A SIMPLE WAY OF MULTIMODAL AND ARBITRARY STYLE TRANSFER

Anh-Duc Nguyen, Seonghwa Choi, Woojae Kim, and Sanghoon Lee

Yonsei University

ABSTRACT

We re-define multimodality and introduce a simple approach to multimodal and arbitrary style transfer. Conventionally, style transfer methods are limited to synthesizing a deterministic output based on a single style, and there has been no work that can generate multiple images of various details, or multimodality, given a single style. In this work, we explore a way to achieve multimodal and arbitrary style transfer by injecting noise to a unimodal method. This novel approach does not require any trainable parameters, and can be readily applied to any unimodal style transfer methods with separate style encoding sub-network in literature. Experimental results show that while being able to transfer an image to multiple domains in various ways, the image quality is highly competitive with contemporary models in style transfer.

***Index Terms*—** Image style transfer, convolutional neural network, deep learning.

1. INTRODUCTION

Image style transfer, which concerns with applying the look of an image to another, has become an increasingly active topic. The need for such an application arises naturally as users may not simply want their photos to be realistic but have an artistic or vintage look from famous paintings as well.

The concept of transferring texture from an image to another was introduced by Hertzman *et al*. [1], but the method considers only low-level feature transfer. The first study that brought deep learning into the picture is perhaps DeepDream [2]. Later, Gatys *et al*. [3] proposed a deep-learning-based method producing more realistic stylized images and showed that deep convolutional neural networks (CNNs) can separate style and content of an image in feature space. However, despite its good performance, the online optimization is frustratingly long. To speed up the transfer time, subsequent works [4] designed end-to-end learning architectures that perform optimization offline, and the final models can do inference even in real time. Li *et al*. [5] presented a whitening and coloring transform in the latent space of an autoencoder and the latent representations of the input and style can be blended to produce the stylized output. Luan *et al*. [6] focused on how to transfer photographic styles to other images. Dumoulin *et al*. [7] proposed conditional instance normalization for image style transfer, which is subsequently refined and upgraded in [8] as adaptive instance normalization (AdaIN). All these methods mainly deal with unimodal style transfer, meaning that given a style image and a target, only one output having the texture of the style and content of the target is produced.

There are several works concerning with multiple styles and multiple outputs in literature. Wang *et al*. [9] proposed to blend multiple styles into an image, but their method can work with only a fixed set of styles. Recently, in cross-domain image translation, Huang *et al*. [10] presented a one-to-many mapping CNN model that deals with synthesizing multiple outputs given a single input. However, the number of domains are limited to only two.

In this work, we re-define multimodality and consider a different scenario in style transfer. Since aesthetic preference may differ from user to user, a deterministic output may not satisfy a wide range of users. Ideally, given a target image and a style image, it is favorable that there are multiple generated image candidates which bear the same style but different color, edge, and some detail information so the users can select the best one based on their preferences. To our knowledge, this is the first work to consider such a scenario. To achieve the multimodality property, based on the work of Huang *et al*. [8], we explore a MULtimodal and ARbitrary style transfer module (MULTAR) which injects noise into the style encoder to produce outputs with varied details. Our contribution is two-fold: (1) we gain an insight into understanding how AdaIN transfers style in feature space and based on the analysis, (2) we propose a simple yet powerful method that helps a unimodal style transfer to be able to generate multiple outputs with slightly different characteristics. Our approach is flexible and can be easily incorporated into any system having a separate style encoding branch. Experiments suggest that our model can produce various features in the output image given a single style input.

2. ADAPTIVE INSTANCE NORMALIZATION FOR STYLE TRANSFER

Since our work is an extension of the AdaIN method [8], we briefly summarize the key idea in this section. The overview of the method is shown in Fig. 1. Let us call the VGG19 [11] encoder as $f(.)$ and the decoder as $g(.)$. Let us also denote the input image and style image as x and y, respectively. AdaIN

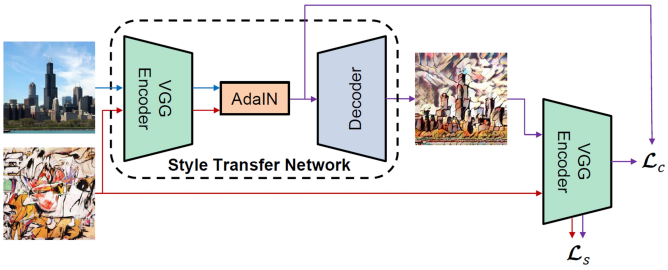

Fig. 1. Overview of the AdaIN method [8], which is also the backbone of our proposed model. Figure is reproduced from the original paper.

