Style Transfer Using Deep Learning

Zi-Xuan Huang (黃梓軒) Chiou-Shann Fuh (傅楸善)
Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan
Email: r06922100@ntu.edu.tw fuh@csie.ntu.edu.tw

ABSTRACT

With the rapid development of deep learning, such as Convolution Neural Network (CNN), how to properly make an image transfer to another image style become a significant part in many areas which can't be ignored. Here we focus on the factors affecting image style in each different kind of images, and how to transfer the most suitable style feature and contains the original shape.
The algorithm allows us to produce new images of high perceptual quality that combine the content of an arbitrary photograph with the appearance of numerous well known artworks.
Our results provide new insights into the deep image representations learned by Convolutional Neural Networks and demonstrate their potential for high level image synthesis and manipulation.

Keywords: *Deep Learning, Scheduling, PBQP, IPPR, CVGIP 2018.*

1.INTRODUCTION

Transferring the style from one image onto another can be considered a problem of texture transfer. In texture transfer the goal is to synthesize a texture from a source image while constraining the texture synthesis in order to preserve the semantic content of a target image. For texture synthesis there exist a large range of powerful non-parametric algorithms that can synthesize photorealistic natural textures by resampling the pixels of a given source texture [1, 2]. Most previous texture transfer algorithms rely on these nonparametric methods for texture synthesis while using different ways to preserve the structure of the target image. In this work we show how the generic feature representations learned by high-performing Convolutional Neural Networks can be used to independently process and manipulate the content and the style of natural images. Conceptually, it is a texture transfer algorithm that constrains a texture synthesis method by feature representations from state-of-the-art Convolutional Neural Networks. Since the texture model is also based on deep image representations, the style transfer method elegantly reduces to an optimization problem within a single neural network. New images are generated by performing a pre-image search to match feature representations of example images. This general approach has been used before in the context of texture synthesis [3] and to improve the understanding of deep image representations [4]. In fact, style transfer algorithm combines a parametric texture model based on Convolutional Neural Networks [3] with a method to invert their image representations [4].

2 BACKGROUND

In this section, we will simply describe some background of DNN. A deep neural network (DNN) consists of a directed graph of layers that receive, process, and output data. As input data come, it will starting from input layer, after performing some operation, the output data will soon become an input data for next layer. Through a number of layer, the data will arrive at final layer to predict the result. The layer between input layer and final layer is also called hidden layer. As the model become complicated, the number of hidden will also increase. In Figure 1, it is a DNN's model looks like.

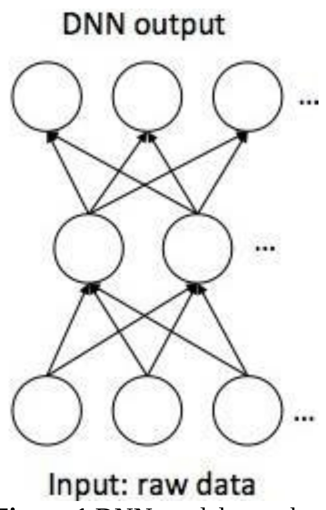


Figure 1 DNN models graph

Hence, we can see DNN models as a directed acyclic model. One well-known class of acyclic feedforward DNNs are Convolutional Neural Networks (CNNs). CNNs normally accept a large matrix or tensor input, such as an RGB image. The input layer of the CNN processes the input tensor, and produces one or more output tensors on its output edge(s). These outputs trigger the execution of subsequent layers.
The output of the CNN is commonly a classification, that is, a weighted distribution of categories — such as dogs or helicopters — that the CNN predicts for the input. Once training is complete, a CNN is stateless; the output is purely a function of the most recent input and the trained, fixed internal weights.
Our job is to use DNN to get content image's feature and style image's feature and combine it.

3. Approach

We choose the VGG [5] network as our convolutional neural networks model, which was trained to perform object recognition and localization [6] and is described extensively in the original work. A given input image is represented as a set of filtered images at each processing stage in the CNN. While the number of different filters increases along the processing hierarchy, the size of the filtered images is reduced by some down sampling mechanism (e.g. max-pooling) leading to a decrease in the total number of units per layer of the network.

