sustainability

MDPI

Article

An Improved Style Transfer Algorithm Using Feedforward Neural Network for Real-Time Image Conversion

Chang Zhou [1], Zhenghong Gu [1], Yu Gao [1] and Jin Wang [2,3,*]

[1] College of Information Engineering, Yangzhou University, Yangzhou 225000, China; zhouchangyz@163.com (C.Z); guzhenghong@yzu.edu.cn (Z.G.); mx120170403@yzu.edu.cn (Y.G)
[2] Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transportation, School of Computer & Communication Engineering, Changsha University of Science & Technology, Changsha 410000, China; jinwang@csust.edu.cn
[3] School of Information Science and Engineering, Fujian University of Technology, Fuzhou 350000, China
* Correspondence: jinwang@csust.edu.cn; Tel.: +86-180-1484-9250

Received: 27 September 2019; Accepted: 11 October 2019; Published: 14 October 2019

Abstract: Creation of art is a complex process for its abstraction and novelty. In order to create those art with less cost, style transfer using advanced machine learning technology becomes a popular method in computer vision field. However, traditional transferred image still troubles with color anamorphosis, content losing, and time-consuming problems. In this paper, we propose an improved style transfer algorithm using the feedforward neural network. The whole network is composed of two parts, a style transfer network and a loss network. The style transfer network owns the ability of directly mapping the content image into the stylized image after training. Content loss, style loss, and Total Variation (TV) loss are calculated by the loss network to update the weight of the style transfer network. Additionally, a cross training strategy is proposed to better preserve the details of the content image. Plenty of experiments are conducted to show the superior performance of our presented algorithm compared to the classic neural style transfer algorithm.

Keywords: style transfer; convolution neural network; cross training; machine learning

1. Introduction

Advanced machine learning technology makes the automatically style transfer possible because of its powerful fitting ability [1–5]. Style transfer as a popular method applicated in artistic creation has attracted much attention. It commonly combines the style information from a style image with the original content image [6–8]. The fused picture preserves the features of the content image and style image simultaneously. The strong ability of features extraction using convolution neural network improves the quality of the synthetic image by style transfer. By adopting style transfer, people can create works of art easily and don't need to care how to professionally draw a picture [9,10]. Additionally, much repetitive work can be omitted and the business costs can be reduced. The improved quality and the automated process make style transfer popular in artistic creation [11–13], font style transformation [14–16], movie effects rendering [17,18] and some engineering fields [19–22].

Traditional style transfer mainly adopts the following methods.

(1) Stroke-Based Rendering: Stroke-based rendering refers to the method of adding a virtual stroke to a digital canvas to render a picture with a particular style [23–25]. The obvious disadvantage is that its application scenarios are only limited to oil paintings, watercolors, and sketches and it's not flexible enough.

(2) Image Analogy: Image analogy is used to learn the mapping relationship between a pair of source images and target images. The source images are transferred by a supervised way. The training set includes a pair of uncorrected source images and corresponding stylized images with a particular pattern [26,27]. The analogy method owns an effective performance and its shortage is that the paired training data is difficult to obtain.
(3) Image Filtering: Image filtering method adopts a combination of different filters (such as bilateral and Gaussian filters) to render a given image [28,29].
(4) Texture Synthesis: The texture denotes the repetitive visual pattern in an image. In texture synthesis, similar textures are added to the source image [30,31]. However, those texture synthesis-based algorithms only use low-level features and their performance is limited.

Recent years, Gatys et al. [6] present a new solution for style transfer combined with the convolution neural network. It regards the style transfer as the optimization problem and adopts iterations to optimize each pix in the stylized picture. The pretrained Visual Geometry Group 19 (VGG19) network is introduced to extract the content feature and style feature from the content image and style image, respectively. Owing the greatly improved performance to the neural network, the method Gatys et al. proposed is also called neural style transfer. Though neural style transfer performs much better than some traditional methods, some drawbacks still trouble the researchers. Firstly, neural style transfer needs to iterate to optimize each pix of the stylized image and it's not applicable to some delay-sensitive applications especially those needing real-time processing. Secondly, though the style features can be well integrated into the stylized image, the content information is inevitable lost. For example, the color in the content image will be mixed with the color in the style image and the lines in the stylized image will show varying degrees of distortion.