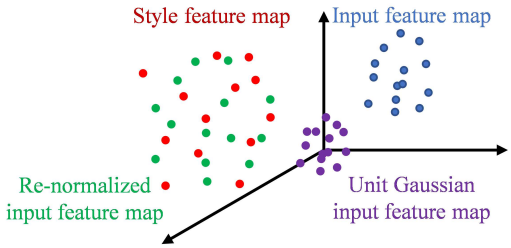

Fig. 2. In AdaIN, the input feature map (blue cloud) is first normalized to a unit Gaussian (purple cloud), and then provided with the mean and variance of the style feature map (red cloud) to become a new representation (yellow cloud) in feature space.

takes two 2D feature maps, and performs normalization on one input, but then uses the statistics of the other input to scale and shift the normalized input instead of using learnable parameters like the usual instance normalization. Mathematically, AdaIN is defined as

$$AdaIN(u, v) = \sigma(v)\frac{u - \mu(u)}{\sigma(u)} + \mu(v), \tag{1}$$

where $\mu(.)$ and $\sigma(.)$ extract the spatial first- and second-order statistics of the input feature maps u and v. In the implementation, we perform $AdaIN(f(x), f(y))$. The AdaIN module is followed by a generator to synthesize the style-transferred image $\hat{x} = g(AdaIN(f(x), f(y)))$. The method by design, however, allows only transfer of the style from an image to another. Because of the impressive performance of AdaIN, in this study, we extend this method so that it is able to synthesize diverse-style outputs based on a single style image.

Before going into details, we present an analysis that leads to the inception of the proposed method. As indicated in [8], style transfer can be done by transferring the statistics between the feature maps of an input and a style image. Visually, the process can be illustrated as in Fig. 2, which shows the re-normalization of an input feature map (blue cloud) using the statistics of a style feature map (red cloud). We conjecture that by shifting the point cloud to a new position, the output (green cloud) can represent a new style (for *e.g.*, a thick stroke), and by scaling the map, the network can control the influence of that style in the result (for *e.g.*, how dominant it is compared to other features). Thus, modifying the scales and preserving the means of the feature maps can produce different features without distorting the given style. Moreover, from the feature space point of view, a linear combination of feature map is as good as the natural basis because it is the feature space that contains semantic information rather than each individual unit [12]. Having these insights, in order to produce different characteristics in the outputs, we can utilize different combinations of the style feature maps encoded by the VGG19 encoder. In the following section, we present a novel approach to execute that idea.

3. MULTAR

3.1. Proposed model

In Section 2, we hypothesize that changing the variances of the feature maps can result in different details in the output images. However, a naive scaling would lead to a network breakdown since the VGG19 backend is kept fixed during the optimization, so any scaling change in the feature maps is not accommodated by the pre-trained weights, which causes irregular activations. In other words, it is required to scale the feature maps in a layer so that the expectation of the inner products between the weights and the features are intact. Let us denote the weights, feature maps of the style image, and scale in layer l by W_l, y_l, and s_l, respectively. Since convolution can be formulated into inner product, for simplicity, we consider inner product here. For a feature map, we enforce the following equality to hold

$$\begin{aligned} E[(W_l^T y_l) \odot s_l] &= E[W_l^T y_l] \odot E[s_l] \\ &= E[W_l^T y_l], \end{aligned} \tag{2}$$

which implies the noise has one-mean. The operator $\odot$ stands for the Hadamard product. The first equality in 2 is the result of the assumption that the inner product and the noise[1] are independent. Also, since scaling by a negative value makes no intuitive sense, we keep the noise strictly positive as well.

