Deep Imitator: Image Imitation Using Convolutional Neural Networks

Abstract

(Report of Project 1) Nowadays, although computers make copy and paste of images very easy, image imitation, especially art works imitation is still a high value and hard task. Masterpieces of art works are rare, hard to get and use of them often involves copy right issues. High level imitation usually relies on manual work. In this report, we propose a new convolutional neural network based high level image imitation model. It produces imitations by combine texture features and content features and renders quickly. Image imitation model performs similarly compare to style transfer models, because they both separate content and style from the original image and render them. The proposed image imitation model is unique in processing separated information and also reduce memory space and processing time. It uses external storage to explicitly store and read style information. It uses a scale adaptive method, allows style features to be delivered at different level. It uses an additive style loss, allows multiple styles to be imitated to one picture at low computational and memory cost. This model is inspired by 2016 CVPR paper Image Style Transfer Using Convolutional Neural Networks.

1 Introduction

Image imitation using computers is a high value and hard task. Image imitation is a challenging task for computers, because it requires the model to do well on 1) separate content and style from the original image. 2) faithfully render the content. 3) render the image in close style. Because of this, image imitation tasks are similar to image style transform in some extent.

To generally separate content from style in natural images is still an extremely difficult problem. However, the recent advance of Deep Convolutional Neural Networks has produced powerful computer vision systems that learn to extract high-level semantic information from natural images. It was shown that Convolutional Neural Networks trained with sufficient labeled data on specific tasks such as object recognition learn to extract high-level image content in generic feature representations that generalize across datasets and even to other visual information processing tasks, including texture recognition and artistic style classification.

In 2016 CVPR paper Image Style Transfer Using Convolutional Neural Networks, Gatys et.al. proposed a image style transfer model that extracts content information and style information from images. The content information is directly represented by feature maps in high layer in CNN, style information is retrieved by self-multiply the vectorized feature maps in lower layers in CNN. Our work also adopted this method to extract content and style information.

Besides rendering image feature in different levels of the convolutional neural network, we want a good image imitation model to be more functional and more efficient. Unlike most of current encoding decoding based or level to level methods, This model use an outside image style database to explicitly store image style feature at different level of the convolutional neural network. When the model needs style information to use on the output image, it can directly load it from outside style database without laborious computation. In style transfer methods, especially level to level style transfer methods, style information are transferred at the corresponding level. Based on previous understanding of the layers in convolutional neural networks, the layers in high position are more likely to reproduce content features, while layers in lower positions are more likely to produce style or texture features of the image. Because image size varies, some information stored in high semantic level of on network may corresponds to lower semantic level in the output network, or vice versa. This model proposed a scale adjustable scheme for adjust style information to wanted level. In current image transfer models, if multiple styles need to be transferred into in output image, it either requires run multiple times in same image, or needs to store multiple networks in graphic memory to run it at once. The proposed model adopts an addible style lose. If incorporate multiple styles is needed, only add the style matrix together and the model can deliver the result. We believe this will save a lot of computational and memory resources.

Comparing with existing method in dealing high level and low level feature in style trans form methods, the proposed model is unique in

i. It uses an outside image style database to explicitly store image style feature at different level of the convolutional neural network. This enables our network to completely decouple styles from the content after learning.
ii. It adapts a scale adjustable scheme for adjust style information to wanted level.
iii. It adapts an addible style lose. When incorporate multiple styles is needed, only add the style matrix together and the model can deliver the result.

The following part of the paper is organized as follows. Part 2 will be related work. In section 3 is the illustration of the proposed image imitation model. Section 4 is the imitation results of the proposed model. In part 5 will be conclusion.

2 Related Works

Image imitation is very close to image style transfer and texture synthesis. In [2] and [3] it offers a description of current style transfer works.

