Arbitrary Style Transfer based on ResNet

Chuan He, Tianran Wang, Zhiyang Lin, Jacob Lin
{ache, tirwang, lzylin, lyclin}@ucdavis.edu

Abstract

Arbitrary style transfer is able to transfer arbitrary artistic styles to content images. Most early feed-forward based models can only achieve efficiency but failed to generalize impressive results facing with unseen styles. Recently a paper has provided a method that tackled this problem with a simple autoencoder network and a whitening and coloring transform. Content and style features are first extracted from the encoder. The whitening and coloring transform then reflect a direct matching of feature covariance of the content features to the style features. Later the transferred features are fed forward into the decoder to obtain the synthesized output images. The method is able to provide arbitrary style transfer because the learning process of the encoder and the decoder is irrelevant to input style images. However, the autoencoder network in the previous work is based on VGG backbone. In our paper, we replace the VGG network with ResNet network to see if skip connections brought by ResNet structure can accelerate the training process and bring better results. We demonstrate the effectiveness of the ResNet in our model by generating comparably good stylized images with comparison to that of the one original model in the same environment. We also show that by using the ResNet, the training process is faster than the original model with a VGG backbone.

1. Introduction

The style transfer task is a branch of computer vision tasks which can be described as an approach of reconstructing or synthesizing texture based on the target image semantic content. The key challenge is how to extract effective representations of the style and then match it in the content image.

This task has been studied for a long time. Many pioneering works have achieved success using traditional image processing methods like histogram matching on linear filter response [1] and non-parametric resampling pixels [2, 3]. These methods typically rely on low-level statistics and often fail to capture semantic structures.

In recent years, the concept of style transfer has been revisited within the concept of deep learning. Gatys et al. [4] provided impressive style transfer results by matching feature statistics in convolutional layers of neural networks. The work of Gatys et al. [4] shows that features correlations (i.e. Gram Matrix) extracted by a trained deep neural network have the remarkable ability to capture image styles. Since then many significant efforts have been made following this framework by minimizing features correlations based on loss functions, through either trained feed-forward network [5-8] or iterative optimization [9-11]. Though feed-forward approaches can be executed efficiently, a major limitation of these approaches is that each network is restricted to a single style. While optimization-based methods can transfer arbitrary styles with pleasing visual quality, the computational cost is often high.

Li et al. [12] proposed a method that achieves generalization, quality and efficiency at the same time. In Li's work, the style transfer task is formulated as image reconstruction process, with the content features being transformed as intermediate layers with regard to the statistics of the style features. The whitening and coloring transform (WCT) is adopted in each intermediate layer to extract content features that exhibit the same statistical characteristics as the style features of the same layer.

The work of Li et al. [12] is so promising that our work is mainly based on it. In the work of Li et al. [12], the VGG-19 network is employed as the encoder to extract features from content images and style images while an inverted VGG-19 network is employed as the decoder. However, in our paper, we replace the VGG-19 network with ResNet-50 and ResNet-101 networks in the encoder and the decoder. Our aim is to find out whether the skip connection characteristic in ResNet networks can further improve the performance of the style transfer task or not. As shown in Figure 1, the single-level style transfer task processes as followings: Content features and style features are first extracted from the upstream encoder. Then a WCT is applied on content features after the encoder such that its covariance matrix matches that of style features. The

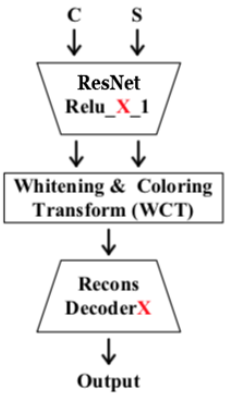

Figure 1: Single-level stylization. The original the VGG structure in the encoder and the decoder is replaced by the ResNet network. With both the encoder and the decoder fixed, and given the content image C and the style image S, our method performs the style transfer through the whitening and coloring transform.