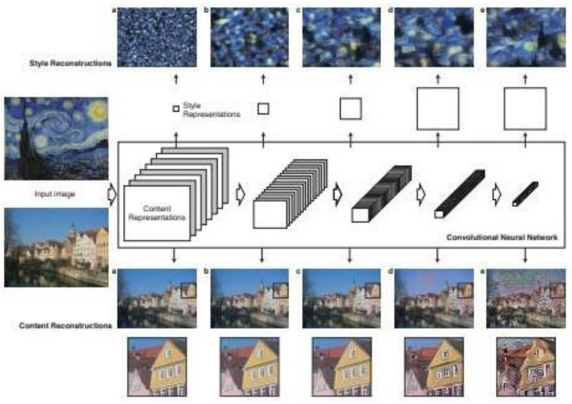


Figure 2: Image representations in a Convolutional Neural Network (CNN).

4. Content representation

Generally each layer in the network defines a non-linear filter bank whose complexity increases with the position of the layer in the network. Hence a given input image x is encoded in each layer of the Convolutional Neural Network by the filter responses to that image. A layer with N_l distinct filters has N_l feature maps each of size M_l , where Ml is the height times the width of the feature map. So the responses in a layer l can be stored in a matrix $F^l \in N_l \times M_l$ where F^l_{ij} is the activation of the i th filter at position j in layer l.

To visualize the image information that is encoded at different layers of the hierarchy one can perform gradient descent on a white noise image to find another image that matches the feature responses of the original image 2 (Fig 1, content reconstructions) [4]. Let p and x be the original image and the image that is generated, and P l and F l their respective feature representation in layer l. We then define the squared-error loss between the two feature representations

$$\mathcal{L}_{\text{content}}(\vec{p}, \vec{x}, l) = \frac{1}{2}\sum_{i,j}\left(F^l_{ij} - P^l_{ij}\right). \qquad (1)$$

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

$$\frac{\partial \mathcal{L}_{\text{content}}}{\partial F^l_{ij}} = \begin{cases} \left(F^l - P^l\right)_{ij} & \text{if } F^l_{ij} > 0 \\ 0 & \text{if } F^l_{ij} < 0\,, \end{cases} \qquad (2)$$

from which the gradient with respect to the image x can be computed using standard error back-propagation (Fig 2, right). Thus we can change the initially random image x until it generates the same response in a certain layer of the Convolutional Neural Network as the original image p.

The higher layers in the network capture the high-level content in terms of objects and their arrangement in the input image but do not constrainthe reconstruction very much. In contrast, reconstructions from the lower layers simply reproduce the exact pixel values of the original image (Fig 1, content reconstructions a–c). We therefore refer to the feature responses in higher layers of the network as the content representation.

5. Style representation

To obtain a representation of the style of an input image, we use a feature space designed to capture texture information [3]. This feature space can be built on top of the filter responses in any layer of the network. It consists of the correlations between the different filter responses, where the expectation is taken over the spatial extent of the feature maps. These feature correlations are given by the Gram matrix $G^l \in N_l \times$ M_l

where G_{ij}^{l} is the inner product between the vectorized feature maps i and j in layer l:

$$G_{ij}^{l} = \sum_{k} F_{ik}^{l} F_{jk}^{l}. \qquad (2)$$

By including the feature correlations of multiple layers, we obtain a stationary, multi-scale representation of the input image, which captures its texture information but not the global arrangement. Again, we can visualize the information captured by these style feature spaces built on different layers of the network by constructing an image that matches the style representation of a given input image (Fig 1, style reconstructions).

Let a and 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

$$E_l = \frac{1}{4N_l^2 M_L^2} \sum_{i,j} (G_{ij}^{l} - A_{ij}^{l}) \qquad (3)$$

and the total style loss is

$$\mathcal{L}_{\text{style}}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l, \qquad (4)$$

where w_l are weighting factors of the contribution of each layer to the total loss.