Having identified how we can scale the feature maps, inspired by the formation of AdaIN, we propose to scale in a similar way as AdaIN does. Concretely, for each feature map of the style image in each layer of VGG19, we perform the following re-normalization

$$y_l := s_l \odot (y_l - \mu(y_l)) + \mu(y_l), \tag{3}$$

where $\mu(.)$ is taken over the spatial positions. The reason we subtract the mean and add it back after re-scaling is that s may not be perfectly one-mean and we do not expect the mean of y_l to be shifted, which may lead to some undesirable change

[1] We use scale and noise interchangeably from then on.

in the style representation in the feature space. Even though the noise can be explicitly shifted to have one-mean, doing so may risk resulting in negative values. This approach offers multi-style outputs with no change in the number of trainable parameters and negligible additional processing time.

3.2. Loss function

In the original work [8], the VGG encoder is used to ensure that the style code of the generated image is the same as that of the style image. However, in our case, the feature maps are perturbed when extracting the latent code. For this reason, we must take that into account when computing the loss. Let us define $\tilde{f}$ the encoder in which the feature maps in each layer are scaled. The style loss, denoted by $\mathcal{L}_s$, is

$$\mathcal{L}_s = \sum_{l \in L} \left\| \mu(f_l(\hat{x})) - \mu(\tilde{f}_l(y)) \right\|_2^2 + \left\| \sigma(f_l(\hat{x})) - \sigma(\tilde{f}_l(y)) \right\|_2^2, \tag{4}$$

where L contains `relu1_1`, `relu2_1`, `relu3_1`, and `relu4_1` of the VGG encoder, and $\hat{x}$ is the generated image. The content loss is almost the same as in [8] but with the noise taken into account

$$\mathcal{L}_c = \left\| AdaIN(f(x), \tilde{f}(y)) - f(\hat{x}) \right\|. \tag{5}$$

Thus, the total loss to minimize is

$$\mathcal{L} = \mathcal{L}_c + \alpha \mathcal{L}_s, \tag{6}$$

where α is a constant to weight $\mathcal{L}_s$.

3.3. Implementation details

For training, we utilized a subset of the MS COCO training set [13] and the provided training split of the Wikiart dataset [14]. We tested the proposed model on the MS COCO test set, Wikiart test set, several provided images in [8,9], and some random images from the Internet. The network structure is the same as in [8] except the output of the decoder being activated by a hyperbolic tangent and the max pooling in VGG19 being replaced by mean pooling. By default, we employed a uniform distribution of one-mean having support of $[0.5, 1.5]$. We found that training with noise from the beginning makes the optimization very difficult. Therefore, we bootstrapped the network by training the decoder the same as in [8] for 20 epochs, and then fine-tune the network for 4 more epochs with the noise injection, new loss, and $\alpha = 1$. The model is implemented in Theano [15].

4. EXPERIMENT

For space considerations, the major content in this section is qualitative results as no known metric is able to fully replace human opinion [16–22], especially in the image synthesis area. Code and more visualizations are available at https://github.com/justanhduc/multar.

4.1. Qualitative performance

For references, we chose the original AdaIN [8] and Multimodal Transfer (MT) [9]. Figs. 3 shows the visual comparisons between the proposed model and the two reference methods. In each case, we synthesized three outputs corresponding to three randomly drawn scales. Our results offer diversity in features, and even look better and more realistic than the benchmarking methods. To check the generalization of the proposed module, Fig. 4 presents more results of Multar using some downloaded images from the Internet. As can be seen from the figure, the outputs have varied characteristics, which lets users select the one they find most artistic.

In terms of inference speed and memory, our model consumes as much resource as AdaIN. The proposed module is versatile that it does not require more trainable parameters nor any ad-hoc training scheme, and it can achieve multimodality with almost no change in computational time and resource.

4.2. Ablation study

We experimented with different distribution supports for the noise. Any noise with mean different from 1 (for *e.g.*, 2) results in complete failure. Regarding the noise support, we additionally tested the network with noise ranges of $[0, 2]$ and $[0.25, 1.75]$, and the results are shown in Fig. 5. As anticipated, wider noise ranges contribute to more diverse features in the results but make it harder to optimize, which makes the results appear noisier. Also, as we put more weight to the α term, the outputs possess more features from the style image but look far less realistic. In general, we found that our choice of hyperparameters brings the most satisfaction.