For the style transfer problem. Traditional image stylization works mainly focus on texture synthesis based on low-level features, which use non parametric sampling of pixels or patches in given source texture images [14, 22, 13] or stroke databases [32, 21]. Their extension to video mostly uses optical flow to constrain the temporal coherence of sampling [5, 20, 33]. A comprehensive survey can be found in [27]. Recently, with the development of deep learning, using neural networks for stylization has become an active topic. Gatys et al. [18] first propose using pre-trained Deep Convolutional Neural Networks (CNN) for image stylization. It generates more impressive results compared to traditional methods because CNN provides more semantic representations of styles. To further improve transfer quality, different complementary schemes have been proposed, including face constraints [38], Markov Random Field (MRF) prior [29], user guidance [9] and controls [19]. Unfortunately, these methods based on an iterative optimization are computationally expensive in run-time, which imposes a big limitation in real applications. To make the run-time more efficient, some work directly learn a feed-forward generative network for a specific style [25, 40, 30] or multiple styles [10, 12, 31] which are hundreds of times faster than optimization-based methods. [3]

Gatys et al. [16] for the first time demonstrated impressive style transfer results by matching feature statistics in convolutional layers of a DNN. Recently, several improvements to [16] have been proposed. Li and Wand [30] introduced a framework based on markov random field (MRF) in the deep feature space to enforce local patterns. Gatys et al. [17] proposed ways to control the color preservation, the spatial location, and the scale of style transfer. The framework of Gatys et al. [16] is based on a slow optimization process that iteratively updates the image to minimize a content loss and a style loss computed by a loss network. It can take minutes to converge even with modern GPUs. On-device processing in mobile applications is therefore too slow to be practical. A common workaround is to replace the optimization process with a feed-forward neural network that is trained to minimize the same objective [24, 51, 31]. These feed-forward style transfer approaches are about three orders of magnitude faster than the optimization-based alternative, opening the door to realtime applications. Wang et al. [53] enhanced the granularity of feed-forward style transfer with a multi-resolution architecture. Ulyanov et al. [52] proposed ways to improve the quality and diversity of the generated samples. However, the above feed-forward methods are limited in the sense that each network is tied to a fixed style. To address this problem, Dumoulin et al. [11] introduced a single network that is able to encode 32 styles and their interpolations. Concurrent to our work, Li et al. [32] proposed a feed-forward architecture that can synthesize up to 300 textures and transfer 16 styles. Still, the two methods above cannot adapt to arbitrary styles that are not observed during training.[2]

3 Image Imitation

To extract and represent image features, a pre-trained model is used. VGG-19 is chosen to represent image features in this report. The VGG-19 model is publicly available at https://www.vlfeat.org/matconvnet/pretrained/

3.1 Content representation [1]

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 Nl distinct filters has Nl feature maps each of size Ml, 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 Fl ∈ RNl×Ml where Fl ij is the activation of the ith 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. Let p~ and ~x be the original image and the image that is generated, and Pl and Fl their respective feature representation in layer l. We then define the squared-error loss between the two feature representations

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

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. 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~.

When Convolutional Neural Networks are trained on object recognition, they develop a representation of the image that makes object information increasingly explicit along the processing hierarchy. Therefore, along the processing hierarchy of the network, the input image is transformed into representations that are increasingly sensitive to the actual content of the image, but become relatively invariant to its precise appearance. Thus, higher layers in the network capture the high-level content in terms of objects and their arrangement in the input image but do not constrain the exact pixel values of the reconstruction very much. In contrast, reconstructions from the lower layers simply reproduce the exact pixel values of the original image. We therefore refer to the feature responses in higher layers of the network as the content representation.

2.2 Style representation [1]

To obtain a representation of the style of an input image, we use a feature space designed to capture texture information. 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 Gl ∈ RNl×Nl, where Glij is the inner product between the vectorized feature maps i and j in layer l:

$$G_{ij}^{l} = \sum_{k} F_{ik}^{l} F_{jk}^{l}. \tag{3}$$

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.

The Gram matrix is used to be stored in the external style database. In the process of computing style loss, the model can load the style Gram matrix and compute loss directely from it.