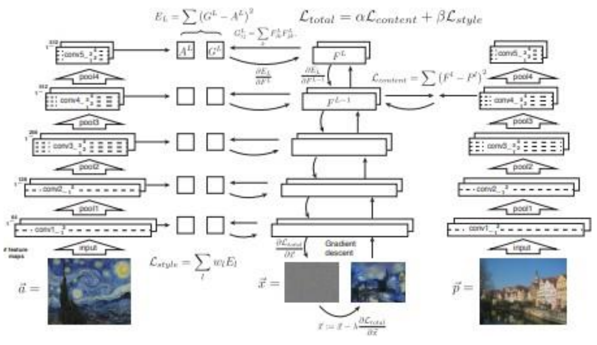

$E_l = \sum (G^l - A^l)^2$
$\mathcal{L}_{total} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style}$
$\mathcal{L}_{content} = \sum (F^l - P^l)^2$
$\mathcal{L}_{style} = \sum_l w_l E_l$
Gradient descent
$\vec{a} =$
$\vec{x} =$
$\vec{p} =$
input
pool1
pool2
pool3
pool4

Figure 3: Style transfer architecture.

6. Style Transfer

To transfer the style of an artwork a onto a photograph p we synthesize a new image that simultaneously matches the content representation of p and the style representation of a. Thus we jointly minimize the distance of the feature representations of a white noise image from the content representation of the photograph in one layer and the style representation of the painting defined on a number of layers of the Convolutional Neural Network. The loss function we minimize is

$$\mathcal{L}_{\text{total}}(\vec{p}, \vec{a}, \vec{x}) = \alpha\mathcal{L}_{\text{content}}(\vec{p}, \vec{x}) + \beta\mathcal{L}_{\text{style}}(\vec{a}, \vec{x}) \qquad (5)$$

where α and β are the weighting factors for content and style reconstruction.

7. Expectation

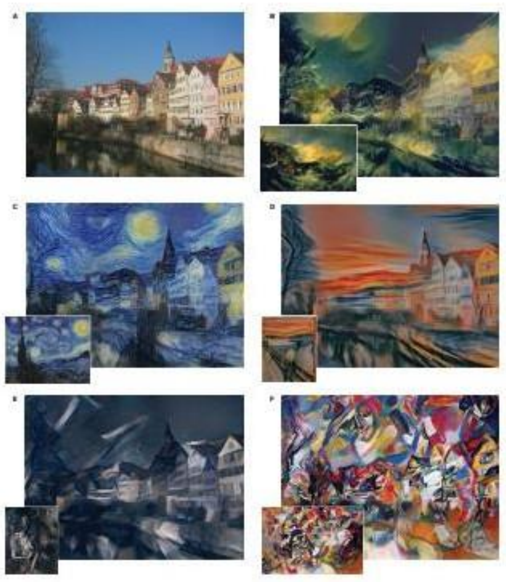
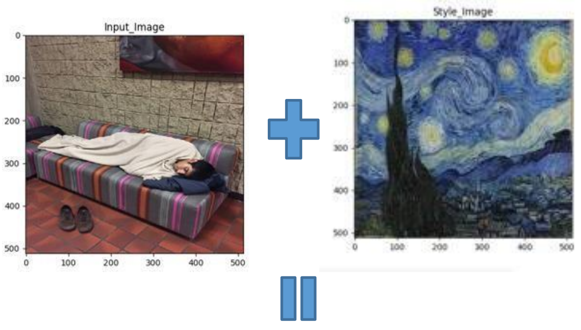


Figure 4: Ours expectation results: Images that combine the content of a photograph with the style of several well-known artworks.

8. Discussion

Here we discuss the parameter $\alpha\beta$ influence in our loss function (5).

First if α=10000 and β=1:



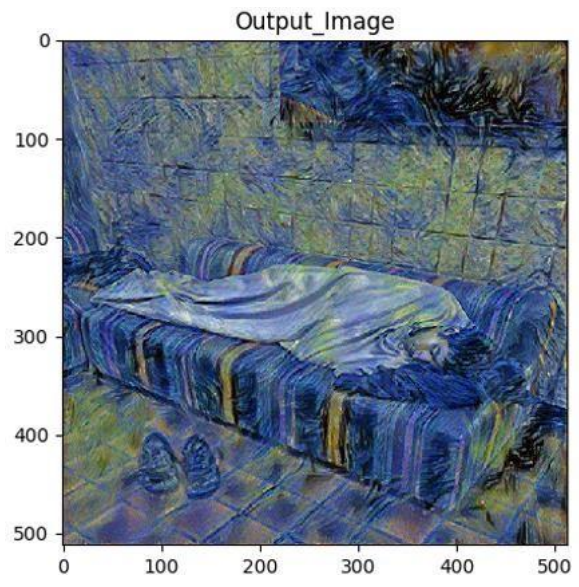


Figure 5: The result output image with parameter α =10000 and β =1.