5. CONCLUSION

In this paper, we have defined a new scenario of multimodality and proposed a method to transform a unimodal style transfer method into a multimodal one. Inspired by the way AdaIN works, we presented an efficient way to scale the feature maps in the style encoding branch, and the model is able to synthesize different outputs of different details given a single style image. Experiments showed that the model can generate an infinite number of different outputs with competitive quality among state-of-the-art methods in the field.

Acknowledgement

This work was supported by Samsung Research Funding Center of Samsung Electronics under Project Number SRFC-IT1702-08.

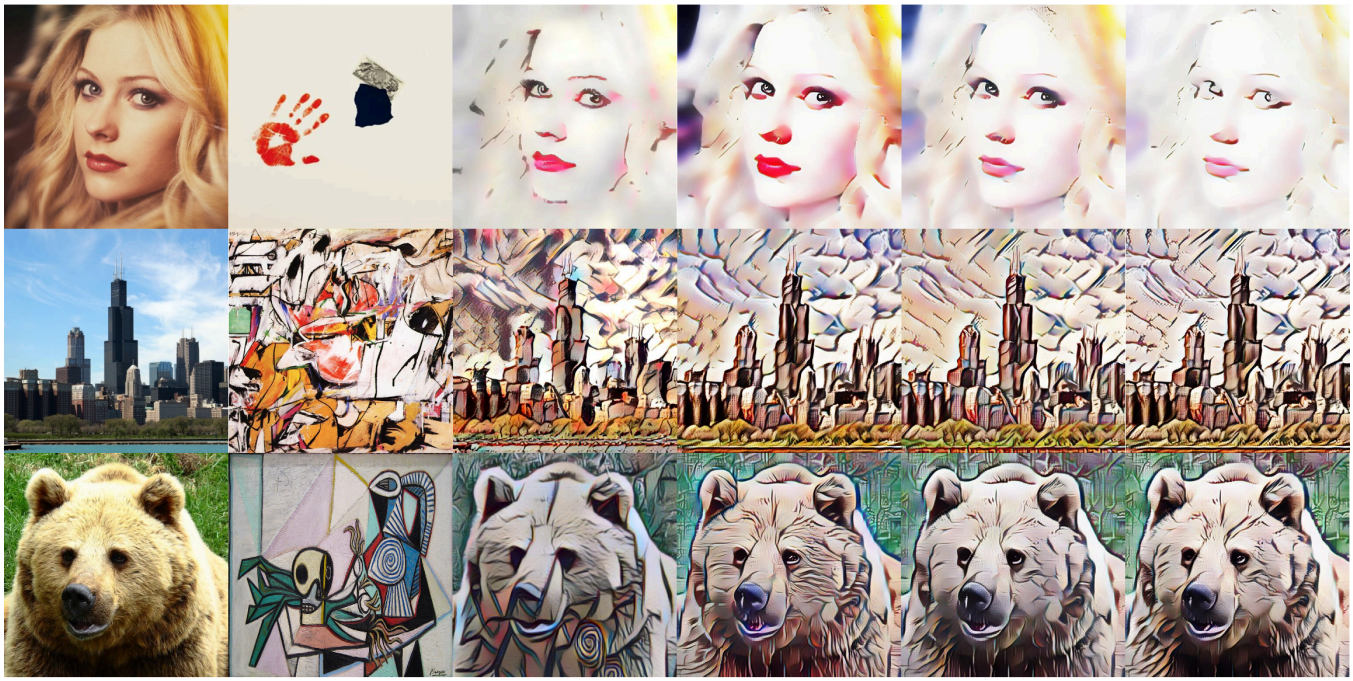

Fig. 3. Qualitative results against AdaIN [8] (the first two rows in the third column) and MT [9] (the last row in the third column). The AdaIN results are taken from the official Github repository. The MT results are obtained by running a re-implementation on a Github repository. We exhibit three different images produced by injecting different one-mean uniform noises. Best viewed in zoom and color.

