Artistic Neural Style Transfer

Shashikanth Srinath, Han Yin, Paras Verma

***Abstract*—Art style is an important aspect of images. The same object under different art style affects people's impression differently. For many decades, artists have been creating paintings that have complex structures,though some recognizable pattern can be observed. Researchers have been working from last 60 years to develop systems with the property of visual perception similar to that of humans. The development of Convolutional Neural Network makes it possible for machines to comprehend the human perceptual term "art style". Although, significant progress have been seen in the field of face and object recognition, Neural style transfer(NST) is one such technique which is still in progress. Style generator CNNs are auto-encoders that takes an input image and synthesize it with an art style from else where. The goal of this work is to build a CNN model which meets the need of real world applications: fast but lightweight.**

***Keywords*—Neural Style Transfer**

I. INTRODUCTION

Image Stylization is being studied for last two decades under the field of non-photorealistic rendering. Earlier, image style transfer was carried out using image analogy, a machine learning technique. System based on image analogy uses a pair of images, A and A', for learning process and then another image B is fed to the system to generate an artistic analogy B' of B. Lack of such training data has led to emergence of new techniques and Neural Style Transfer (NST) is one such technique that is gaining popularity.

NST, based on Deep Neural Networks, is a process of adding an artistic style to an image while preserving its content. It is an optimization technique that takes as input 3 images content image, style image and input image which is styled using style image while preserving the contents of content image. Fig 1 is an example of content image and style image. Fig 2 is the corresponding result of NST on these two images.

Convolutional Neural Networks (CNN), a class of Deep Neural Networks, are considered to work most efficiently for image processing tasks. CNN process input information in a feed forward manner.

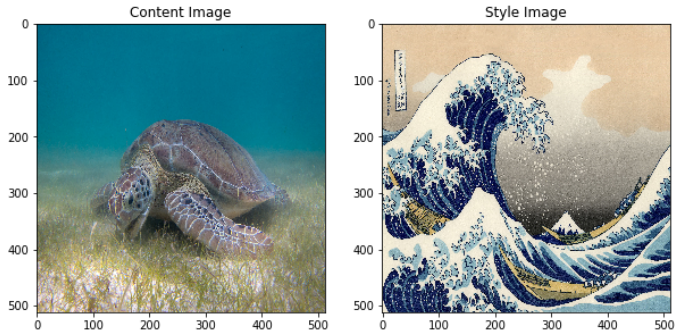


Fig. 1: Content Image: Sea turtle; Style image: Ocean waves painting

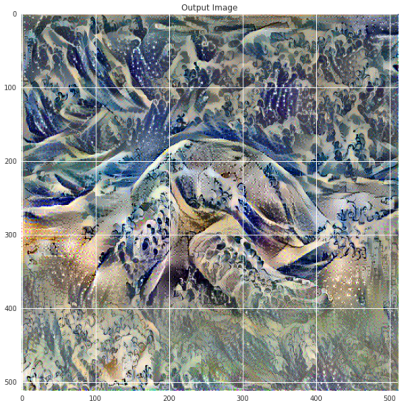


Fig. 2: Final output of NST. It can be observed that the output has the artistic style of style image while it preserves the content of content image

A CNN is comprised of layers and each layers consists image filters that helps in extracting useful features. The extracted features are called feature-maps. Instead of focusing on the pixel values of the image, the network, hierarchically, transforms the image into a representation that focuses more on preserving the actual content of the image. It is easy to check the output information of each layer by reconstructing the image using feature map.

When we reconstruct image at lower layers of CNN, it was observed that pixel values are preserved but on the other hand, reconstructing the

image at higher layers preserves the overall structure of the image. We just need features maps from higher layers for content representation. The style representation involves feature responses form multiple layers. To capture style information, a feature space, which is built using feature maps from all the layers, is used. It has feature correlations among layers which is used to capture the texture of style image but does not care about global arrangement.

Therefore by comparing the feature response of a given image on hidden layers with a target image of certain art style, a loss in art style difference can be quantified. More over, by minimizing this loss value with back propagation, the art style of input image can be transformed towards the target art style. This optimization method was first put forward by Gatys et al. [3] in 2015 and approved by many further studies. However this optimization method suffers from a high computational cost and unstable performances issue due to the nature of back propagation.