We can see that it is a successful transfer since the output image is not only contains the man who is sleeping but the style is also change to the style image's style. Next, we reduce the value of α with same content image and style image as input image.

If α=5000 and β=1:

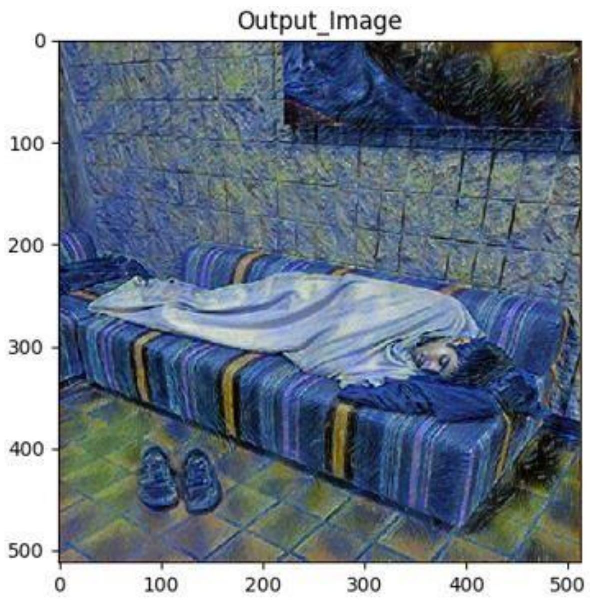


Figure 6: The result output image with parameter α =5000 and β =1.

As we see above, since we reduce the weight of style image, the shape which content image controll is more clearly than first output image. We now reduce the value of α again.

If α=1000 and β=1:

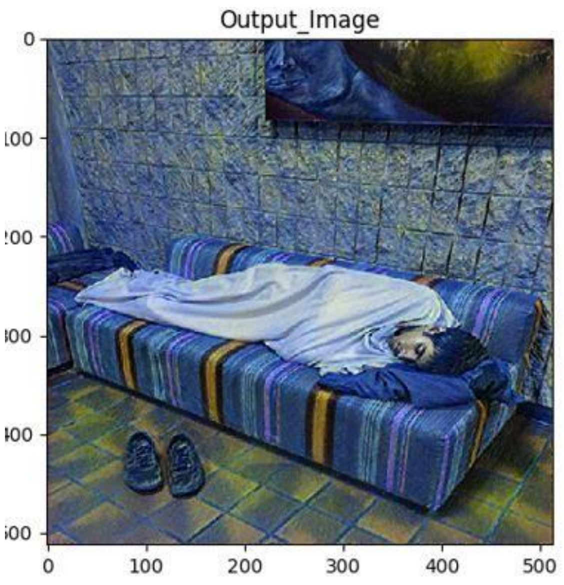


Figure 7: The result output image with parameter α =1000 and β =1.

As we expected, the shape reserved more details, but the cost is that the style transfer is become not obvious anymore.

On the contrary, as we increase the value of α.

If α=50000 and β=1:

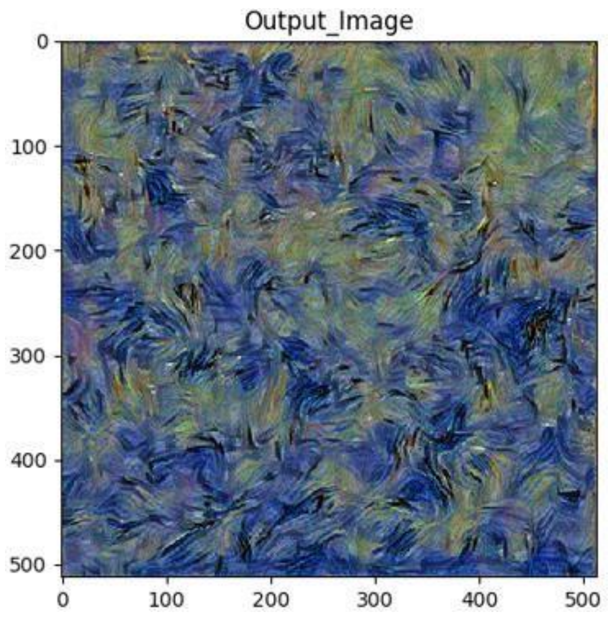


Figure 8: The result output image with parameter α =50000 and β =1.