In order to make up for the lack of the classic neural style transfer, an improved style transfer algorithm adopting a deep neural network structure is presented. The whole network is composed of two parts, style transfer network and loss network. The style transfer network conducts a direct mapping between the content image and the stylized image. The loss network computes the content loss, style loss, and TV loss between the content image, style image, and stylized image generated by the style transfer network. Then the weight of the style transfer network can be updated according to the calculated loss. The style transfer network needs to be trained while the loss network adopts the first few layers of the pretrained VGG19. A cross training strategy is presented to make the style transfer network to preserve more detailed information. Finally, numerous experiments are conducted and the performances are compared between our presented algorithm and the classic neural style transfer algorithm.

We outline the paper as follows. Section 1 introduces the background of style transfer. Some parallel works are summarized in Section 2. Section 3 demonstrates the effects of feature extraction from different layers of VGG19. Section 4 has a specific illustration of our proposed algorithm. Section 5 conducts the experiments and analyzes the experiment results. Merits and demerits are discussed in Section 6. Section 7 makes a conclusion for the whole paper.

2. Related Work

A deep network structure with multiple convolutional layers is proposed for image classification [32]. Small filters are introduced for detailed features extraction and less parameters need to be trained simultaneously. Due to the favorable expansibility of the well trained VGG network, many other researchers adopt it as a pretrained model for further training.

In order to solve the content losing problem, a deep convolution neural network with dual streams is introduced for feature extraction and an edge capture filter is introduced for synthetic image quality improving [33]. The convolution network contains two parts, detail recognizing network and style transfer network. A detail filter and a style filter are respectively applied to process the synthesized images from the detail recognizing network and style transfer network for detail extraction and color extraction. Finally, a style fuse model is used to integrate the detailed image and the color image into a high-quality style transfer image.

A style transfer method for color sketch synthesis is proposed by adopting dilated residual blocks to fuse the semantic content with the style image and it works without dropping the spatial resolution [34]. Additionally, a filtering process is conducted after the liner color converts.

A novel method combined with the local and global style losses is presented to improve the quality of stylized images [35]. The local style preserves the details of style image while the global style captures more global structural information. The fused architecture can well preserve the structure and color of the content image and it reduces the artifacts.

An end-to-end learning schema is created to optimize both the encoder and the decoder for better features extraction [36]. The original pretrained VGG is fine-tuned to adequately extract features from style or content image.

In order to preserve the conspicuous regions in style and content images, the authors adopt a localization network to calculate the region loss from the SqueezeNet network [37]. The stylized image can preserve the conspicuous semantics regions and simple texture.

Advanced Generative Adversarial Networks (GAN) technology is introduced to style transfer for cartoon images [38]. Network training using the unpaired images makes the training set easier to build. To simulate the sharp edges of cartoon images, the edge loss is added to the loss function. The Gaussian smoothing method is first used to blur the content image, and then the discriminator determines the blurred image as a negative sample. A pre-trained process is executed in the previous several epochs for the generator to make the GAN network converge more quickly.

A multiple style transfer method based on GAN is proposed in [39]. The generator is composed of an encoder, a gated transformer, and a decoder. The gated transformer contains different branches and different styles can be adopted by passing different branches.

3. Features Extraction from VGG19

In a pretrained convolution neural network, the convolution kernels own the ability to extract the features from a picture. Therefore, similarly as the classic neural style transfer, we adopted VGG19 to extract the features from content and style images. VGG19 is a very deep convolution neural network trained with ImageNet dataset and has excellent performance in image classification, object positioning, etc. VGG19 owns good versatility for features extractions and many works adopt it as the pretrained model. Different from the classic neural style transfer, we first analyze the extracted features by the VGG19 to select the suitable layers for feature extraction.

