ARBITRARY STYLE TRANSFER IN REAL-TIME WITH ADAPTIVE INSTANCE NORMALIZATION

Xun Huang & Serge Belongie
Department of Computer Science
Cornell University
Ithaca, NY 14850, USA
{xh258,sjb344}@cornell.edu

ABSTRACT

Gatys et al. (2015) recently introduced a neural algorithm that renders a content image in the style of another image, achieving so-called *style transfer*. However, their framework requires a slow iterative optimization process, which limits its practical application. Fast approximations with feed-forward neural networks have been proposed to speed up neural style transfer. Unfortunately, the speed improvement comes at a cost: the network is usually tied to a fixed set of styles and cannot adapt to arbitrary new styles. In this paper, we present a simple yet effective approach that for the first time enables arbitrary style transfer in real-time. At the heart of our method is a novel adaptive instance normalization (AdaIN) layer that aligns the mean and variance of the content features with those of the style features. Our method achieves speed comparable to the fastest existing approach, without the restriction to a pre-defined set of styles.

1 INTRODUCTION

The seminal work of Gatys et al. (2015) showed that deep neural networks encode not only the content but also the *style* information of an image. Moreover, the style and content of an image are somewhat separable: it is possible to change the style of an image while preserving its content. However, the original style transfer algorithm of Gatys et al. (2015) relies on a prohibitively slow optimization process, which can take up to several minutes even with modern GPUs.

Significant effort has been devoted to accelerating neural style transfer. Several independent works have attempted to train a style transfer network that performs fast feed-forward stylization (Johnson et al., 2016; Ulyanov et al., 2016a; Li & Wand, 2016). *Instance normalization* (IN), a layer similar to batch normalization (BN) (Ioffe & Szegedy, 2015) but performing normalization within a single example instead of a mini-batch, has been found to significantly improve the quality of feed-forward stylization (Ulyanov et al., 2016b). A major limitation of these feed-forward approaches is that each network is tied to a fixed style. In practical application scenarios, it becomes impossible to transfer the input image to an arbitrary user-specified style. Dumoulin et al. (2017) addressed this problem by introducing a *conditional instance normalization* layer with which a single network encodes multiple (32 in their experiments) styles at the same time. Still, the network is constrained to a finite set of styles and their interpolations. Very recently, Chen & Schmidt (2016) proposed a feed-forward architecture that can perform arbitrary style transfer thanks to a style swap layer. However, their style swap layer creates a new computational bottleneck, preventing it from real-time applications.

It seems that normalization layers play a special role in style transfer: (1) replacing batch normalization with instance normalization dramatically improves quality and (2) learning different normalization parameters enables the network to encode different styles. In this paper, we take a closer look at the role of instance normalization in style transfer. We conjecture that **instance normalization performs style normalization**. Based on this hypothesis, we propose a new normalization scheme named *adaptive instance normalization* (AdaIN). AdaIN takes two feature maps as inputs, and simply adjusts the channel-wise mean and variance of the content feature map to match those of the style feature map. Through experiments, we find AdaIN effectively transfers the style of the style feature map to the content feature map. A decoder network is then learned to map the transformed

feature map back to the image space. Our method is three orders of magnitude faster than Gatys et al. (2015), without sacrificing the flexibility of transferring inputs to arbitrary new styles.

2 BACKGROUND

2.1 INSTANCE NORMALIZATION

Ulyanov et al. (2016a) recently demonstrated the possibility of training a feed-forward neural network that transforms a given content image to a predefined style. Later, Ulyanov et al. (2016b) found that significant quality improvement could be achieved simply by replacing batch normalization layers with instance normalization layers:

$$\text{IN}(x) = \gamma \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \beta \tag{1}$$

Different from BN layers, here $\mu(x)$ and $\sigma(x)$ are the mean and standard deviation, computed across spatial dimensions independently for each channel *and each sample*. γ and β are scaling and shifting parameters that are learned from data. Another difference with BN is that IN is also applied at test time unchanged, while BN usually replaces mini-batch statistics with population statistics.