Since we increase the weight of style image, content image gradually can't hold its shape. Now we decide to increase α again.

If α=100000 and β=1:

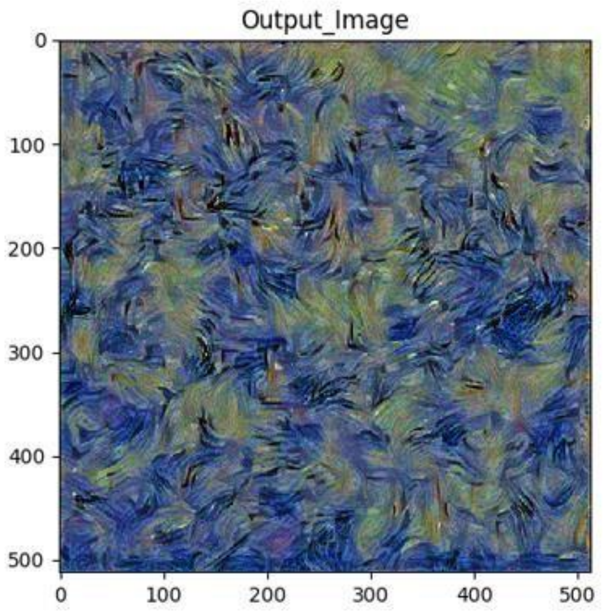
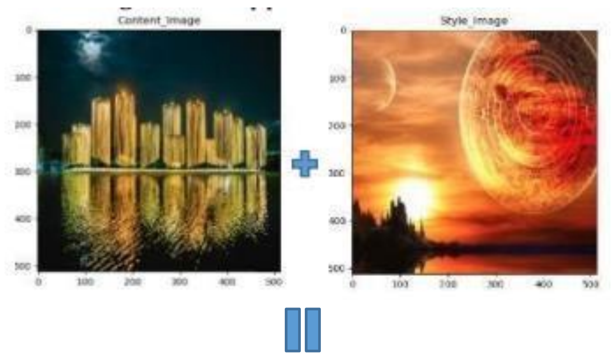


Figure 9: The result output image with parameter α =100000 and β=1.

We can see that we almost cannot recognition what the output it is since its shape is totally lost.

9. Application

Here are some different applications for our tools.