Since the features extracted by each layer of the VGG19 have multiple channels and cannot be directly visualized, we adopt the gradient descent algorithm to reconstruct the original image according to the features extracted by the different layer of the VGG19. The reconstructed images are initialized with Gaussian noise and then we put the initialized images into the VGG19. Then the extracted features are compared in the same layer and their L_2 loss are calculated. Next, the L_2 loss is back propagated to the reconstructed image and the reconstructed image is updated according to the gradient. When reconstructing the content image, we directly use the extracted features to calculate the L_2 loss, as shown in Formula 1.

$$\mathcal{L}_c^j(\mathrm{y},\hat{\mathrm{y}}) = \frac{1}{H_j W_j C_j}\left\|\varphi_j(y) - \varphi_j(\hat{y})\right\|_2^2 \tag{1}$$

where y and ŷ denote the original image and the reconstructed image, respectively. φ_j denotes the output value of the *j*-th layer of VGG19 network. H_j, W_j, C_j denote the width, height, and number of channels of the *j*-th layer of VGG19 network. $\|x\|_2$ denotes the Euclidean norm of the vector x. When reconstruct the style image, the Gram matrix needs to be firstly calculated by Formula 2.

$$\mathrm{G}^j(\mathrm{y}) = f^j \cdot {f^j}^{\mathrm{T}} \tag{2}$$

where f^j denotes the reshaped matrix using the extracted features of the *j*-th layer of VGG19. Then the style loss can be defined as Formula 3.

$$\mathcal{L}_s = \frac{1}{C_j}\left\|G^j(y) - G^j(\hat{y})\right\|_2^2 \tag{3}$$

We exchange the layers used for features extraction and conduct much experiments. The experiment results are shown as follows.

As we can clearly see from Figure 1, the lower layers of VGG19 can preserve much more detailed information of content images. While, the deeper layers of VGG19 are more interested in the regular texture which represents the style of a picture. Therefore, the lower layers of VGG19 are more suitable for content features extraction and the deeper layers are more suitable for style features extraction.

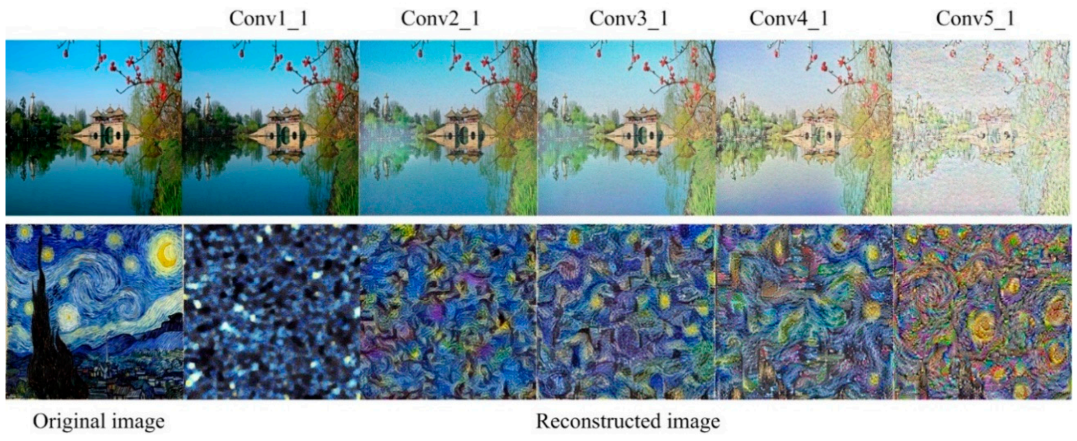

Figure 1. Feature extraction effects of different of the VGG19.