Again, we can visualise 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). This is done by using gradient descent from a white noise image to minimise the mean-squared distance between the entries of the Gram matrices from the original image and the Gram matrices of the image to be generated.

Let ~a and ~x be the original image and the image that is generated, and Al and Gl their respective style representation in layer l. The contribution of layer l to the total loss is then

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^{l} - A_{ij}^{l}\right)^2 \tag{4}$$

and the total style loss is

$$\mathcal{L}_{\text{style}}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l, \tag{5}$$

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

When transfer multiple style information is needed, compute the algorithmic average of the Gram matrix and use it as the Gram matrix in (4), which achieves the same loss as adding multiple loss together in the gradient computation pross.

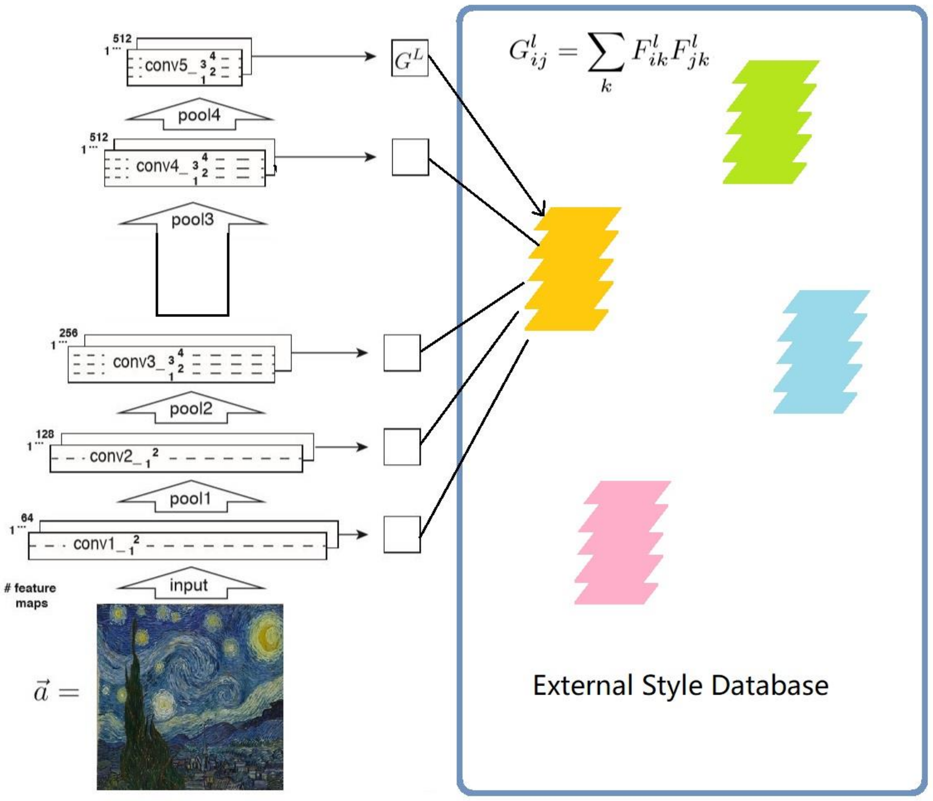
The derivative of El with respect to the activations in layer l can be computed analytically:

$$\frac{\partial E_l}{\partial F_{ij}^{l}} = \begin{cases} \frac{1}{N_l^2 M_l^2} \left(\left(F^l\right)^{\mathrm{T}} \left(G^l - A^l\right)\right)_{ji} & \text{if } F_{ij}^{l} > 0 \\ 0 & \text{if } F_{ij}^{l} < 0 \,. \end{cases} \tag{6}$$