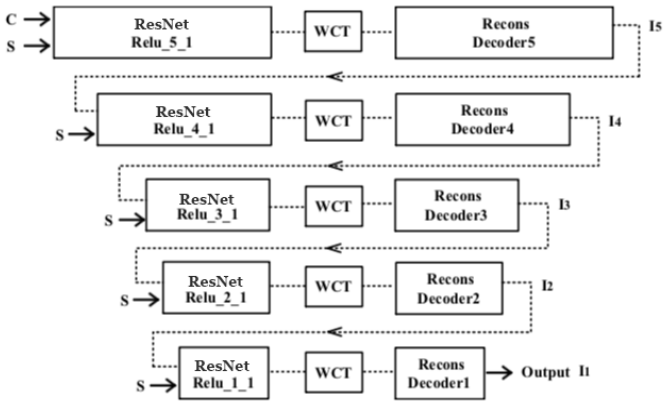

Figure 2: Multi-level stylization. The single-level stylization is extended to multi-level stylization so that the statistics of the style at all levels can matched. The result obtained by matching higher-level statistics of the style is treated as the new content to continue to match lower-level information of the style.

transformed features are then fed forward into the downstream decoder to obtain the stylized image.

Same as the work of Li et al. [12], we also develop a multi-level stylization pipeline, as depicted in Figure 2. Features are extracted from multiple layers and each one goes through a WCT sequentially in each layer. The multi-level pipeline is proved to be able to generate synthesized images with higher visual quality while less computational costs.

Since the overall architecture of our work is similar to the work of Li et al. [12], we inherit the advantage from the previous work. Our model is able to achieve arbitrary style transfer because the learning process of the image reconstruction decoder is irrelevant with both the content image and the style image. Once given a new style image, we only need to extract its feature covariance matrices and apply them to the content features through WCT. Thing is the same when a new content image comes. This is the fundamental difference from existing feed-forward networks that require some learning process of predefined styles. Another contribution of our work is that we use the ResNet instead of the VGG as the backbone network. Experiments show that by introducing the ResNet, the training process is able to get converged quickly and the synthesized result is comparably good with regard to the original one in the same environment. Code is available at https://github.com/Alvinhech/resnet-autoencoder.

2. Related Work

Style Transfer

The problem of style transfer comes from non-photorealistic rendering [13], and is closely related to texture synthesis and transfer [2, 14, 15]. Early methods like histogram matching on linear filter response [16] and non-parametric sampling [2, 3] reply on low-level features and often fails to capture semantic structures. Also, at the early time, existing style transfer methods are mostly example-based. The image analogy method [17] aims to determine the relationship between a pair of images and then apply it to stylize other images. As it is based on finding dense correspondence, analogy-based approaches [3, 18, 19] often require that a pair of image depicts the same type of scene. Therefore, these methods do not scale to the setting of arbitrary style images well.

Gatys et al. [4] for the first time demonstrated impressive style transfer results by matching feature statistics in convolutional layers of a DNN, which is able to perform arbitrary artistic style transfer. In their work, feature correlations (i.e. Gram Matrix) are compared and matched between deep features extracted by a trained network classifier within an iterative optimization network. Since then numerous extensive methods have been put forward to address different aspects including speed [5, 6, 20], quality [21-24], user control [25], diversity [8, 26], semantic understanding [3, 27] and photorealism [28].

