An Empirical Consistency Evaluation of Image Style Transfer Methods

Bo Zhang **Haoming Zhang**
Department of Mathematics, University of California, San Diego
{b4zhang, haz260}@ucsd.edu

Abstract

Image Style Transfer can be considered as a process of synthesizing the semantic content from a target image while preserving the texture style. Gatys et al., for the first time introduced a deep neural network algorithm that successfully separates image content and style and then recombines to generate a new image of another style based upon an iterative optimization process. Based on the work of Gatys et al., Johnson et al. proposed the method of using perceptual loss functions for training feed-forward networks for image style transfer tasks. Later, Huang et al. presented an approach of arbitrary style transfer in real-time by adopting an adaptive instance normalization(AdaIn) layer that aligns the mean and variance of the content features with those of the style features. However, it's hard to define style, an artistic notion, on a measurable scale. Thus, analyzing and comparing the performances of different methods has always been an open issue. In this study, we believe that a good image style transfer method should be consistent in its result. That is, the result of a direct style transfer from a realism image to an artistic painting should be the same as the result of an indirect process (e.g. a photo first being transferred to the style of Picasso then to Van Gogh). In this study, proposed a new evaluation method based on consistency and compared the performance of the three aforementioned methods.

1.Introduction

Image transformation tasks can be expressed as transfering the style of an image to another while preserving the content. Many classic problems can be framed as an image style transfer problem. Johnson et al. [1] listed several examples of possible application, including denoising, super-resolution, and colorization where the input is a degraded image (noisy, low-resolution, or grayscale) and the output is a high-quality color image. Also, based upon [1], an example from computer vision includes semantic segmentation and depth estimation, where the input is a color image and the output image encodes semantic or geometric information about the scene. However, a fundamental prerequisite of solving the image transformation task has been to find image representations of semantic content that is independent from the representations of style.

Gatys et al. [2] showed that deep neural networks (DNNs) encode not only the semantic content but also the style information. Moreover, Gatys et al. [2], for the first time, demonstrated an approach to successfully separate image content from style. Based on the theory of separating content from style, Gatys et al. [2] demonstrated impressive style transfer results by matching second-order feature statistics representing style and content in convolutional layers of a DNN. Even though the style transfer of [2] is flexible enough to combine the content and style of any arbitrary images, the process is prohibitively slow for industrial applications for the reason of relying on an iterative optimization process. Thus, significant effort has been devoted to accelerating neural style transfer, [1] attempted to train a feed-forward neural networks to replace the optimization process. The feed-forward neural network was trained with *perceptual loss functions* that depend on high-level features from a pre-trained *loss network*. Thus the method [1] is capable of performing stylization in a single pass, which runs in real-time and three orders of magnitude faster than [2]. However, the major limitation of the

feed-forward networks is that each trained model is restricted to a specific style. Thus, to transfer an image to a new style, the feed-forward neural network need to be retrained, still restricting its practical applications. Huang et al. [3] presented an approach of arbitrary style transfer in real time, combining a speed similar to the feed-forward approach [1] and the flexibility of the optimization-based framework [2]. It adopted an adaptive instance normalization (AdaIN) layer, which simply adjusts the mean and variance of the content input to match those of the style input. A decoder network is then learned to generate the final stylized image by inverting the AdaIn output back to the image space.

It seems that the method of Huang et al. [3] is the best in terms of both generating speed and flexibility. However, to measure the performance of a style transfer method, it is not enough to only consider speed and flexibility. As the focus of style transfer is to rendering the semantic content of an image in another style, the content should be preserved during image style transfer process and the change of style should be thorough and independent. Thus we focus our study on comparing the consistency of those three aforementioned methods as a measure of performance.

2. Methods

2.1 Gatys et al. [2]

The core of the method raised by Gatys et al. [2] is to generally separate content from style in natural images. It [2] showed how the generic feature representations learned by high-performing Convolutional Neural Network can be used to independently process and manipulate the content and the style of natural images. Gatys et al. [2] used the feature space provided by a normalised version of the 16 convolutional and 5 pooling layers of the 19-layer VGG network such that the mean activation of each convolutional filter over images and positions is equal to one. And the fully connected layers were then abandoned.