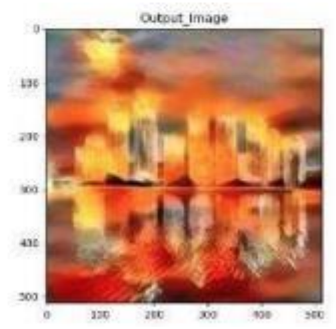


Figure 10: Application 1

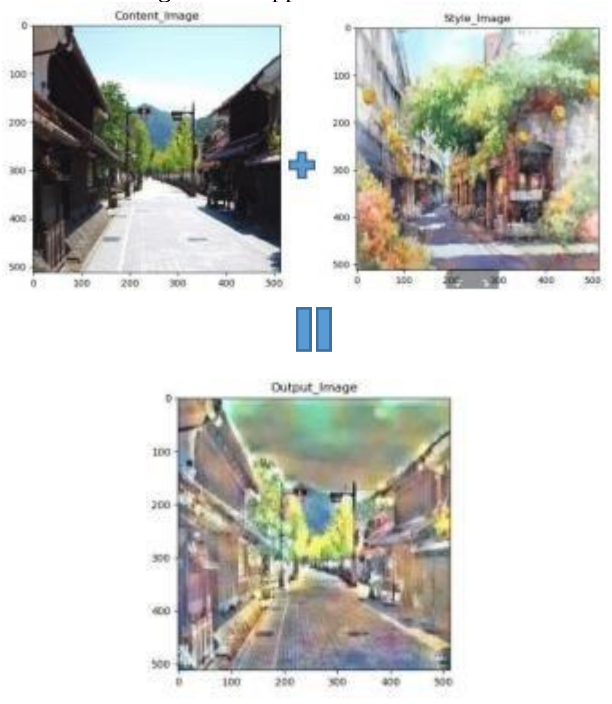


Figure 11: Application 2

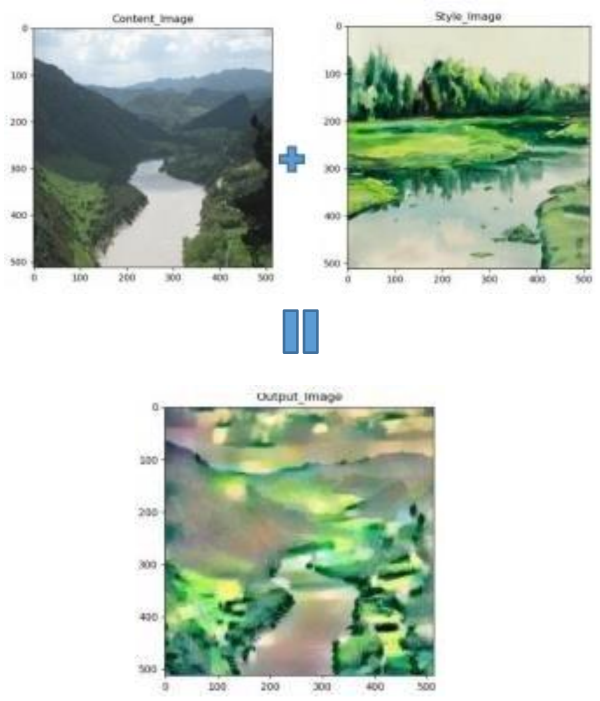


Figure 12: Application 3

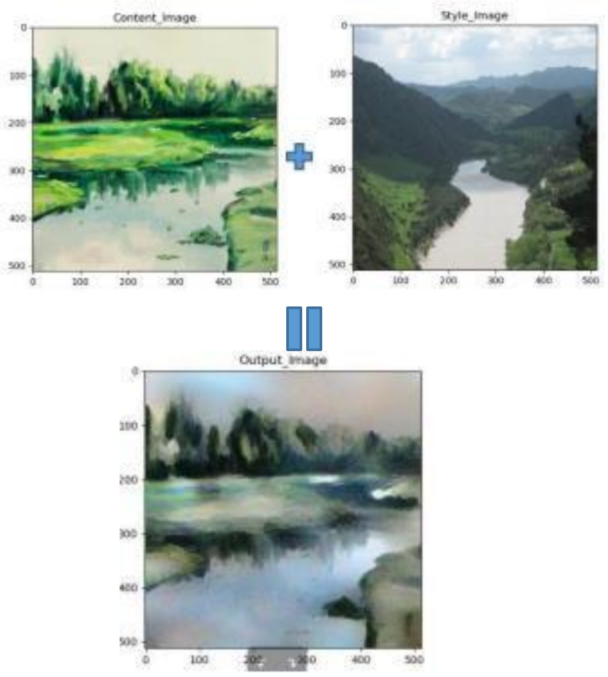


Figure 13: Application 4

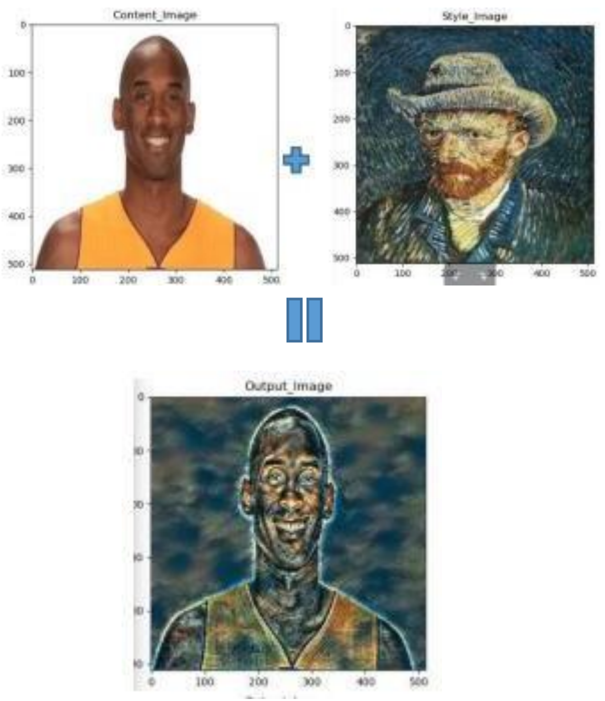
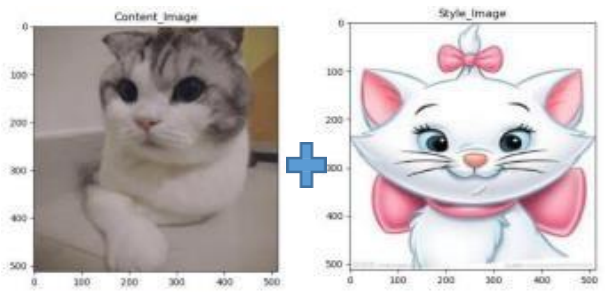