Most recently, a number of new methods have been proposed to enable transferring multiple styles using a single network, including a model that conditioned on binary selection units [26], a network that learns a number of new filters for every new style [29], and a novel conditional normalization layer that learns normalization parameters for each style [30]. Our method has its original idea from the work of Chen *et al.* [31] in which the content feature is swapped with the closest style feature locally. This idea is later been applied the following works [32, 33] which turn to learn a general mapping from the style image to style parameters and directly adjust the content feature to match the mean and variance of the style feature. However,

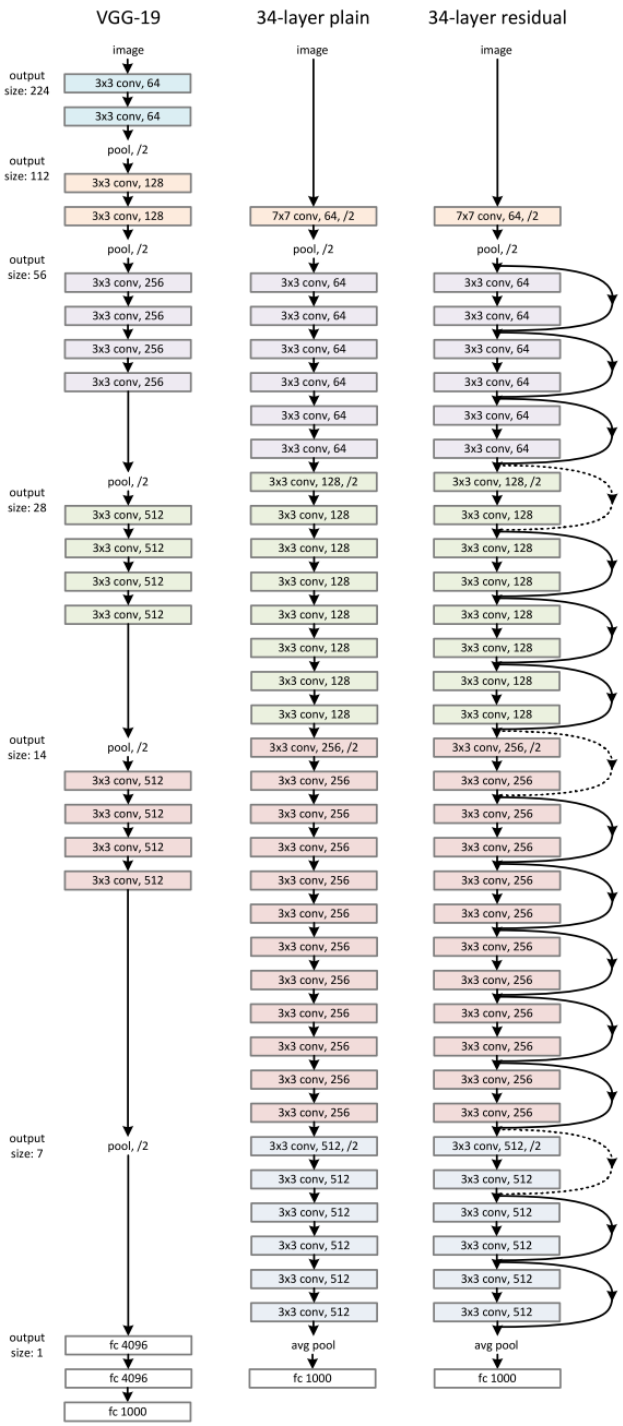

Figure 3: The VGG and the ResNet networks architecture. Left: the VGG-19 model. Middle: a plain network with 34 parameter layers. Right: a residual network with 34 parameter layers. The dotted shortcuts increase dimensions.

the generalization ability of the learned models on unseen styles is still limited.

Autoencoder

The autoencoder is a type of neural network used to learn efficient data coding in an unsupervised manner. The aim of an autoencoder is to learn a representation (encoding) for a set of data, like the features of a content image or style image. Along with the reduction side, a reconstructing side is learnt, where the autoencoder tries to generate from the reduced encoding a representation as close as possible to its original input, hence its name. One advantage of the autoencoder is that it is sometimes possible to generate new data that is similar to the training data but not exactly the same. Recently autoencoder has been used in style transfer tasks [29, 34, 35]. An autoencoder always consists of two parts, the encoder and the decoder. Typically, the decoder is the invert version of the encoder.