2.1.1 Content Representation

In each layer of the CNN, an input image $\hat{x}$ is encoded by a filter responses to that image. A layer of N_l distinct filters has N_l feature maps each of sizes M_l, where M_l is the height times the width of the feature map. So the responses in a layer l can be stored in a matrix $F^l \in R^{N_l \times M_l}$ where F^l_{ij} is the activation of the i^{th} filter at position j in layer l. So, let $\hat{p}$ and $\hat{x}$ be the original image and the image that is generated, and P^l and F^l their corresponding feature representation in layer l [2]. The squared-error loss between the two feature representation can be defined as:

$$L_{content}(\hat{p},\hat{x},l) = \frac{1}{2}\sum_{i,j}(F^l_{ij} - P^l_{ij})^2$$

The derivative of this loss with respect to the activation in layer l equals

$$\frac{\partial L_{content}}{\partial F^l_{ij}} = (F^l - P^l)_{ij} \; if \; F^l_{ij} > 0 \; else \; 0$$

Thus the gradient with respect to the image $\hat{x}$ can be computed using standard error back-propagation. Thus one can perform gradient descent on a white noise image to minimize the content loss, and thus find an image that matches the content feature responses of the original image. However, the layer l to choose to define the loss function depend on the requirement of user. Along the processing hierarchy of the network, detailed pixel information is lost while the high-level content of the image is preserved. In contrast, reconstructions from low level will explicitly capture detailed information of the image such as pixel values. Thus usually high-level layers are preferred to define the content loss.

2.1.2 Style Representation

The style representation used in Gatys et al. [2] consists of the correlations between the different filter responses. The feature correlations of a certain layer l can be represented by a Gram matrix :

$$G^l_{ij} \in R^{N_l \times N_l}, where \;\; G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}$$

Gatys et al. [2] combine feature correlations of multiple layers to construct the style loss. Let $\hat{a}$ and $\hat{x}$ be the original image and the image that is generated, and A^l and G^l their respective style representation in layer l. The contribution of layer l

to the total loss is then $E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (A_{ij}^l - G_{ij}^l)^2$ and then the total style loss is $L_{style}(\hat{a}, \hat{x}) = \sum_{l=0}^{L} w_l E_l$ where w_l is the weights determining the contribution of each layer to the total style loss. The derivative of E_l with respect to the activation F_{ij}^l is:

$$\frac{\partial E_l}{\partial F_{ij}^l} = \frac{1}{N_l^2 M_l^2} (F^l)^T (A^l - G^l)_{ij} \text{ if } F_{ij}^l > 0 \text{ else } 0$$

Thus the gradients of the total style loss with respect to the generated image $\hat{x}$ can be computed using standard error back-propagation. Thus we can use gradient descent from a white noise image to minimise the mean-squared distance between entries of the Gram matrices from the original image and the Gram matrices of the image to be generated [2].

Thus, to transfer the style of a photogram $\hat{p}$ to the style of an artwork $\hat{a}$, [2] synthesise a new image that simultaneously matches the content representation of $\hat{p}$ and the style representation of $\hat{a}$. Thus the total loss function is defined as:

$$L_{total}(\hat{p}, \hat{a}, \hat{x}) = \alpha L_{content}(\hat{p}, \hat{x}) + \beta L_{style}(\hat{a}, \hat{x})$$

where α and β are the weights determining the strength of reconstruction from content image verses style image. In summary, Gatys et al. [2] jointly minimize the distance of feature representations of a white noise image from the content representation of a photogram in one layer and the style representation of the painting defined on a number of layers of the CNN through gradient descent method.

Below are sample style transfer examples that we generated using method of Gatys et al. [2]:

Figure 1. Example stye transfer results of Gatys et al.

2.2 Johnson et al. [1]

Johnson et al. [1] combined the benefits of feed-forward image transformation tasks and optimization-based methods for image generation by training feed-forward transformation networks with perceptual loss functions. As shown in Figure 2, the system consists of two components: an *image transformation network* f_W and *a loss network ø* used to define several loss functions $l_1, l_2, \ldots, l_k$. The *image transformation network* is trained to transform input images into output images. The loss network pre-trained for image classification is used to define perceptual loss functions that measure perceptual differences in content and style between images [1]. Suppose the *image transformation network* is a deep residual convolutional neural network parameterized by weights W; it transforms input images x into output image $\hat{y}$ via the mapping $\hat{y} = f_W(x)$. Each loss function computes a scalar value $l_i(\hat{y}, y_i)$ measuring the difference between the output image $\hat{y}$ and a target image y_i [1]. Thus the optimal value of W can be determined using stochastic gradient descent in minimizing a weighted combination of loss functions [1]:

$$W^* = argmin_W \, \mathrm{E}_{x,\{y_i\}} \left[\sum_{i=1} \lambda_i l_i(f_W(x), y_i) \right]$$

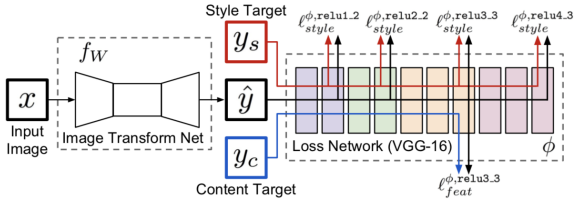

Figure 2. System Overview from Johnson et al. [1]

In Figure 3 we generated some sample style transfer examples using method of Johnson et al. [1].