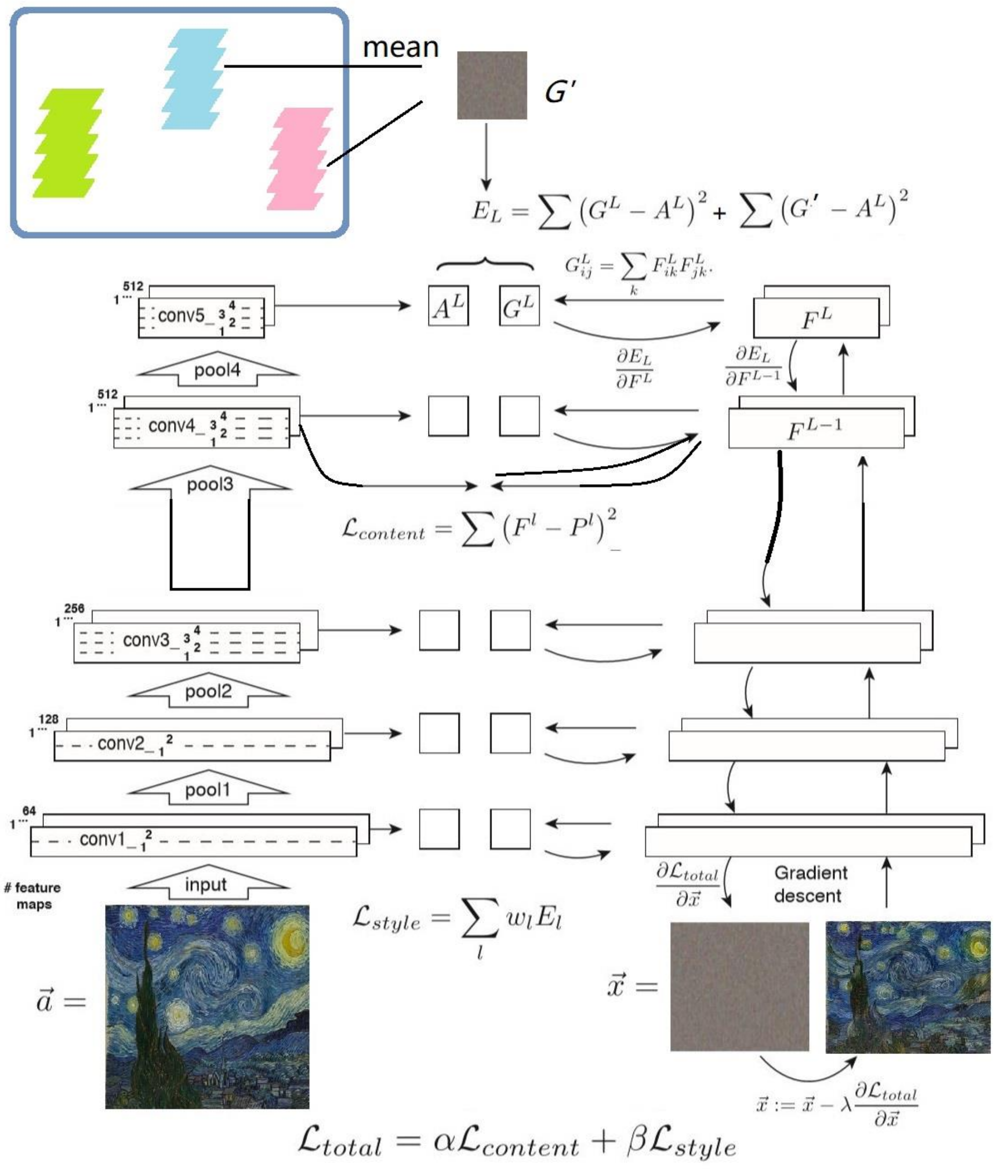
The gradients of El with respect to the pixel values ~x can be readily computed using standard error back-propagation.