Figure 14: Application 5



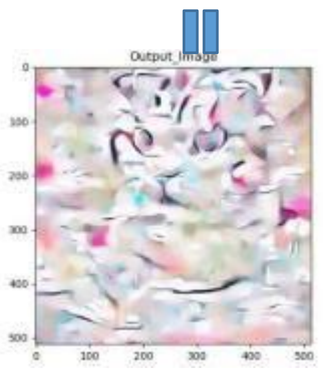


Figure 15: Application 6

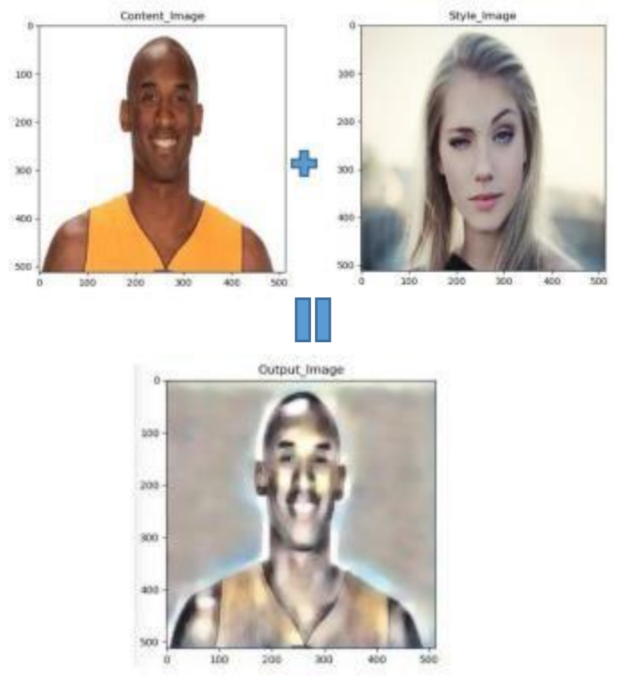


Figure 16: Application 7

10. Conclusion

Here we demonstrate how to use our tool to make one image style transfer to another style image. Although most of applications can transfer perfectly, there still have some applications seems not very well, such like the application in Figure 15 and 16. Since the applications in Figure 10 to 14 is doing traditional style transfer, Figure 15 and 16 has already expected to do some shape fusion. In other words, our tools stills cannot effective doing shape fusion, and this is the main challenge for our future work.

.

10. References

[1] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge, Image Style Transfer Using Convolutional Neural Networks

[2] M. Berning, K. M. Boergens, and M. Helmstaedter. SegEM: Efficient Image Analysis for High-Resolution Connectomics. Neuron, 87(6):1193–1206, Sept. 2015. 2

[3] C. F. Cadieu, H. Hong, D. L. K. Yamins, N. Pinto, D. Ardila, E. A. Solomon, N. J. Majaj, and J. J. DiCarlo. Deep Neural Networks Rival the Representation of Primate IT Cor2421 tex for Core Visual Object Recognition. PLoS Comput Biol, 10(12):e1003963, Dec. 2014. 8

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. arXiv:1412.7062 [cs], Dec. 2014. arXiv: 1412.7062. 2

[5] M. Cimpoi, S. Maji, and A. Vedaldi. Deep filter banks for texture recognition and segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3828–3836, 2015. 2

[6] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. arXiv:1310.1531 [cs], Oct. 2013. arXiv: 1310.1531. 2