2.3 Huang et al. [3]

As an approach to transfer arbitrary new style in real-time, the method of Huang et al. [3] adopts an *adaptive instance normalization* (AdaIn) layer in the algorithm. An overview of the style transfer algorithm of Huang et al. is shown in Figure 4. The first few layers of a fixed VGG-19 network was used

to encode the content and style images. Then an AdaIN layer was used to perform style transfer in the

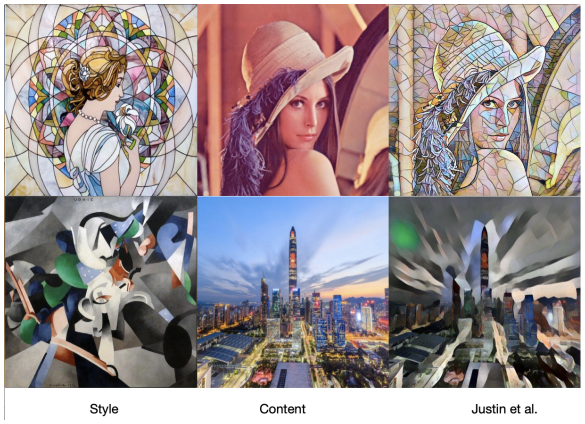

Figure 3. Example style transfer results of Johnson et al.

feature space. Then a decoder is learned to invert the AdaIN output to the image spaces to generate the image of a new style. Huang et al. [3] used the same VGG encoder to compute a content loss L_c and a style loss L_s. The method to compute L_c and L_s will be introduced later.

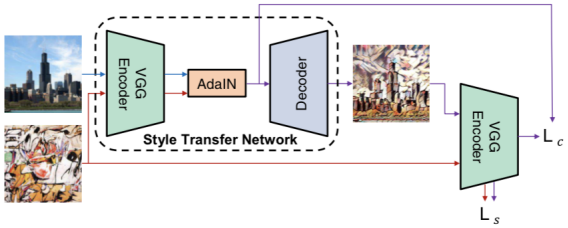

Figure 4. Style transfer algorithm overview from Huang et al.[3].

AdaIn receives a content input x and a style input y and simply aligns the channel-wise mean and variance of x to match those of y.

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

That is, AdaIN rescales the normalized content input with $\sigma(y)$ and shift it with $\mu(y)$. Notice that $\sigma(y)$ and $\mu(y)$ are computed across spatial dimensions independently for each channel and each sample [3]. Thus, AdaIN performs style transfer in the feature space based on the definition of style [3] being represented by the variance and mean calculated across the spatial locations.

Let c be the content image, s an arbitrary style image and T the entire style transfer network. The encoder f is fixed to be the first few layers of a pre-trained VGG-19 of classification object [3]. Let t be the output of AdaIN:

$$t = \mathrm{AdaIN}(f(c), f(s))$$

A randomly generated decoder g is trained to invert t back to the image space, generalizing the stylized image $T(c, s) = g(t)$.

In order to train the decoder, we define the total loss as $L = L_c + \lambda L_s$ where L_c is the content loss in Figure 4 and L_s is the style loss in Figure 4. The content loss L_c is defined as the Euclidean distance between the target features and the features of the output image where the content target is represented as the AdaIN output t [3].

$$L_c = \| f(g(t)) - t \|_2$$

Since the AdaIN layer only transfers the mean and standard deviation of the style features, the style loss only need to match these statistics [3]. Since Huang et al. [3] consider style to be the mean and variance across channel-wise direction, the style loss is defined as:

$$L_c = \sum_{i=1}^{L} \left|\left| \mu(\phi_i(g(t))) - \mu(\phi_i(s)) \right|\right|_2 + \sum_{i=1}^{L} \left|\left| \sigma(\phi_i(g(t))) - \sigma(\phi_i(s)) \right|\right|_2$$

where ϕ_i denotes a layer in VGG-19 used to compute the style loss [3].