2.2 CONDITIONAL INSTANCE NORMALIZATION

Instead of learning a single set of scaling and shifting parameters γ and β, Dumoulin et al. (2017) proposed to learn a different set of parameters γ_s and β_s for each style s:

$$\text{CIN}(x; s) = \gamma_s \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \beta_s \tag{2}$$

During training, a style image and its index s are randomly chosen from a fixed set of styles $s \in 1, 2, ..., S$. A style transfer network then transfers the content image to the chosen style, in which the corresponding γ_s and β_s are used in the conditional instance normalization layers of the network. The style loss is computed with respect to the chosen style image.

3 INTERPRETING INSTANCE NORMALIZATION

Despite the great success of (conditional) instance normalization, the reason why they work particularly well for style transfer remains elusive. Ulyanov et al. (2016b) attribute the success of IN to its invariance to the contrast of the content image. However, IN takes place in the feature space, therefore it should have more profound impacts than a simple contrast normalization in the pixel space. Empirically, we also found that IN remains effective even when all the training images are normalized to the same contrast. Perhaps even more surprising is the fact that the scaling and shifting parameters in IN layers could completely change the style of the output image.

It has been known that the feature statistics in convolutional layers of a deep neural network encode the style information of an image. While Gatys et al. (2015) use the correlation statistics as their optimization objective, Li et al. (2017) recently showed that matching other statistics, including per-channel mean and variance, are also effective for style transfer. Motivated by these facts, we hypothesize that instance normalization performs a form of *style normalization* by normalizing the feature statistics (per-channel mean and variance). Our hypothesis could easily explain IN's effectiveness in style transfer: (1) In Ulyanov et al. (2016b), the shifting and scaling parameters in IN layers are learned to normalize the feature maps to the (fixed) target style. Training is facilitated because the rest of the network can focus on content manipulation while discarding the original style information. (2) In Dumoulin et al. (2017), different IN parameters can be learned to normalize the input to different styles.

4 ADAPTIVE INSTANCE NORMALIZATION

Inspired by our interpretation that instance normalization performs style normalization, we propose an adaptive instance normalization (AdaIN) layer that *adaptively* normalizes the input to an arbitrarily given style. AdaIN receives a content input x and a style input y. Unlike BN, IN or conditional IN, AdaIN no longer has any learnable parameters. Instead, it adaptively computes the scaling and shifting parameters from the style input:

$$\text{AdaIN}(x, y) = \sigma(y) \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \mu(y) \tag{3}$$

in which we simply use the standard deviation of style input $\sigma(y)$ as scaling, and the mean of style input $\mu(y)$ as shifting. Similar to IN, the statistics are computed across spatial locations.

5 EXPERIMENTS

5.1 TRAINING

Our style transfer network takes a content image and an arbitrary style image as inputs, and produces an output image that recombines the content of the former and the style latter. We adopt an encoder-decoder architecture, in which the encoder is fixed to the first few layers (up to `relu4_1`) of a pre-trained VGG-19 (Simonyan & Zisserman, 2015). After encoding both the content and style images in feature space, we feed both feature maps to an AdaIN layer which aligns the mean and variance of the content feature map to those of the style feature map. A randomly initialized decoder is trained to map the AdaIN output back to the image space. The pre-trained VGG-19 is again used to compute the content loss and style loss for the decoder. An illustration of our architecture can be found in Appendix A. Code will be made publicly available.