Based on Gatys' work, later researches try to solve the issues by training another CNN to learn the mapping performed by the whole optimization process like in [10] and [4]. This approach is often referred to as "generator based method" or "model optimization based method", in comparison to the "image optimization based" in Gatys' work since the former method optimize a generator model during training and the later method perform optimization on input images directly. Thus in this paper we refer to these 2 different kinds of method as "generator approach" and "optimization approach" respectively.

With the trained generator CNN, a stylized image can be generated with a single round of feed forward process, which significantly reduces the computational cost. However the generator approach is also challenging in certain aspects like muti-style transform.

In this paper, we introduce our work of designing and training a generator CNN which can turn any input image into a user selected art style. The work is built upon the idea of generator approach and specifically inspired by the work of Chen et al. [1]. Our goal is to make a generator CNN that fits the need of real world applications, particularly in processing speed and art style variety aspect. To meet the goal, we built the generator in 2 steps: First we generate sufficient amount of training sample for each art style using Gatys' method with a pre-trained VGG-16; Then we train the generator CNN with those samples.

This paper is divided into following sections: Section II is related work which describes the development of art style transform field. Section III provides methodology that illustrate the architecture of generator CNN and the training process in detail. Section IV provides the analysis of the proposed method. Section V discusses the result of the approach while concluding at section VI

II. Related Works

Studies into texture perception exists since decades ago and classic algorithms of analysing or extracting texture patterns like Julesz ensemble [5] [12] have been put forward by researchers. However, not until recent years the topic of art style transformation came to light with the rise of CNNs. Gatys et al.[3] first connected art style, which is a very abstract concept, to the features learned by the hidden layers of CNNs. They proposed a method of art style transformation by optimizing the input image of CNN to minimize the loss in feature responses between input image and target images. In their approach, a pre-trained image recognition CNN, VGG-16, is used as a feature extractor. 2 target images, one for content feature and one for style feature, are fed separately to the CNN and the response on each layer of the CNN are recorded as 2 sets of target responses. Then a white noise image is given as the original input image to the CNN and the response on each layer of the CNN is also computed through feed forward. A loss is defined as a weighted average between the input responses and 2 sets of target responses. At last the loss is optimized using back propagation. During the back propagation process, the parameters of the CNN are locked so only the pixel values in the input image are changed. When the optimization finished, at a given threshold of loss value or number of back propagation performed, the white noise input is turned into a combination of the target content image and style image.

Inspired by Gatys' work, many other researches such as[8] [9] [7] continued to the image

optimization approach. These works mainly attempt to improve the optimization process by modifying the loss function and adding image processing techniques during the process. Li Y's work [8] replaces the feature response mean square error on each layer by an activation distribution alignment and proves its effectiveness. In Risser et al.'s work[9] an extra term, histogram loss is added to the loss function to increase the stability of the optimization process. More over, they replaced global loss with localized loss in computing style loss and applied dynamic parameter adjustment in loss function during training. Li S's work [7] also replace the loss function with the Laplacian loss to better preserve the content detail in the optimization result.

Although optimization approach showed great potential in art style transformation, it is not cost effective since the optimization must be performed every time. Another CNN based approach tries to solve it by training a new CNN, usually in the form of an auto encoder, to do the optimization work [4][10][11][6]. In this way, the mapping from content image to stylized image is done by one feed forward process instead of iterative back propagation. Johnson et al.'s work [4] is one of the earliest that adopted this idea. In their work a deep CNN which has a architecture equivalent to 17 convolutional layers is implemented to learn and perform the mapping. Ulyanov et al.'s work[10] took a similar approach using a CNN with different architecture which has multiple branches operate under different resolutions. Both work uses a VGG-16 network as a loss network, which computes the loss between generator output and content/style target just as it does in image optimization approaches. A later work from Ulyanov et al.[11] uses the same network architecture as Johnson's [4] but trained it to perform a classic texture synthesis method, Julesz ensemble[5], to reduce the computational cost in that method. The loss is then defined as the difference between Gibbs distribution of target and generator output. The issue of these approaches is that the trained CNN generator can only apply a single art style. For multiple art styles, multiple generators have to be trained.