In Figure 5 we generated some sample style transfer examples using the method of Huang et al.[3].

2.4 Consistency Evaluation

A good image style transfer method is supposed to be able to separate the semantic content and style of images perfectly and combine the content and style from different sources into a new image without any loss of information. In other words, We expect the result image have exactly the same semantic content

as the corresponding content source and the same style as the style source.

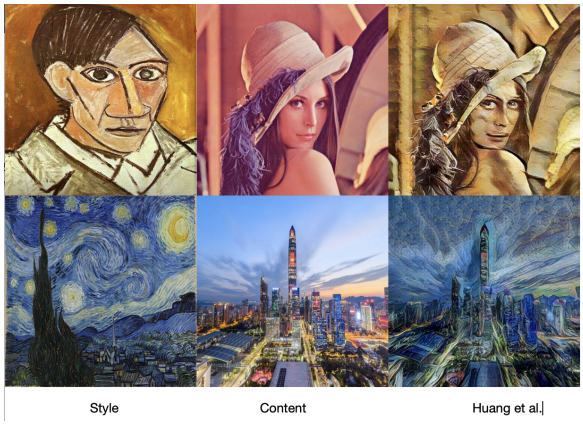

Figure 5. Example style transfer results of Huang et al.

Based on this assumption, we expect a good method to be consistent with different style transfer paths, which means two results with the same content and style setting from two different style transfer paths should be exactly the same.

To evaluate the consistency, we calculate both the content loss and the style loss from [2] between the results from different pathways. Here, smaller losses represent a better performance in consistency.

3. Experiment

In this section, we will conduct experiments to evaluate the consistency of the three methods: the iterative optimization from Gatys et al.[1] [2], the feed-forward networks from Johnson et al.[2] [1] and the arbitrary style transfer in real time from Huang et al.[3] [3].

For each method, we generate four pairs of pictures where each pair consists of the result pictures with the same content and style from a direct transfer path and an indirect path. For example, a direct transfer path is transferring Lenna into the style of Mosaic while the corresponding indirect path is transferring Lenna into the style of Rain Princess and then transferring the previous result into the style of Mosaic. Here, the four pairs of comparison are: (a) Lenna to Rain Princess to Mosaic and Lenna to Mosaic, (b) Lenna to Candy to Udnie and Lenna to Udnie, (c) Shenzhen to Mosaic to Rain Princess and Shenzhen to Rain Princess, (d) Shenzhen to Udnie to Candy and Shenzhen to Candy.

After generating 4 pairs of images for each group, we calculate the content loss and the style loss between each pair and compare the losses among different models. We expect the best method to have the smallest content loss and style loss.

4. Result

In this section, we will discuss the result from previous section and compare the consistency of the three methods. Figure 6, together with Figure 7 displays the difference between direct style transfer and indirect style transfer. It is especially clear in Figure 6 that the image content is preserved better in direct transfer than indirect transfer. This result is reasonable as there is only one round content information loss in direct transfer compared to two round content information loss in indirect style transfer. However, to figure out which method is the most consistent in terms of both content loss and style loss, we resort to Figure 8 to have a quantitative comparison.

If a method is consistent, then the loss of direct style transfer and indirect style transfer should be the same. That is, a perfectly consistent method should have the difference between loss of direct style transfer and indirect style transfer to be 0. The loss contains both style loss and content loss, thus we plotted the difference in loss between direct style transfer and indirect style transfer in terms of both content loss and style loss in Figure 8.

From Figure 8 we can see that Gatys et al.[2] is the most consistent in terms of style loss and Huang et al. [3] is the most consistent in terms of content loss with an exception in the third plot.

[1] We run 1000 iterations of [2] using Gatys' public implementation: https://github.com/leongatys/PytorchNeuralStyleTransfer
[2] We use CompVis' public implementation: https://github.com/CompVis/adaptive-style-transfer
[3] We use the public implementation from pytorch's examples: https://github.com/pytorch/examples/tree/master/fast_neural_style