[7] A. Efros and T. K. Leung. Texture synthesis by nonparametric sampling. In Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on, volume 2, pages 1033–1038. IEEE, 1999. 1

[8] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pages 341–346. ACM, 2001. 1

[9] D. Eigen and R. Fergus. Predicting Depth, Surface Normals and Semantic Labels With a Common Multi-Scale Convolutional Architecture. pages 2650–2658, 2015. 2

[10] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture Synthesis Using Convolutional Neural Networks. In Advances in Neural Information Processing Systems 28, 2015. 3, 4

[11] U. Guc¸l ¨ u and M. A. J. v. Gerven. Deep Neural Networks ¨ Reveal a Gradient in the Complexity of Neural Representations across the Ventral Stream. The Journal of Neuroscience, 35(27):10005–10014, July 2015. 8

[12] D. J. Heeger and J. R. Bergen. Pyramid-based Texture Analysis/Synthesis. In Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '95, pages 229–238, New York, NY, USA, 1995. ACM. 3

[13] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pages 327–340. ACM, 2001. 1

[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the ACM International Conference on Multimedia, pages 675–678. ACM, 2014. 3

[15] S. Karayev, M. Trentacoste, H. Han, A. Agarwala, T. Darrell, A. Hertzmann, and H. Winnemoeller. Recognizing image style. arXiv preprint arXiv:1311.3715, 2013. 2

[16] S.-M. Khaligh-Razavi and N. Kriegeskorte. Deep Supervised, but Not Unsupervised, Models May Explain IT Cortical Representation. PLoS Comput Biol, 10(11):e1003915, Nov. 2014. 8

[17] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling. Semi-supervised Learning with Deep Generative Models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 3581–3589. Curran Associates, Inc., 2014. 2

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012. 2

[19] M. Kummerer, L. Theis, and M. Bethge. Deep Gaze I: Boost- ¨ ing Saliency Prediction with Feature Maps Trained on ImageNet. In ICLR Workshop, 2015. 2, 8

[20] V. Kwatra, A. Schodl, I. Essa, G. Turk, and A. Bobick. ¨ Graphcut textures: image and video synthesis using graph cuts. In ACM Transactions on Graphics (ToG), volume 22, pages 277–286. ACM, 2003. 1

[21] J. E. Kyprianidis, J. Collomosse, T. Wang, and T. Isenberg. State of the "Art": A Taxonomy of Artistic Stylization Techniques for Images and Video. Visualization and Computer Graphics, IEEE Transactions on, 19(5):866–885, 2013. 8

[22] H. Lee, S. Seo, S. Ryoo, and K. Yoon. Directional Texture Transfer. In Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering, NPAR '10, pages 43–48, New York, NY, USA, 2010. ACM. 1

[23] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. pages 3431–3440, 2015. 2

[24] A. Mahendran and A. Vedaldi. Understanding Deep Image Representations by Inverting Them. arXiv:1412.0035 [cs], Nov. 2014. arXiv: 1412.0035. 3, 6

[25] J. Portilla and E. P. Simoncelli. A Parametric Texture Model Based on Joint Statistics of Complex

Wavelet Coefficients. International Journal of Computer Vision, 40(1):49–70, Oct. 2000. 3, 4

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. arXiv:1409.0575 [cs], Sept. 2014. arXiv: 1409.0575. 3

[27] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. arXiv:1312.6034 [cs], Dec. 2013. 3

[28] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556 [cs], Sept. 2014. arXiv: 1409.1556. 3

[29] J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. Neural computation, 12(6):1247–1283, 2000. 2

[30] L. Wei and M. Levoy. Fast texture synthesis using treestructured vector quantization. In Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pages 479–488. ACM Press/Addison-Wesley Publishing Co., 2000. 1

[31] D. L. K. Yamins, H. Hong, C. F. Cadieu, E. A. Solomon, D. Seibert, and J. J. DiCarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. Proceedings of the National Academy of Sciences, page 201403112,
May 2014. 8 2422

[32] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale boundconstrained optimization. ACM Transactions on Mathematical Software (TOMS), 23(4):550–560, 1997. 6

[33] N. Ashikhmin. Fast texture transfer. IEEE Computer Graphics and Applications, 23(4):38–43, July 2003. 1