3.3 Image Imitation Model

To make imitation of an artwork, we transfer both content features and style features from original artwork a to the output image x. We jointly minimize the content loss and style loss using the loss function mentioned above. Losses are computed at designated layer of the convolutional neural network, and we assign a loss parameter for each of the two losses. The model structure is demonstrated as following:



The external style database is built upon the extracted Gram matrix.


mean
G'
$E_L = \sum (G^L - A^L)^2 + \sum (G' - A^L)^2$
$G^L_{ij} = \sum_k F^L_{ik} F^L_{jk}$.
A^L
G^L
F^L
$\frac{\partial E_L}{\partial F^L}$
$\frac{\partial E_L}{\partial F^{L-1}}$
F^{L-1}
conv5_1 2 3 4
512
pool4
conv4_1 2 3 4
pool3
$\mathcal{L}_{content} = \sum (F^l - P^l)^2$
conv3_1 2 3 4
256
pool2
conv2_1 2
128
pool1
conv1_1 2
64
input
# feature maps
$\frac{\partial \mathcal{L}_{total}}{\partial \vec{x}}$
Gradient descent
$\mathcal{L}_{style} = \sum_l w_l E_l$
$\vec{a} =$
$\vec{x} =$
$\vec{x} := \vec{x} - \lambda \frac{\partial \mathcal{L}_{total}}{\partial \vec{x}}$
$\mathcal{L}_{total} = \alpha \mathcal{L}_{content} + \beta \mathcal{L}_{style}$

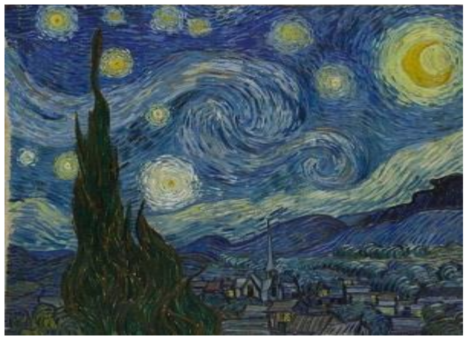
If the style Gram matrix of the input image is already in the external style database, then we no need to compute it from the original image. Only use G' instead. If more than one image's style is needed, simply add the target Gram matrixes in external style database to build G'.

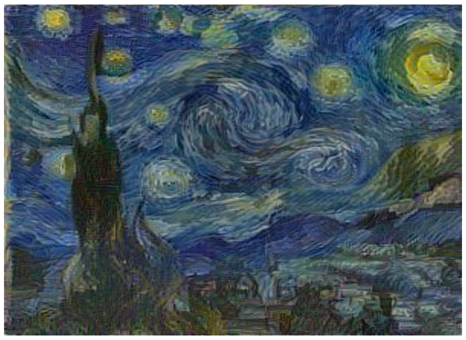
4 Results

4.1 Single Image Imitation

To demonstrate the results of the proposed image imitation model, we tested it on multiple artworks. The parameter of content loss is set to 0.005, and the parameter of style loss is set to 5, which are empirically good for image imitation. Content loss is computed on layer 'relu4_2', Style losses are computed on 'relu1_1', 'relu2_1', 'relu3_1', 'relu4_1', 'relu4_2', 'relu5_1'. Iterations are set to 5000. Run with TensorFlow and GTX 1050Ti for 7 minutes.