Generative Adversarial Networks

The Generative Adversarial Network (GAN), which trains an adversarial generator and discriminator simultaneously, has shown its advantages in the field of image generation. [36-42]. Various improvements to the GAN framework have been proposed, such as conditional GAN [43, 44], multi-stage processing [37, 45], and better training objectives [46, 47]. GANs have also been applied to cross-domain image generation [44, 48-52]. However, the style transfer task is hard to be handled using the GAN network because of the missing ground-truth image pairs. To solve this problem, in the work of Zhang et al. [53], they adopt a loss network that minimizes the perceptual difference of synthesized images with content and style target and provides the supervision of the generative network learning.

3. Technical Approach

In our paper, the style transfer task is tackled as two process, an image reconstruction process and a feature transformation process. The prior process is handled by the autoencoder (an encoder and a decoder) to extract features from content and style images and then invert them back to the RGB space. While the later process is handled by the whitening and coloring transform.

Our models take two images as inputs, one labeled as the content image and another as the style image. They will be processed by the same encoder to extract low-level and

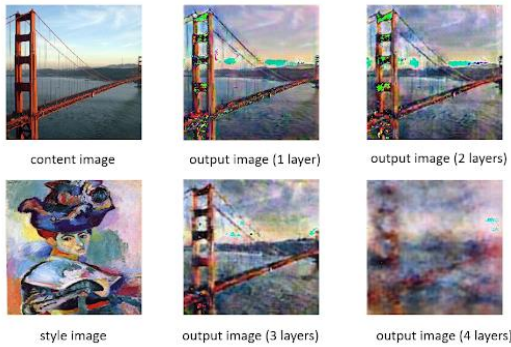

Figure 4: Results of using different numbers of layers.

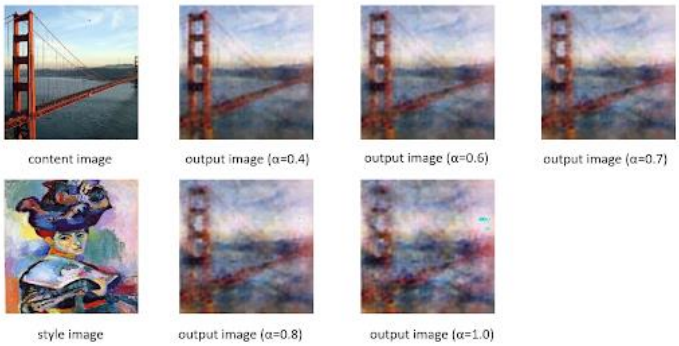

Figure 5: Controlling the stylization on the weight of transfer effects. The larger the alpha is, the more transferred features are used.

high-level features. Then these two feature maps from the encoder will be fed to the WCT to match the statistics of the content feature map to the style feature map and to create a transformed feature map. Afterward, a decoder will transform the mixed feature map back to an image-like form, which completes the style transfer process.

Autoencoder based on ResNet

Our project adopts autoencoders instead of GANs as the general framework for style transfer tasks. We employ the ResNet-50 and ResNet-101 network instead of the VGG-19 network as the encoder. Since the depth of the ResNet-50 network is much deeper than the VGG-19 network, here we only show the difference between architectures of the VGG-19 network and the ResNet-34 network, as shown in Figure 3. The VGG-19 network can be divided into 5 layers with each layer having 2-4 convolutional layers. Although ResNet-34 is deeper, it also can be divided into 5 layers while each layer contains more convolutional layers. In the same way, we can also divide ResNet-50 and ResNet-101 into 5 layers. Thus, the encoder of either the ResNet-50 or the ResNet-101 network contains more layers compared with that of the VGG-19 network. The decoder is the inverted version of encoder thus is symmetrical to the encoder, with the nearest neighbor up sampling layer used for enlarging feature maps, as shown in Figure 1. To evaluate with features extracted at different layers, we select feature maps at five layers of the ResNet-50 and the ResNet-101 network, i.e., Relu_X_1(X=1,2,3,4,5), and train five decoders accordingly.