4. Proposed Method

4.1. Data Processing

We firstly process the input data of the network. In order to avoid the problem of color mixing in the classic neural style transfer, a gray conversion is conducted for the content image. We use the classic physiology formula to transform each RGB pixel of content image into grey pixel as Formula 4.

$$Gray(x) = R(x)\cdot 0.299 + G(x)\cdot 0.587 + B(x)\cdot 0.144 \tag{4}$$

where R(x), G(x), and B(x) denote the value of the Red, Green, and Blue (RBG) channels of the pixel x in content image respectively. After the gray conversion, the original content image with three RGB channels is transformed into a gray image with one grayed channel. Whereas, the style image is an RGB picture with three channels, the grayed content image still needs to be converted to the form of three channels to match the format of the style image. Thus, we just simply stack three identical grayed channels as the content image with RGB format. The gray conversion is shown in Figure 2.

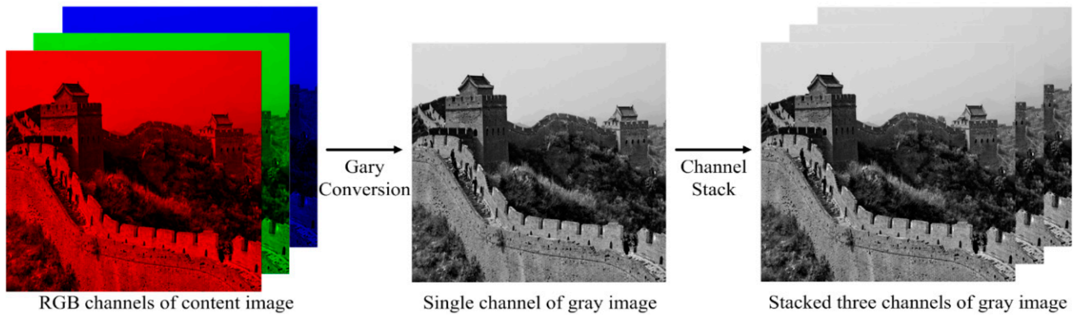
Gary Conversion
Channel Stack
RGB channels of content image
Single channel of gray image
Stacked three channels of gray image

Figure 2. Gray conversion.

Another problem we need to solve is that the style image we used is not limited to the fully texture image. Some regular texture may only concentrate in a centralized area, and we need to use those local features to render the whole content image. Therefore, it's necessary to conduct the data augmentation for the style image to enhance the local features. Following operations are taken for data augmentation.

(1) Zoom in on the original image and then crop the image of the same size.
(2) Randomly rotate the image at a certain angle and change the orientation of the image content.
(3) Flip the image horizontally or vertically.
(4) Randomly occlude part of the image.
(5) Randomly perturb RGB value of each pixel of the image by adding salt and pepper noise or Gaussian noise.

The data augmentation is illustrated as Figure 3.

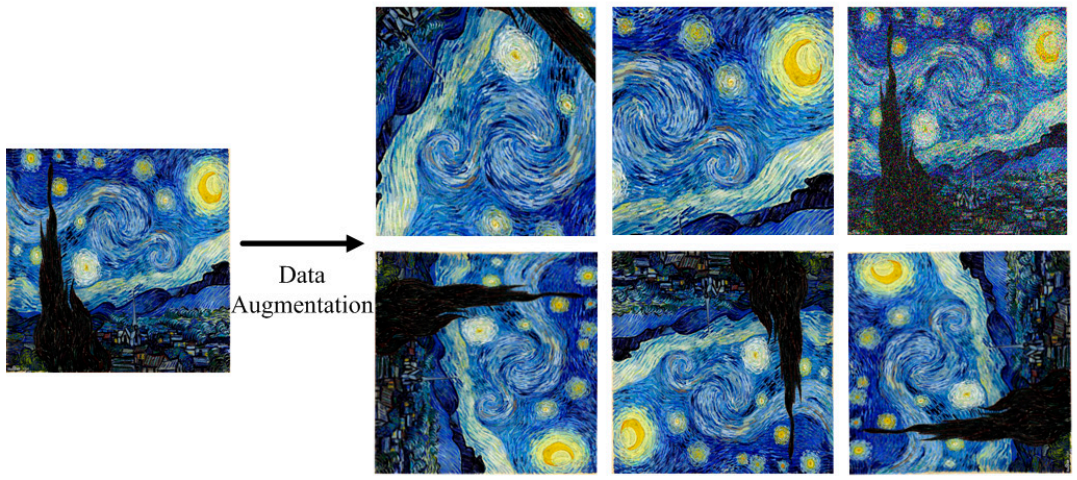

Figure 3. Data augmentation.