Interestingly, we found that the decoder could *not* be successfully trained if it contains IN layers. This supports our hypothesis: using IN in the decoder will normalize the output to a single style, preventing the decoder from generating images in different styles. We also experimented with a baseline method using concatenation instead of AdaIN to combine style and content. However, the decoder fails to disentangle the style information from the content of the style image

5.2 RESULTS

In Appendix B we show example results generated by our style transfer network. Our model is able to produce visually appealing results for arbitrary new styles that are never observed during training. Our method also achieves comparable speed (around 0.02 seconds for a 256×256 image) to Ulyanov et al. (2016a); Johnson et al. (2016), because our AdaIN layer only introduces a negligible computation cost. However, the quality of our generated images are competitive to the optimization-based method (Gatys et al., 2015) and the feed-forward method specific to a single style (Ulyanov et al., 2017), while in some cases our method is slightly behind in quality.

6 CONCLUSION

In this paper, we revisit the role of instance normalization in style transfer, and present a simple extension that for the first time allows arbitrary style transfer in real time. Our preliminary experiments have demonstrated very promising results. We are currently exploring more complicated network architectures such as a residual architecture (Johnson et al., 2016) or an architecture with additional skip connections from the encoder. We also plan to investigate whether replacing AdaIN with correlation alignment (Sun et al., 2016) could further improve quality. Another interesting direction would be applying AdaIN to texture synthesis.

ACKNOWLEDGMENTS

We thank Facebook for the donation of GPU servers used in our experiments.

A ARCHITECTURE

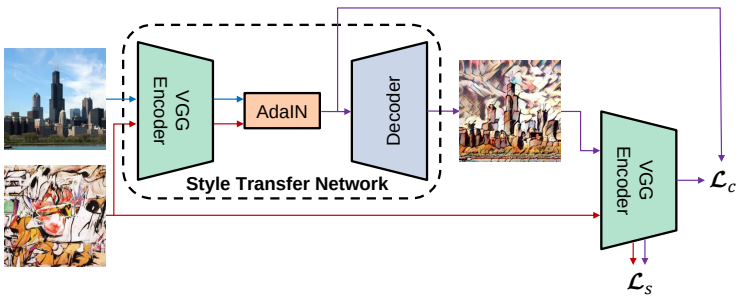


Figure 1: An overview of our style transfer network.

B EXAMPLES

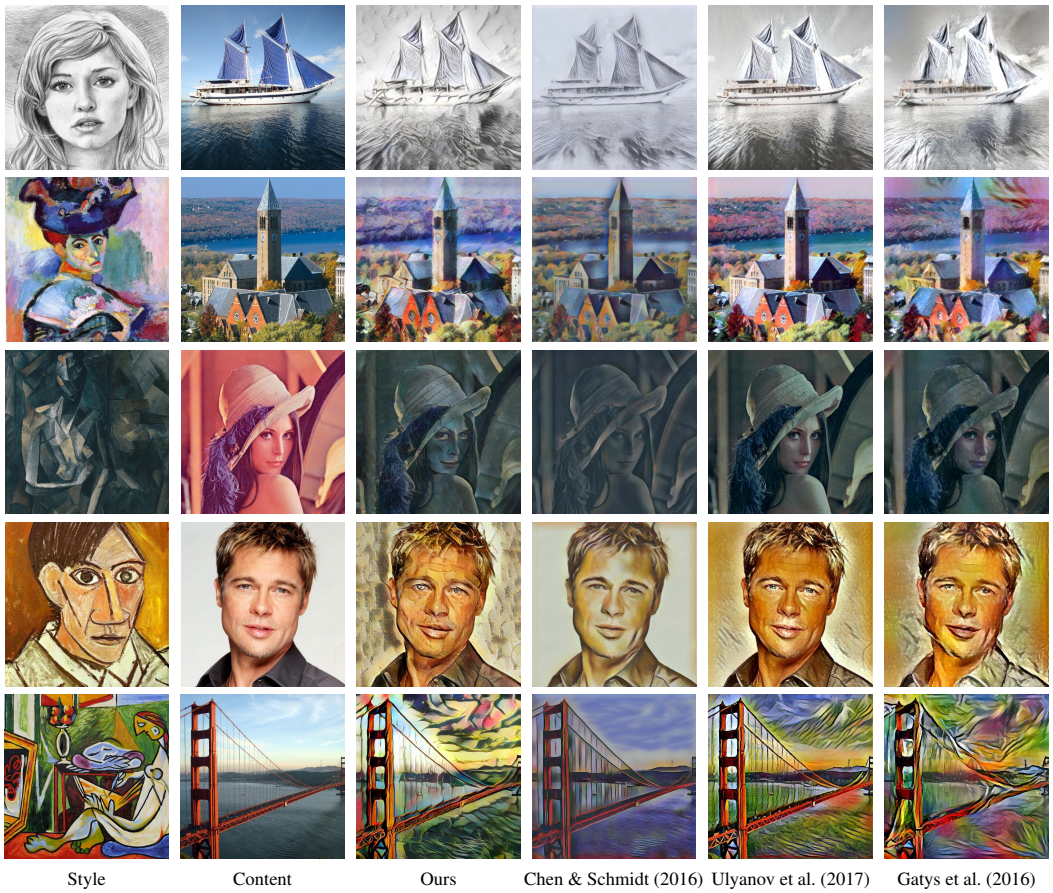
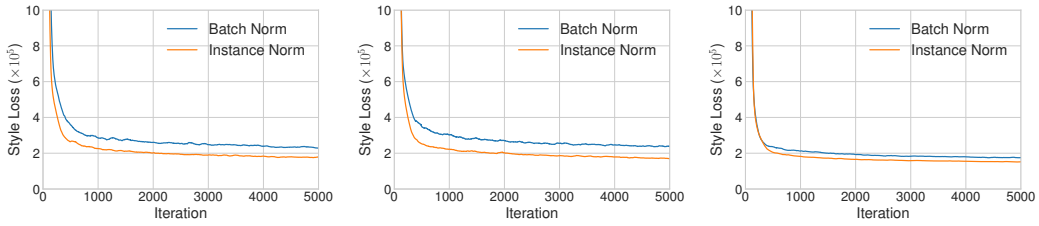