Whitening and Coloring Transform

Given a pair of content image *Ic* and style image *Is*, we first extract their vectorized ResNet feature maps $Fc \in R^{(C*Hc*Wc)}$ and $Fs \in R^{(C*Hs*Ws)}$ at a certain layer (e.g., Relu_3_1), where *Hc*, *Wc* (*Hs*, *Ws*) are the height and width of the content (style) feature, and *C* is the number of channels. The decoder will reconstruct the original image *Ic* if *Fc* is directly fed into it. But in actual cases, we need to use a whitening and coloring transform to adjust *Fc* with respect to the statistics of *Fs*. The function of WCT is to directly transform the *Fc* to match the covariance matrix of *Fs*. It consists of two steps, the whitening transform step and the coloring transform step.

Whitening transform. We first center *Fc* by subtracting its mean vector *Mc* before whitening. Then we transform *Fc* to *Fc'* linearly as in (1) such that the feature maps are uncorrelated (*Fc' FcT = I*),

$$Fc' = Ec\ Dc^{(-1/2)}\ EcT\ Fc, \quad (1)$$

where *Dc* is a diagonal matrix with the eigenvectors of the covariance matrix $FcFcT \in R^{(C*C)}$, and *Ec* is the corresponding orthogonal matrix of eigenvectors, satisfying that *Fc FcT = Ec Dc EcT*.

Coloring transform. We first center *Fs* by subtracting its mean vector *Ms* and then carry out the coloring transform, which is essentially the inverse operation of the whitening step to transform *Fc'* linearly as in (2) such that we obtain *Fcs'* which has the desired correlations between its feature maps (*Fcs' Fcs'T = Fs FsT*),

$$Fcs' = Es\ Ds^{(1/2)}\ EsT\ Fc', \quad (2)$$

where *Ds* is a diagonal matrix with the eigenvalues of the covariance matrix $FsFsT \in R^{(C*C)}$, and *Es* is the corresponding orthogonal matrix of eigenvectors.
Finally, we re-center the *Fcs'* with the mean vector *Ms* of the style.

We also introduce a control parameter that defines the degree of the style transfer so that the users can choose the balance between stylization and content preservation. After the WCT, we blend the transformed feature *Fcs'* with the content feature *Fc* as in (3) before feeding it to the decoder:

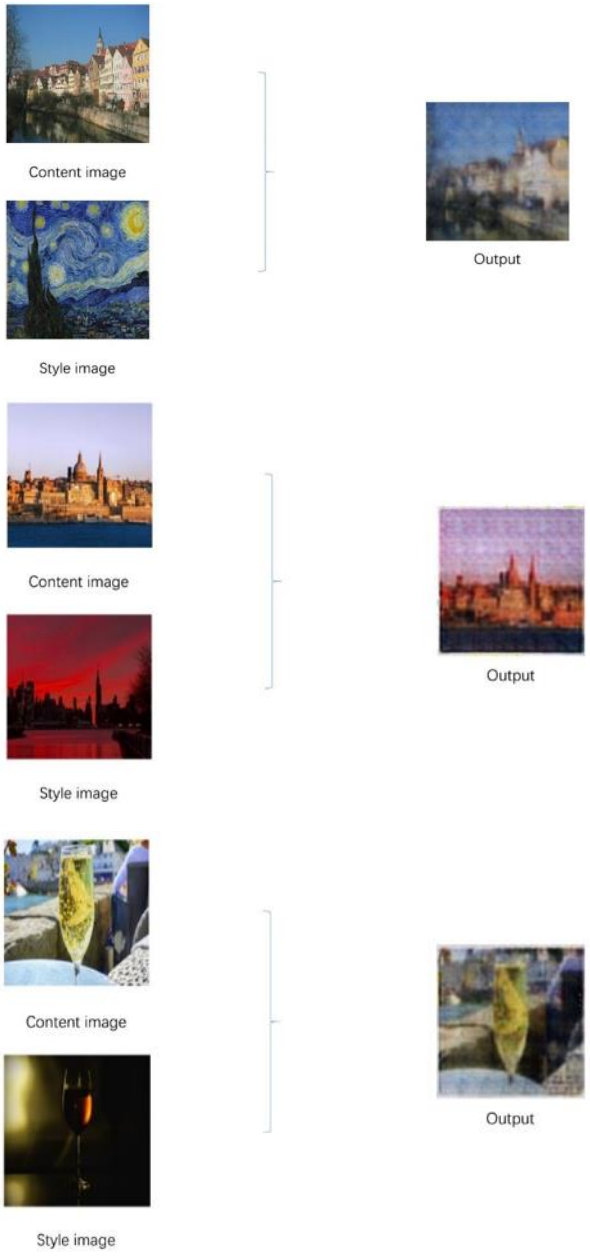

Figure 6: Results from our model based on ResNet-50, here we set alpha=0.8.

$$Fcs' = \alpha Fcs' + (1-\alpha) Fc, \quad (3)$$

where α serves as the style weight for users to control the transfer effect.

Multi-level Coarse-to-Fine Stylization

We adopt multi-level stylization for the model. We use 5 layers to extract both high-level and low-level features. The higher layer features capture more complicated local structures, while lower layer features carry more low-level information like colors and textures.

As shown in Figure 2, we start by applying the WCT on features from Relu_5_1 to obtain a coarse stylized result and regard it as the new content image to further adjust features in lower layers. If we reverse the feature processing order, like in fine-to-coarse order by starting with Relu_1_1, low level information will probably be destroyed after manipulating higher level features.

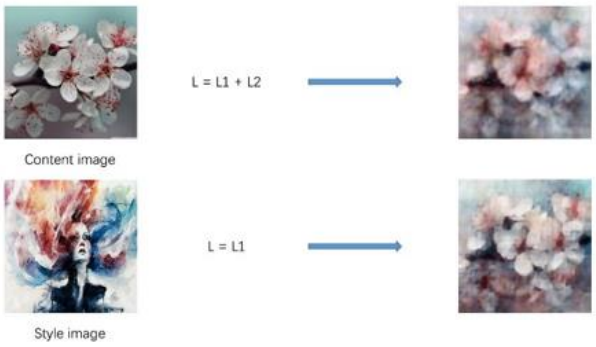

Figure 7: Results from using different loss functions. Using the sum of L1 and L2 as the loss function is able to transfer more style to the synthesized image but on the other hand may introduce blur problems compared with only using the L1.

Loss Function Definition

For the encoder part, we directly use trained components of the ResNet-50 and the ResNet-101 network, because it is well-trained on the COCO dataset. It is already very powerful in extracting features from pictures and there is simply no need to retrain the encoder part. The decoder is the only part we need to train but it cannot be trained alone without the encoder. Since we already have a pre-trained encoder part, we use it to generate ground truth training data based on the COCO dataset. Each training image *I* will be encoded as a feature map *Fi*, and it will be transformed to a new image *I'*. To obtain a good decoder, we want to minimize the difference between *I* and *I'*, expecting the encoded image can be fully reverted back to the original one. We denote the difference between *I* and *I'* as *L1*, but *L1* is not good enough for style transfer tasks. Although pixel information may be almost identical for *I* and *I'*, we want the decoder be able to preserve feature information as well. We encode *I'* again and get another feature map *Fi'*, and want to minimize the difference between *Fi* and *Fi'*, denoted as *L2*. Our loss function *L* is simply the sum of *L1* and *L2* so that the decoder not only decode pixel color information but also decode high-level semantics as well.