Fig. 4. Our results with random targets and styles from Internet. The first and second columns are targets and styles, respectively. The others are generated images. Best viewed in zoom and color.

Fig. 5. Ablation study on noise level. From left to right: A target, a style image and stylized images with the noise distribution supports of $[0.5, 1.5]$, $[0.25, 1.75]$, and $[0, 2]$. Best viewed in color and zoom.

6. REFERENCES

[1] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin, "Image analogies," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. pp. 327–340, ACM. 1

[2] Alexander Mordvintsev, Christopher Olah, and Mike Tyka, "Inceptionism: Going deeper into neural networks," *Google Research Blog. Retrieved June*, vol. 20, no. 14, pp. 5, 2015. 1

[3] Leon A Gatys, Alexander S Ecker, and Matthias Bethge, "Image style transfer using convolutional neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition*. pp. 2414–2423, IEEE. 1

[4] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S Lempitsky, "Texture networks: Feed-forward synthesis of textures and stylized images," in *ICML*, pp. 1349–1357. 1

[5] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang, "Universal style transfer via feature transforms," in *Advances in Neural Information Processing Systems*, pp. 386–396. 1

[6] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala, "Deep photo style transfer," *CoRR, abs/1703.07511*, vol. 2, 2017. 1

[7] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur, "A learned representation for artistic style," *Proc. of ICLR*, 2017. 1

[8] Xun Huang and Serge J Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *ICCV*, pp. 1510–1519. 1, 2, 3, 4

[9] Xin Wang, Geoffrey Oxholm, Da Zhang, and Yuan-Fang Wang, "Multimodal transfer: A hierarchical deep convolutional neural network for fast artistic style transfer," in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2017, vol. 2, p. 7. 1, 3, 4

[10] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz, "Multimodal unsupervised image-to-image translation," *arXiv preprint arXiv:1804.04732*, 2018. 1

[11] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 1

[12] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013. 2

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. pp. 740–755, Springer. 3

[14] Wei Ren Tan, Chee Seng Chan, Hernán E Aguirre, and Kiyoshi Tanaka, "Ceci n'est pas une pipe: A deep convolutional network for fine-art paintings classification," in *2016 IEEE International Conference on Image Processing*. pp. 3703–3707, IEEE. 3

[15] The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, and Anatoly Belikov, "Theano: A python framework for fast computation of mathematical expressions," *arXiv preprint arXiv:1605.02688*, 2016. 3

[16] Jongyoo Kim, Anh-Duc Nguyen, Sewoong Ahn, Chong Luo, and Sanghoon Lee, "Multiple level feature-based universal blind image quality assessment model," in *2018 IEEE International Conference on Image Processing*. pp. 291–295, IEEE. 3

[17] Woojae Kim, Jongyoo Kim, Sewoong Ahn, Jinwoo Kim, and Sanghoon Lee, "Deep video quality assessor: From spatio-temporal visual sensitivity to a convolutional neural aggregation network," in *European Conference on Computer Vision*, pp. 219–234. 3

[18] Jongyoo Kim, Anh-Duc Nguyen, and Sanghoon Lee, "Deep cnn-based blind image quality predictor," *IEEE transactions on neural networks and learning systems*, , no. 99, pp. 1–14, 2018. 3

[19] Jongyoo Kim and Sanghoon Lee, "Deep learning of human visual sensitivity in image quality assessment framework," in *2017 IEEE Conference on Computer Vision and Pattern Recognition*. 3

[20] Jongyoo Kim and Sanghoon Lee, "Fully deep blind image quality predictor," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 1, pp. 206–220, 2017. 3

[21] Jongyoo Kim, Hui Zeng, Deepti Ghadiyaram, Sanghoon Lee, Lei Zhang, and Alan C Bovik, "Deep convolutional neural models for picture-quality prediction: Challenges and solutions to data-driven image quality assessment," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 130–141, 2017. 3

[22] Sewoong Ahn and Sanghoon Lee, "Deep blind video quality assessment based on temporal human perception," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. pp. 619–623, IEEE. 3