4.2. Network Model

The whole network contains two components and they are style transfer network and loss network. The style transfer network realizes a direct mapping between the content image and the stylized image. Then the stylized image is inputted to the loss network to calculate the content and style losses with inputted content and the style image. Next, the weight of the style transfer network will be updated according to the losses calculated in the loss network using gradient descent algorithm. The style transfer network is a deep neural network composed of multiple convolution layers and residual blocks. The weight of each layer in the style transfer network is randomly initialized while the loss network adopts the first few convolution layers of the pretrained VGG19 network. During the training process, only the weight of the style transfer network will be updated. The size of the images must be the same during the training phase, while in the test phase, we can input different sizes of images. The whole network structure and the operation flow are shown in Figure 4.

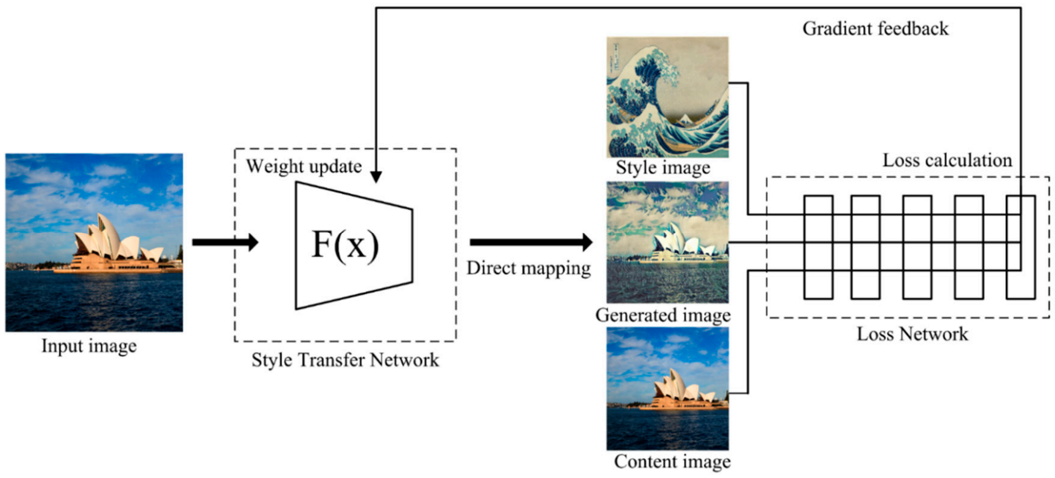

Figure 4. Network model and workflow.

4.3. Style Transfer Network

The style transfer network is stacked by multiple convolution layers, residual blocks, and deconvolution layers. The convolution layers and deconvolution layers adopt short stride for down sampling and up sampling, respectively. Specifically, the style transfer network is composed of four convolution layers, five residual blocks, and two deconvolution layers. Besides the output layer, each convolution or deconvolution layer is followed by a Relu activation layer. The residua block is firstly represented by He et al. [40] in 2016. After two liner transformations, the input data and its initial value is added through a "shortcut" and then the added value is inputted to the Relu activation layers. The whole structure of the style transfer network is shown as Figure 5.