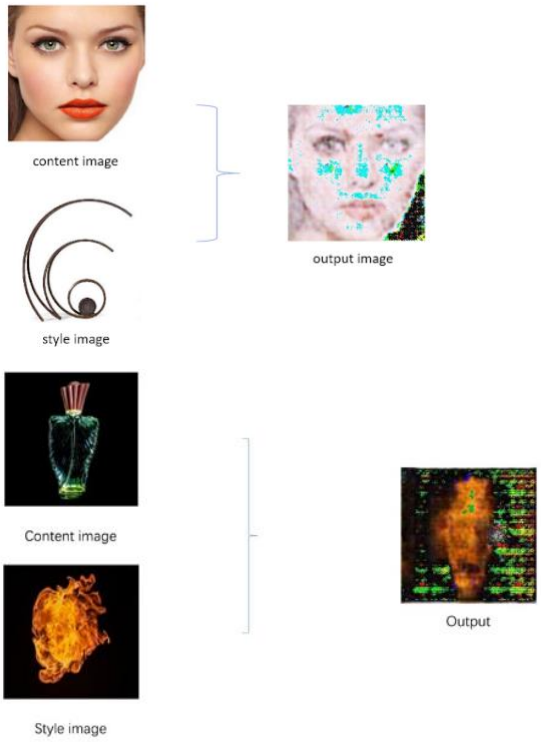

Figure 8: Some failure cases which contains too much noises in the synthesized image.

4. Experiments

We implement the code on the PyTorch platform. We employ one Nvidia Tesla K80 GPU and eight standard CPUs on Google Cloud Platform to train, evaluate and test the model. The database we used is the Microsoft COCO 2017 dataset which contains 118K images for training (18GB) and 5K images for evaluating(1GB). In our experiments, the learning rate is 10^(-4), batch size is 8 and the total epoch is 10. Although Li *et al.* [12] take input images of the size 512 * 512, due to size of images in COCO dataset and the limited resources we have, we train our models with input images of the size 224 * 224. We are unable to compare the performance between the model using ResNet and that using VGG by directly comparing the visual quality of our result images with those shown in the work of Li *et al.* [12] since the difference between the input image sizes. However, as in our experiment, we still find that if we remove all the skip connections in the autoencoder, the training process take longer time and output result is less impressive compared to the one with skip connections.

In Figure 4, the ResNet architecture again shows impressive ability to extract features from images and models with deeper ResNet structure obviously perform much better than those with shallower structure in encoding and recovering features.

In Figure 5, we show the ability of our model to control the strength of transfer effect. The parameter α is free to change even the training process is done. This mechanism allows user to pick the most satisfied result from a number of synthesized images by adjusting the size of α.

Figure 6 shows results of our model based on the ResNet-50 network. The performance is not perfect but we can still that the synthesized image remains the general content information while the color and pattern is in reference with the style image.

However, we also find that there are unexpected noises in our output images. Reasons may be that the model is not fully trained and the input image size is too small. Further training with larger input images may solve this problem. This problem typically emerges when the contrast of the style image is too extreme. For example, in Figure 8, the style image has too sharp contrast and in fact, it does not show a universal style compared to other style images. We believe that the lack of universal style in the style images results in random noises in the output images. We also find that the number of affected pixels is proportional to the α, the ratio hyperparameter of the WCT layer, which to some extent proves our conjecture that the noises are from the decoder part. The decoders are simply unable to recover the encoded image due to the sparsity of feature map from the style image.

5. Conclusions

We extend the arbitrary style transfer work of Li *et al.* [12] by replacing the VGG network with ResNet network. In other words, we add skip connections in each layer of both the down sampling process in encoders and up sampling process in decoders. By unfolding the image generation process via the Resnet autoencoder, we use whitening and coloring transforms to match the statistical distributions and correlations between the intermediate features of content and style. We also present a multi-layer structure to make better performance. Experiments are implemented to evaluate the performance of the neural network, and to explore some factors that would influence the performance of it. As for the unexpected noises, insufficient training and extreme contrast could be the main factors that are causing the problem. Further training and more experiments are needed to determine the cause and solve the problem.