Therefore later researches like [1] and [2] aimed at generate multiple art styles from a single CNN network. Their approaches are based on the finding that CNN generators trained for different art styles can share most of parameters, thus only a small part of parameters needs to be saved for each art style. In Chen et al.'s work[1], the parameters of certain hidden convolutional layers are chosen as the style specific parameters. A different set of parameters of those layers are saved for each distinct art style. In Dumoulin et al.'s work[2], a relationship between an art style the scaling and shifting parameters of normalization layers. Therefore the parameters of normalization layers are used to distinguish different art styles.

III. METHODOLOGIES

As mentioned in section II, in previous works 2 different approaches are taken to fulfill the NST task, which is the optimization based approach and the generator based approach.

Comparing the two approaches, the optimization approach can take any style as target as long as a target style feature image is provided. However, it is slower since the back propagation has to be performed every time. Also the convergence of loss value is not guaranteed, which can cause unstable performance in a real world application. The generator approach is much faster in stylized image generation since it only takes one go of feed forward. However, its disadvantage is also obvious: an individual CNN has to be trained for each style. For a real world application, it means a lot of parameters have to be saved which makes the application undesirably heavy. For our application, we want it to have the process speed of the generator approach while maintaining the style diversity provided by the optimization approach. Thus we took a combined approach to keep the advantage of both approaches.

Our approach consists of 2 steps. The first step is to generate training samples for each of the art styles supported by our application with the optimization approach, using a pre-trained VGG-16 network as the feature extractor. The second step is to train a CNN stylized image generator that supports multiple art styles using the generated samples.

A. CNN Model-VGG16

In our work, we our using a pre-trained VGG16 CNN model for training samples. Fig 3 shows the

architecture of a VGG16 network. The blue colored layers are convolutional layers,the peach colored layers are pooling layers and last three grey colored layers are fully connected(first two) and softmax layers(Third). We only need the convolutional and pooling layers as they act as feature extractors. Fully connected and softmax layers are classifiers. Therefore, we are not using them.

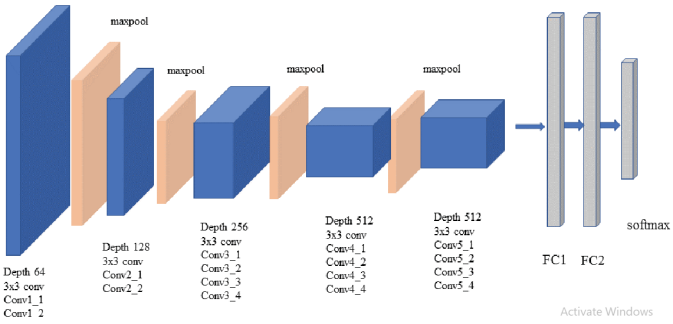


Fig. 3: VGG 16 architecture

B. Generator CNN Architecture

The architecture of the generator is inspired by the design of Chen et al[1]. Chen's work shows that CNN generators trained for different art styles can share a great percentage of parameters and with the changing of parameters in a small number of hidden layers the CNN can generate a wide variety of styles in the output. Figure 4 shows the architecture of our generator CNN. The CNN consists of encoder layers, style bank layers and decoder layers, where the encoder and decoder layers parameters are shared among all styles while the style bank layers have unique parameters for each of the styles. In this way, our application only needs to store the style bank parameters for each individual art style and switch among them when the user selects different art styles.

The Encoder layers consist of 3 convolutional layers: $3 \times 9 \times 9 \times 8$ (input depth, height, width, output depth) kernel size with stride 2, padding valid; $8 \times 3 \times 3 \times 16$ kernel size with stride 2, padding valid; $16 \times 3 \times 3 \times 32$ kernel size with stride 1, padding same. Accordingly, the decoder layers consist of 3 convolutional layers: $32 \times 3 \times 3 \times 16$ kernel size with stride 1; $16 \times 3 \times 3 \times 8$ kernel size with fractional stride $\frac{1}{2}$; $8 \times 9 \times 9 \times 3$ kernel size with fractional stride $\frac{1}{2}$. The style bank layers consist of

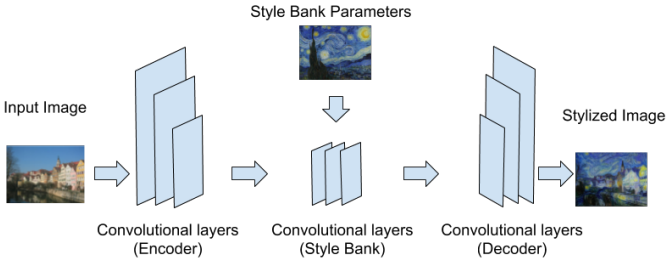


Fig. 4: The architecture of our generator CNN

2 layers, both have $32 \times 3 \times 3 \times 32$ kernel size and stride 1, padding same. All convolutional layers are followed by an instance normalization layer and a ReLU layer except for the output layer.