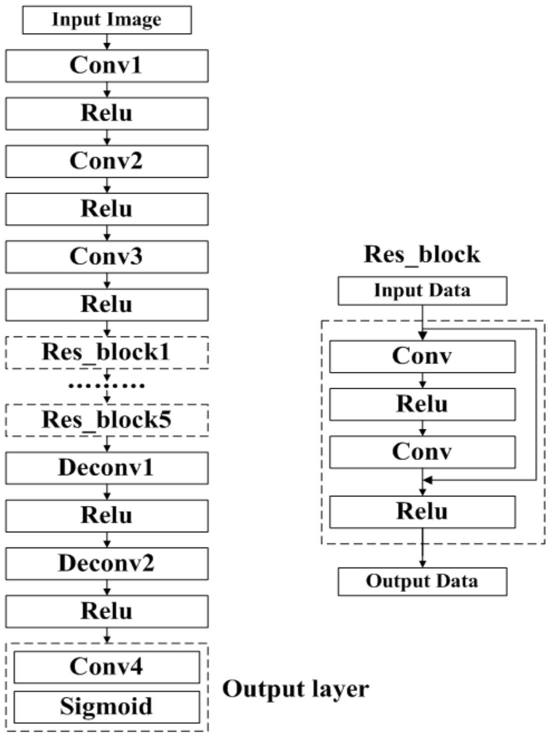

Figure 5. Style transfer network.

4.4. Loss Network and Loss Function

The input of the loss network contains three parts, the style image, the stylized image generated by the style transform network, and the content image. The loss network adopts the first few layers of the VGG19 to extract the features of images and its structure and workflow is shown as Figure 6.

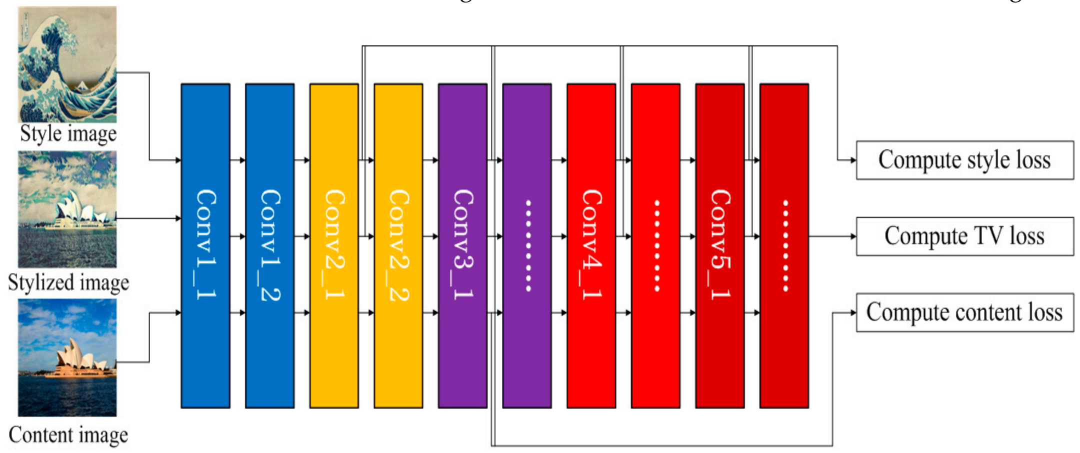

Figure 6. Loss network.

In previous sections, we analyze the ability of feature extraction for different layers in VGG19. The shallower convolutional layers extract the lower features of the image, thus preserving a large amount of detailed information. While the deeper convolution layers can extract higher features in the image, thereby preserving the style information of the image. According to the above rules, we finally adopt "Conv3_1" in VGG19 to extract the content features. Similarly, we adopt "Conv2_1", "Conv3_1", "Conv4_1", and "Conv5_1" in VGG19 to extract the style features.