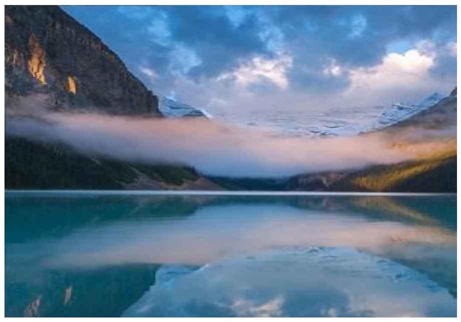
The Starry Night

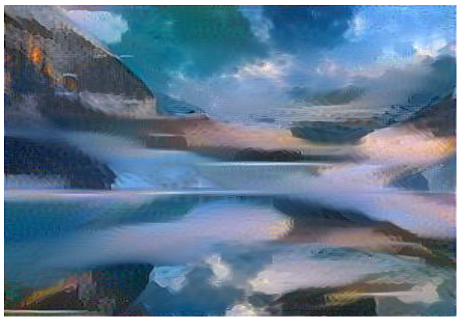


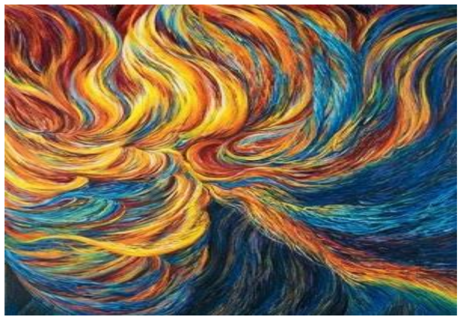


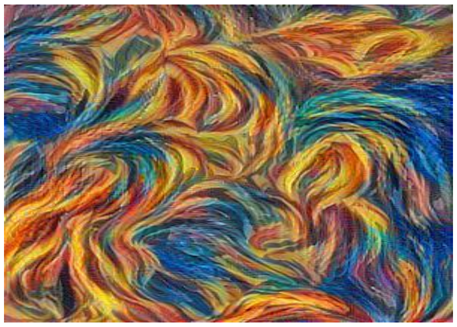
Original

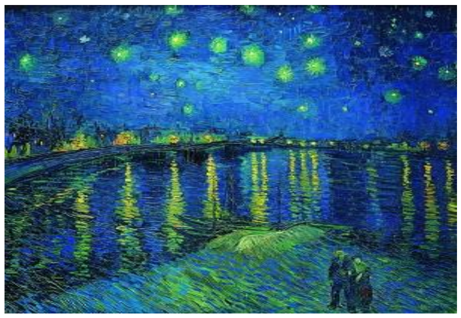
Imitation

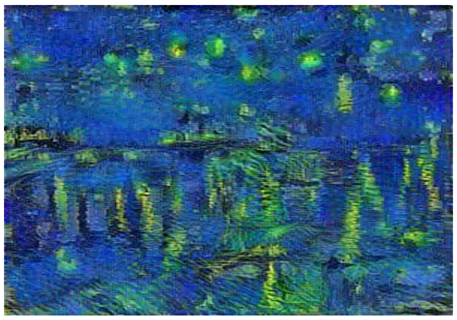






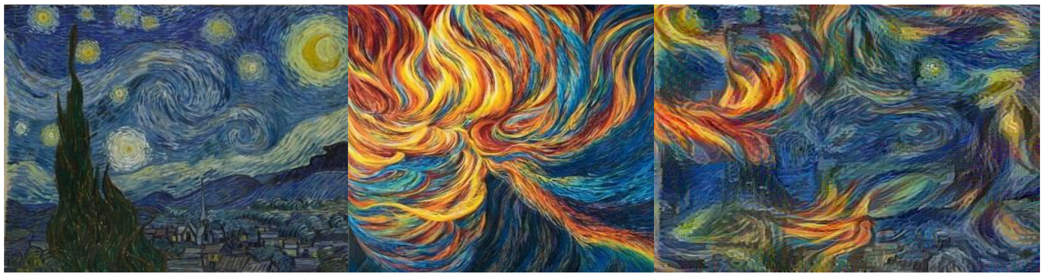




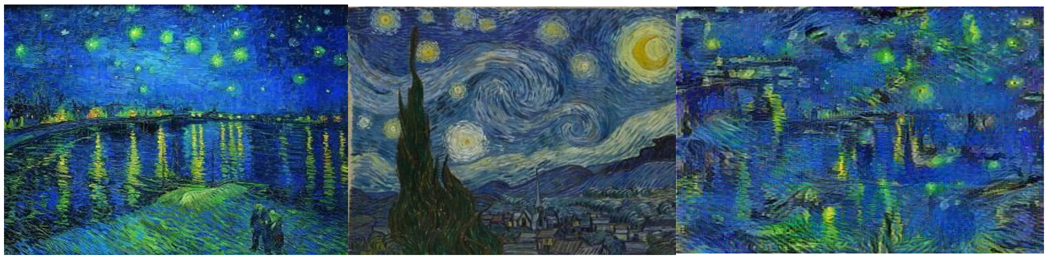


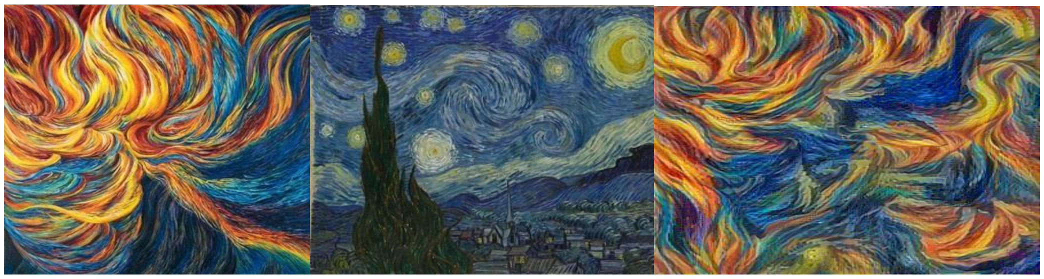
4.1 Multi Style Image Imitation

To show the ability of the proposed image imitation model on multi style image imitation, we tested it on multiple artworks. The parameter of content loss is set to 0.005, and the parameter of input image style loss is set to 5, style loss from other image is set to 2, which are empirically good for image imitation. Content loss is computed on layer 'relu4_2', Style losses are computed on 'relu1_1', 'relu2_1', 'relu3_1', 'relu4_1', 'relu4_2', 'relu5_1'. Iterations are set to 5000. Run with TensorFlow and GTX 1050Ti.



Original Additional Style Result





4 Conclusion

In this paper, we propose a novel method for image imitation. It uses an outside image style database to explicitly store image style feature at different level of the convolutional neural network. The addible style loss also makes it easier to do multiple style imitation. It may inspire further understanding of style and content representative in CNN networks.

References

[1] Leon A. Gatys &Alexander S. Ecker (2016) Image Style Transfer Using Convolutional Neural Networks CVPR 2016 (2415 - 2423)

[2] Chen Dongdong et.al. (2017) An Explicit Representation for Neural Image Style Transfer CVPR 2017