C. Training of the Generator CNN

The training of the generator CNN consists of 2 steps: the training of encoder/decoder layers and the training of style bank layers. Firstly the training of encoder and decoder is done without the style bank layers, which means the output of encoder layers goes directly into the decoder layers. It can be done since the size of the last encode layer, style bank layers and the first decoder layer are the same. The training uses the input images as the target images as in the training of an auto-encoder. After the training of encoder and decoder layers,

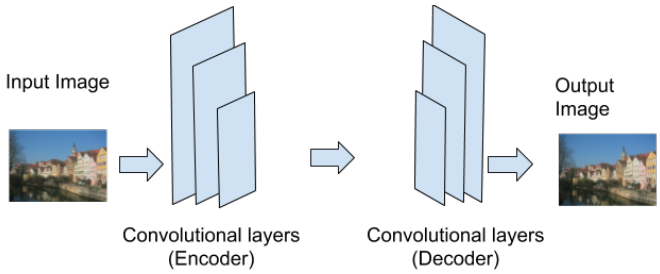


Fig. 5: First step: training of the auto-encoder

their parameters are fixed and the training of style bank layers is done using origin image as input and stylized image as target output. Once the training on one style is done, the style bank layers parameters are saved and training on another style starts until all supported styles are trained.

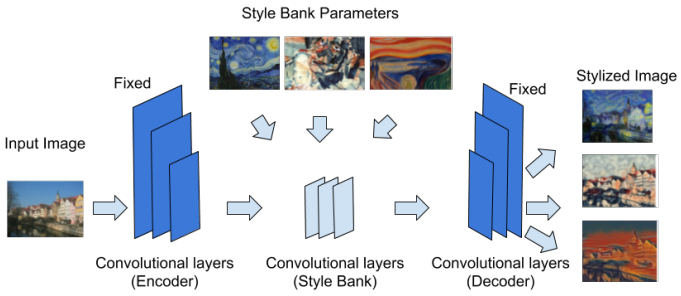


Fig. 6: Second step: training of the style bank

D. Defining Loss Function

For the sample generating process, we followed the setting of Gatys. The total loss L_t consists of two parts: content loss L_c and style loss L_s.

$$L_t = \alpha L_c + \beta L_s \tag{1}$$

$$\alpha/\beta = 0.01 \tag{2}$$

L_c and L_s are pixel wise mean square errors between input responses and the corresponding target responses.

For the generator CNN training, there are 2 loss functions, one for encoder and decoder training and one for style bank training. For encoder and decoder training, the loss L_1 is defined as the pixel wise mean square error between the input and output image. For style bank training, the loss L_2 is defined as the pixel wise mean square error between the output image and the target stylized image.

$$L_1 = \sum_{i=1}^{N} (Pout_i - Pin_i)^2/N \tag{3}$$

$$L_2 = \sum_{i=1}^{N} (Pout_i - Ptar_i)^2/N \tag{4}$$

where N is the number of pixels in the image and Pin_i, $Pout_i$, $Ptar_i$ are ith pixels in the input, output and target images.

E. Image Optimization in CNN

As discussed in the previous sections, we know that the deeper layers of CNN capture high level features of the image which represents the content. Our target image must maintain the content which requires us to minimize the difference between the deeper layers in the feature representation of the content and the output image. By doing this, we will make sure that the two images retain similar features in the deeper layers to match the content of each other The loss between the output image and the content and the style image needs to be minimized to obtain a visually appealing combination of these images. To achieve this, we have tried both first order and second order optimization algorithms available. While first order methods use gradient to minimize the loss function, second order methods use the second derivative to minimize the loss function. For example, L-BFGS uses an approximation of Hessian. Similar to the content loss, we can calculate style loss between the output image and the style image as the difference between the activations of the style filters for each image. As the combined image of content and style goes through our generator CNN, it excites some filters in both content layers and the style layer. By adding these losses, we obtain the total loss on which optimization algorithms are applied.