References

[1] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In SIGGRAPH, 1995.

[2] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH*, 2001.

[3] O. Frigo, N. Sabater, J. Delon, and P. Hellier. Split and match: example-based adaptive patch sampling for unsuper- vised style transfer. In *CVPR*, 2016.

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

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414-2423, 2016.

[11] C. Li and M. Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[12] Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., & Yang, M. H. (2017). Universal style transfer via feature transforms. In *Advances in Neural Information Processing Systems* (pp. 386-396).

[13] J. E. Kyprianidis, J. Collomosse, T. Wang, and T. Isenberg. State of the" art: A taxonomy of artistic stylization techniques for images and video. *TVCG*, 2013.

[14] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999.

[15] M. Eladand P.Milanfar. Style transfer via texture synthesis. *arXiv preprint arXiv:1609.03057*, 2016.

[16] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *SIGGRAPH*, 1995.

[17] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *SIGGRAPH*, 2001.

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

[19] Y. Shih, S. Paris, F. Durand, and W. T. Freeman. Data-driven hallucination of different times of day from a single outdoor photo. In *SIGGRAPH*, 2013.

[20] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016.

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

[22] C. Li and M. Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *CVPR*, 2016

[23] 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.

[24] X. Wang, G. Oxholm, D. Zhang, and Y.-F. Wang. Multimodal transfer: A hierarchical deep convolutional neural network for fast artistic style transfer. In *CVPR*, 2017.

[25] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman. Controlling perceptual factors in neural style transfer. In *CVPR*, 2017.

[26] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Diversified texture synthesis with feed-forward networks. In *CVPR*, 2017.

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

[28] F. Luan, S. Paris, E. Shechtman, and K. Bala. Deep photo style transfer. In *CVPR*, 2017.

[29] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua. Stylebank: An explicit representation for neural image style transfer. In *CVPR*, 2017.

[30] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. In *ICLR*, 2017.

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

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

[33] 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.

[34] Zhang, Y., Zhang, Y., & Cai, W. (2018). Separating style and content for generalized style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Vol. 1).

[35] Shen, T., Lei, T., Barzilay, R., & Jaakkola, T. (2017). Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems* (pp. 6830-6841).

[36] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136*, 2016.

[37] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie. Stacked generative adversarial networks. *arXiv*, 2016.

[38] 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.

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

[40] V. A. Sindagiand, V. M. Patel. Generating high-quality crowd density maps using contextual pyramid cnns. In *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017.

[41] H. Zhang, V. Sindagi, and V. M. Patel. Image de-raining using a conditional generative adversarial network. *arXiv preprint arXiv:1701.05957*, 2017.

[42] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *arXiv preprint arXiv:1612.03242*, 2016.

[43] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. In *ICML*, 2016.
[44] P. Isola, J. Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.
[45] E. L. Denton, S. Chintala, R. Fergus, et al. Deep genera- tive image models using a laplacian pyramid of adversarial networks. In *NIPS*, 2015.
[46] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *NIPS*, 2016.
[47] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.
[48] Y. Taigman, A. Polyak, and L. Wolf. Unsupervised cross-domain image generation. In *ICLR*, 2017.
[49] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adapta- tion with generative adversarial networks. *arXiv preprint arXiv:1612.05424*, 2016.
[50] M.-Y. Liu and O. Tuzel. Coupled generative adversarial net-works. In *NIPS*, 2016.
[51] M.-Y. Liu, T. Breuel, and J. Kautz. image-to-image translation networks. *arXiv:1703.00848*, 2017.
[52] T. Kim, M. Cha, H. Kim, J. Lee, and J. Kim. Learning to discover cross-domain relations with generative adversarial networks. *arXiv preprint arXiv:1703.05192*, 2017.
[53] Zhang, H., & Dana, K. (2017). Multi-style generative network for real-time transfer. *arXiv preprint arXiv:1703.06953*.