structures while incorporating the target styles. For comparison, we show the results obtained with our original approach in Fig. 5d, *i.e.*, obtained after training new convnets for each image in Fig. 5b. The results are globally comparable to the results obtained using the pre-trained convnets. We nevertheless remarked that to get a reasonable stylization one should use images of similar semantic content as the one used to pre-train the style transfer networks. When there is a semantic difference, our approach still transfers the styles but may generate unrealistic results, such as blue house or red river. For the domelike artifact in Fig. 5d, there exist nearly invisible color differences in the dark area, so the network fails to transform all the area into blue.

3.3. Retraining for new styles

We mentioned in Section 2.3 that our stylization networks share the same convolutional layers but have different in-



Fig. 6: Style networks are pre-trained for each image pair in the first row. Fixing the convolutional layers of these pre-trained models, we retrain the normalization parameters to transfer the styles between the photos in (a). The results are the two bottom rows of (c) to (e) for each pre-trained model. The results generated by randomly initialized and fully trained networks are presented in (b).

stance normalization parameters, which control the styles. This design is due to Dumoulin *et al.* [9]. We show below that it is sufficient to retrain the instance normalization parameters to adapt our networks to a new style, even though the convolutional layers are pre-trained using a single pair of images. Given the extreme scarcity of training data, this property was not guaranteed, as the role of the convolutional filters and normalization parameters could have not been completely disentangled, letting the former still control part of the stylization.

We trained a pair of networks on a first pair of images. We fix the convolutional layers and retrain both networks using another pair of images by optimizing only on the instance normalization parameters. Fig. 6c-e present results obtained with this approach. The results are qualitatively comparable to those generated by randomly initialized and fully trained networks. Let us highlight nevertheless that this adaptation of the instance normalization parameters works better when the networks are pre-trained on two images of completely different styles (colorwise).

4. CONCLUSION

We designed a new method for effective photo stylization between two images that consists in training a pair of deep convnets with cycle- and self-consistency losses. Despite high-quality results on several examples, there is still room for improvement of our results. In particular, we should try to reduce the artifacts at the boundary of different semantic regions and reduce the overexposure that sometimes appears in small regions of the results. A direct extension of this work could be to train another network to predict the style parameters in $g_{\theta}(\cdot)$ directly from an image, while using the proposed loss, to be able to use arbitrary style at runtime.

5. REFERENCES

- [1] F. Luan, S. Paris, E. Shechtman, and K. Bala, “Deep photo style transfer,” in *CVPR*, 2017.
- [2] Y. Li, M.-Y. Liu, X. Li, M.-H. Yang, and J. Kautz, “A closed-form solution to photorealistic image stylization,” in *ECCV*, 2018.
- [3] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, “Image analogies,” in *SIGGRAPH. ACM*, 2001.
- [4] A. A. Efros and W. T. Freeman, “Image quilting for texture synthesis and transfer,” in *SIGGRAPH. ACM*, 2001.
- [5] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley, “Color transfer between images,” *IEEE Computer graphics and applications*, vol. 21, no. 5, 2001.
- [6] L. A. Gatys, A. S. Ecker, and M. Bethge, “Image style transfer using convolutional neural networks,” in *CVPR*, 2016.
- [7] J. Johnson, A. Alahi, and F.-F. Li, “Perceptual losses for real-time style transfer and super-resolution,” in *ECCV*. Springer, 2016.
- [8] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky, “Texture networks: Feed-forward synthesis of textures and stylized images,” in *ICML*, 2016.
- [9] V. Dumoulin, J. Shlens, and M. Kudlur, “A learned representation for artistic style,” *Proc. of ICLR*, 2017.
- [10] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens, “Exploring the structure of a real-time, arbitrary neural artistic stylization network,” in *BMVC*, 2017.
- [11] X. Huang and S. J. Belongie, “Arbitrary style transfer in real-time with adaptive instance normalization,” in *ICCV*, 2017.
- [12] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, “Universal style transfer via feature transforms,” in *Advances in Neural Information Processing Systems*, 2017.
- [13] G. Puy and P. Pérez, “A flexible convolutional solver with application to photorealistic style transfer,” *arXiv:1806.05285*, 2018.
- [14] A. Levin, D. Lischinski, and Y. Weiss, “A closed-form solution to natural image matting,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, 2008.
- [15] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Deep image prior,” *arXiv:1711.10925*, 2017.
- [16] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *ICCV*, 2017.
- [17] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in *CVPR*, 2017.
- [18] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, “High-resolution image synthesis and semantic manipulation with conditional gans,” in *CVPR*, 2018.
- [19] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv:1409.1556*, 2014.
- [20] H. V. Vo, N. Q. Duong, and P. Perez, “Structural inpainting,” *arXiv:1803.10348*, 2018.
- [21] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015.
- [22] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *CVPR*, 2016.
- [23] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” in *NIPS-W*, 2017.