[3] Chen Dongdong .et.al Coherent Online Video Style Transfer ICCV 2017

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

[13] A. A. Efros and T. K. Leung. Texture synthesis by nonparametric sampling. In ICCV, 1999.

[14] M. Elad and P. Milanfar. Style-transfer via texture-synthesis. arXiv preprint arXiv:1609.03057, 2016

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

[32] P. Litwinowicz. Processing images and video for an impressionist effect. In Proc. ACM SIGGRAPH, pages 407–414. ACM, 1997.

[5] A. Bousseau, F. Neyret, J. Thollot, and D. Salesin. Video watercolorization using bidirectional texture advection. In ACM Trans. Graph. (Proc. of SIGGRAPH), volume 26, page 104. ACM, 2007.

[20] J. Hays and I. Essa. Image and video based painterly animation. In Proc. of NPAR, pages 113–120. ACM, 2004.

[33] J. Lu, P. V. Sander, and A. Finkelstein. Interactive painterly stylization of images, videos and 3d animations. In Proc. of I3D, pages 127–134. ACM, 2010.

[27] J. E. Kyprianidis, J. Collomosse, T. Wang, and T. Isenberg. State of the" art: A taxonomy of artistic stylization techniques for images and video. IEEE Trans. on Visualization and Computer Graphics, 19(5):866–885, 2013.

[16] 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.

[17] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. Science, 313(5786):504 – 507, 2006.

[30] J. Malik, S. Belongie, J. Shi, and T. Leung. Textons, contours and regions: Cue integration in image segmentation. In Proceedings of the 7th IEEE International Conference on Computer Vision, volume 2, pages 918–925. IEEE, 1999

[35] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. arXiv preprint arXiv:1611.00850, 2016.