While the first order optimization methods like Adadelta and Gradient Descent requires more time to minimize the loss, second order optimization methods like Adam and L-BFGS tries to converge the loss faster but they are costly to compute. The Fig 7 shows the loss vs time values for different optimizer in style transfer.

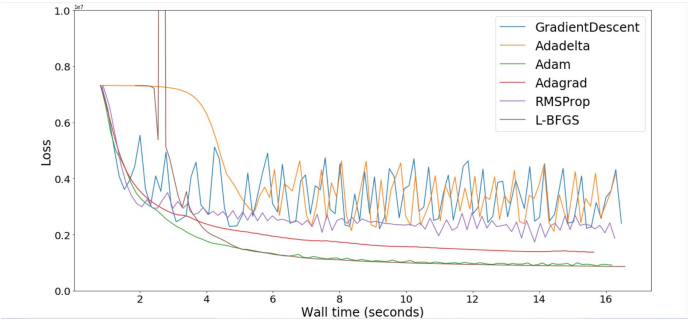


Fig. 7: Loss vs Time for different optimizer

IV. EXPERIMENTAL ANALYSIS

We first obtain the training sample from a pre-trained VGG16 network similar to [3]. We obtained 50 samples using this method. These samples were used for training generator CNN. Style banks store the parameters corresponding to different style images. The user of this application can choose from

Fig. 8: Input image for the network

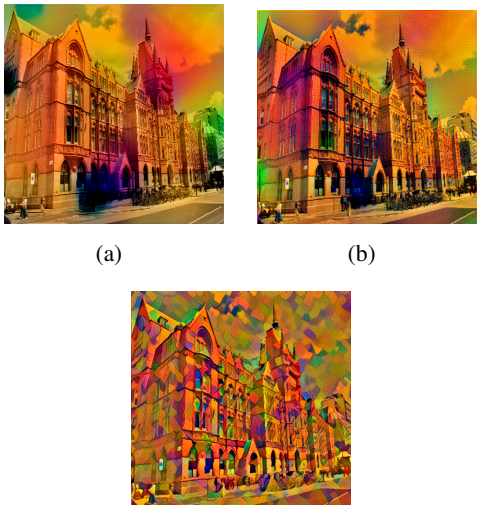
(a) (b)

Fig. 9: Style Reconstruction

these different styles and provide a content image which will undergo style transfer. The image transition through the encoder layers can be visualized by reconstructing the image from the response at each convolutional layer. By this we can see the style invariant features learned by the encoder layers. The response is basically the feature maps that each layer will output. Fig 8 shows input image used for style transfer.

We can visualize the style representation using the feature space which is built using responses from each style bank layer. Feature space has correlation between these layers which is further used obtain multi-scale representation. This representation is used to capture texture information.

Fig 9 shows the reconstruction of style image at the style bank layers. It can be observed that as the feature space in these layers includes responses from more layers, the global arrangement decreases but texture and style from the given image clearly appears on the generated image. Similarly the combined image can be reconstructed at each decoder layers which can be seen in Fig 10. Along the



(a) (b)

(c)

Fig. 10: Output image reconstruction at each layer

network hierarchy, as numbers of filters increases, the size of filtered image needs to be reduced. So a downsampling mechanism is used to reduce the total number of units per layer.

In our network configuration, the style bank can be incrementally trained for new style without having to train the entire network. This advantage comes from decoupling the encoders and the style bank. The parameters of the encoders are fixed while training the style bank layers. This architecture potentially reduces the learning time which makes it suitable for a software application.

V. Experiment Results

Fig 11 shows the output of the generator CNN with respect to the given input and style. This net-

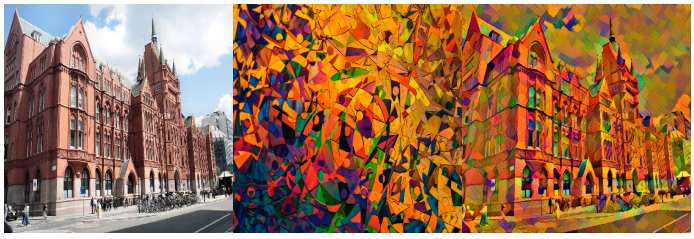


Fig. 11: Output of the network trained on Adam optimizer

work was trained for 100 iterations using Gradient Descent as the optimizer. The network was trained on the samples generated by pre trained VGG-16.