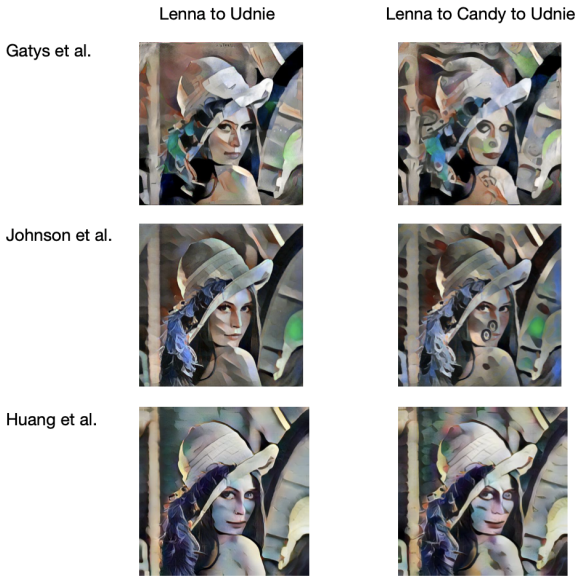

Figure 6. Comparison of the effect of direct style transfer vs indirect style transfer. The direct transfer 'Lenna to Udnie' means directly transfer the Lenna image to the style of Udnie. The indirect transfer 'Lenna to Candy to Udnie' means transfer the Lenna image first to the style of Candy then to the style of Udnie.

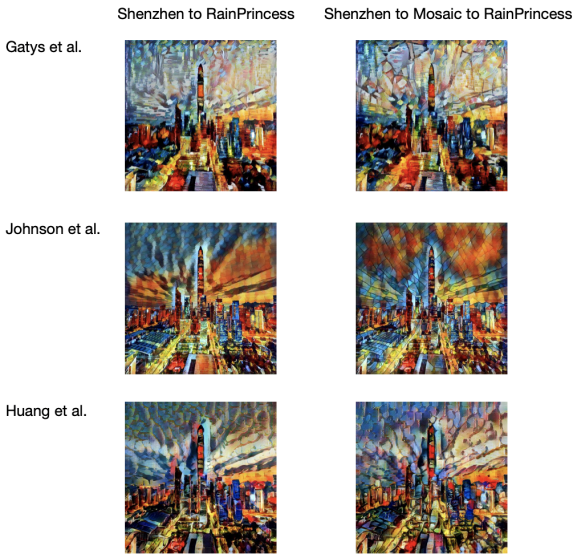

Figure 7. Comparison of the effect of direct style transfer vs indirect style transfer. The direct transfer 'Shenzhen to RainPrincess' means directly transfer the Shenzhen image to the style of RainPrincess. The indirect transfer 'Shenzhen to Mosaic to RainPrincess' means transfer the Shenzhen image first to the style of Mosaic then to the style of RainPrincess.

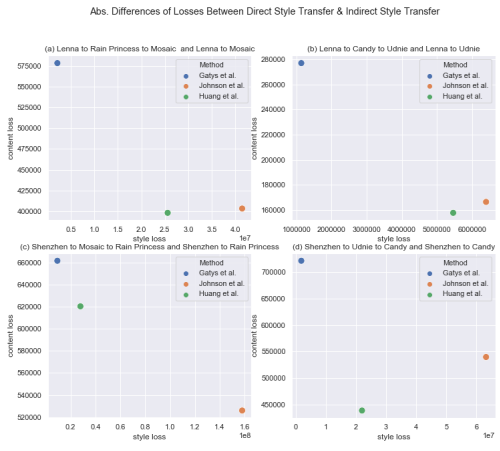

Figure 8. Differences in Content Loss and Style Loss between Direct Style Transfer and Indirect Style Transfer over four different paths of transfer.

5. Discussion and Conclusion

In this paper, we compared the model consistency in style transfer based on methods of Gatys et al. [2] Johnson et al. [1] and Huang et al. [3]. As the purpose of style transfer is to changing the style of the image while preserving the content, the content loss should be controlled as low as possible during consecutive style transfer for the image. Thus we believe that consistency is a reasonable measure of the performance of the style transfer method.

However, as we use the loss function in Gatys et al. [2] to measure and compare the content loss and style loss for all three methods, the method of Gatys et al. [2] may have an advantage in the result of consistency as it [2] is trained to minimize those losses. Also, as we only attempted one approach to measure consistency, our method might not be perfect. We believe there exists other better approaches to resolve this issue.

Acknowledgments

We would like to thank Weijian Xu (COGS 181 TA) for his time and helpful discussions.

References

[1] J. Johnson, A. Alexandre, F. Li. Perceptual losses for real-time style transfer and super-resolution. In ECCV, 2016.

[2] L. A. Gatys, A. S. Ecker, M. Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.

[3] X. Huang, S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In ICCV, 2017.

[4] A. Sanakoyeu, D. Kotovenko, S. Lang, B. Ommer. A style-aware content loss for real-time HD style transfer. In ECCV, 2018.