Figure 2: Example style transfer results. All the tested content and style images are never observed by our network during training.

C ADDITIONAL EXPERIMENTS

We run the code of improved texture networks Ulyanov et al. (2017) to perform single-style transfer, with IN or BN layers. As expected, the model with IN converges faster than the BN model (Fig. 3 (a)). To test the explanation in Ulyanov et al. (2017), we then normalize all the training images to the same contrast by performing histogram equalization on the luminance channel. As shown in Fig. 3 (b), IN remains effective, suggesting the explanation in Ulyanov et al. (2017) to be incomplete. To verify our hypothesis, we normalize all the training images to the same style (different from the target style) using a pre-trained style transfer network provided by Johnson et al. (2016). According to Fig. 3 (c), the improvement brought by IN become much smaller when images are already style normalized. The remaining gap can explained by the fact that the style normalization with Johnson et al. (2016) is not perfect. Also, models with BN trained on style normalized images can converge as fast as models with IN trained on the original images. Our results indicate that IN does perform a kind of style normalization.



(a) Trained with original images. (b) Trained with contrast normalized images. (c) Trained with style normalized images.

Figure 3: To understand the reason for IN's effectiveness in style transfer, we train an IN model and a BN model with (a) original images in MS-COCO Lin et al. (2014), (b) contrast normalized images, and (c) style normalized images using a pre-trained style transfer network Johnson et al. (2016). The improvement brought by IN remains significant even when all training images are normalized to the same contrast, but are much smaller when all images are (approximately) normalized to the same style. Our results suggest that IN performs a kind of style normalization.

REFERENCES

Tian Qi Chen and Mark Schmidt. Fast patch-based style transfer of arbitrary style. *arXiv*, 2016.

Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *ICLR*, 2017.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv*, 2015.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *JMLR*, 2015.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *ECCV*, 2016.

Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. Demystifying neural style transfer. *arXiv*, 2017.

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 *ECCV*, 2014.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *AAAI*, 2016.

Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016a.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv*, 2016b.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. *arXiv*, 2017.