While Adam optimizer is fast when compared to others, training using adam might be computationally expensive for software applications when there is a need to add new style to the application. Hence we compare the results obtained from choosing gradient descent as the optimizer to the adam's. From Fig 12, we can see that image produced is almost same as the one produced by the adam optimizer. Major changes can be seen in the images

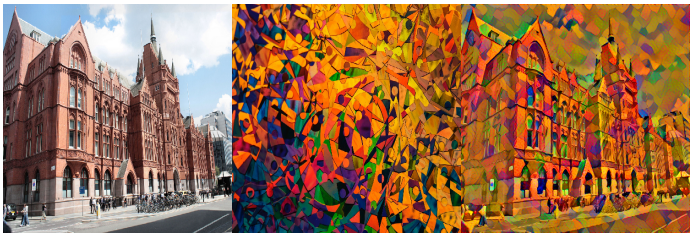


Fig. 12: Output of the network trained on Gradient Descent optimizer

constructed at lower number of iterations. But as the number of iterations increase, the loss drops down and stabilizes after certain point in both the optimizers. Choosing gradient descent as the optimizer, we can see the total loss for different iterations in Fig 13.

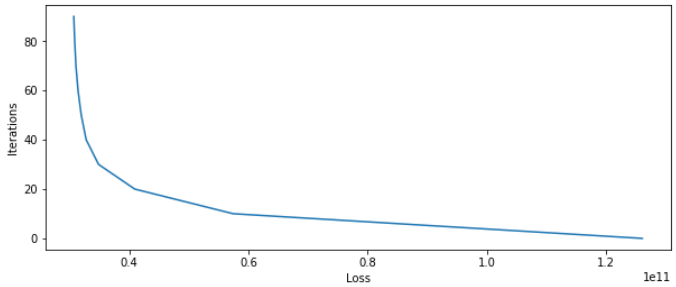


Fig. 13: Loss vs Iterations while training the network

Fig 14 and Fig 15 shows the output of gatys' approach and the proposed generator CNN approach respectively. We can see that our architecture, while minimizing the training time and without having to load huge VGG network, it performs the style transfer. Qualifying for a software application, our network has the virtues of a feed forward network while occupying lesser memory.

Some of the outputs that were produced from this application is shown in Fig 16 and Fig 17

Fig. 14: Style transfer using Gatys' approach

Fig. 15: Style transfer using our approach

Fig. 16: Style transfer 1

VI. CONCLUSION

In our project, we successfully designed and implemented a generator CNN for art style transformation, which meets the need of a real world application. The built CNN model is capable of generating user selected stylized images with minor

Fig. 17: Style transfer 2

computational cost. We validated the correctness of the model by visualizing convolutional layer feature responses and correlations. The performances of different optimizers during model training are compared and recorded for future implementations.

In the end, our experiment proves the feasibility of art style changing products being deployed onto platforms with less computational power and storage space, such as smartphones, TVs or even digital cameras.

REFERENCES

[1] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stylebank: An explicit representation for neural image style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1897–1906, 2017.

[2] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *arXiv preprint arXiv:1610.07629*, 2016.

[3] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[4] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[5] Bela Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, 1981.

[6] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European conference on computer vision*, pages 702–716. Springer, 2016.

[7] Shaohua Li, Xinxing Xu, Liqiang Nie, and Tat-Seng Chua. Laplacian-steered neural style transfer. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1716–1724, 2017.

[8] Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. Demystifying neural style transfer. *arXiv preprint arXiv:1701.01036*, 2017.

[9] Eric Risser, Pierre Wilmot, and Connelly Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. *arXiv preprint arXiv:1701.08893*, 2017.

[10] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, volume 1, page 4, 2016.

[11] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6924–6932, 2017.

[12] Song Chun Zhu, Xiu Wen Liu, and Ying Nian Wu. Exploring texture ensembles by efficient markov chain monte carlo-toward a" trichromacy" theory of texture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):554–569, 2000.