Content loss describes the difference of features between stylized image and content image. It can be calculated using Formula 5.

$$\mathcal{L}_{content}(y,\hat{y}) = \frac{1}{BS \cdot n_H^l n_W^l n_C^l} \sum_{i=1}^{n_H^l} \sum_{j=1}^{n_W^l} \sum_{k=1}^{n_C^l} (c_{i,j,k}^l - \hat{c}_{i,j,k}^l)^2 \tag{5}$$

where BS denotes the batch size of the input data. n_H^l, n_W^l, and n_C^l denotes the height, width, and number of channels of the l-th layer, respectively. $c_{i,j,k}^l$ and $\hat{c}_{i,j,k}^l$ represent the $i \times j$-th value of the k-th channel after the content image and stylized image are activated by the l-th layer of VGG19.

Gram matrix can be seen as an eccentric covariance matrix between features of an image. It can reflect the correlation between the two features. Additionally, the diagonal elements of the Gram matrix also reflect the trend of each feature that appears in the image. Gram matrix can measure the features of each dimension and the relationship between different dimensions. Therefore, it can reflect the general style of the entire image. We only need to compare the Gram matrix between different images to represent the difference of their styles. The gram matrix can be calculated using Formula 6.

$$G_{k,k'}^l(y) = \sum_{i=1}^{n_H^l} \sum_{j=1}^{n_W^l} c_{i,j,k}^l \cdot \hat{c}_{i,j,k}^l \tag{6}$$

where k and k' both denote the number of channels in the l-th layer.

Style loss means the difference between the Gram matrix of the stylized image and the Gram matrix of the style image. It can be calculated using Formula 7.

$$\mathcal{L}_{style}^l(y,\hat{y}) = \frac{1}{n_H^l n_W^l n_C^l} \sum_{k=1}^{n_C^l} \sum_{k'=1}^{n_C^l} (G_{k,k'}^l(y) - G_{k,k'}^l(\hat{y}))^2 \tag{7}$$

where $G^l(y)$ and $G^l(\hat{y})$ denote the Gram matrix of extracted features in l-th layer for the style image and stylized image.

Then we define the total style loss as the weight sum of all layers and it can be defined as Formula 8.

$$\mathcal{L}_{\text{style}}(\text{y}, \hat{\text{y}}) = \sum \lambda^l \cdot \mathcal{L}^l_{style}(\text{y}, \hat{\text{y}}) \tag{8}$$

where λ^l denotes the weight of l-th layer.

In order to make the generated stylized image smoother, TV loss is introduced to be a regularizer to increase the smoothness of the generated image. TV loss calculates the square of the difference between each pixel and the next pixel in the horizontal and vertical directions. TV loss can be calculated using Formula 9.

$$\mathcal{L}_{TV} = \sum_{i=1}^{n_H^l - 1} \sum_{j=1}^{n_W^l} \sum_{k}^{n_C^l} (c_{i,j,k} - c_{i+1,\text{j},k})^2 + \sum_{i=1}^{n_H^l} \sum_{j=1}^{n_W^l - 1} \sum_{k}^{n_C^l} (c_{i,j,k} - c_{i,\text{j}+1,k})^2 \tag{9}$$

Finally, we can define the total loss as the weight sum of $\mathcal{L}_{content}$, $\mathcal{L}_{\text{style}}$, and $\mathcal{L}_{TV}$. The total loss can be represented as Formula 10.

$$\mathcal{L}_{total} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style} + \gamma\mathcal{L}_{TV} \tag{10}$$

where α, β and γ are three adjustment factors and their values can be adjusted according to the actual demand. We will have a discussion on their values in Section 5.3.

The final target in the training phase is to minimize $\mathcal{L}_{total}$. The weight of the style transfer network will be updated according to the total loss by gradient descent algorithm.

4.5. Cross Training Strategy

In order to preserve as much content information as possible, we use a cross training method by rotationally adopting different loss function. When the num of iteration is even, we adopt the original total loss as loss function, otherwise, the content loss will be chosen as the loss function. The loss function can be defined as Formula 11.

$$\mathcal{L}_{total} = \begin{cases} \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style} + \gamma\mathcal{L}_{TV} & if\ iteration\%2 == 0 \\ \mathcal{L}_{content} & otherwise \end{cases} \tag{11}$$

The workflow of the training process is shown as Figure 7.

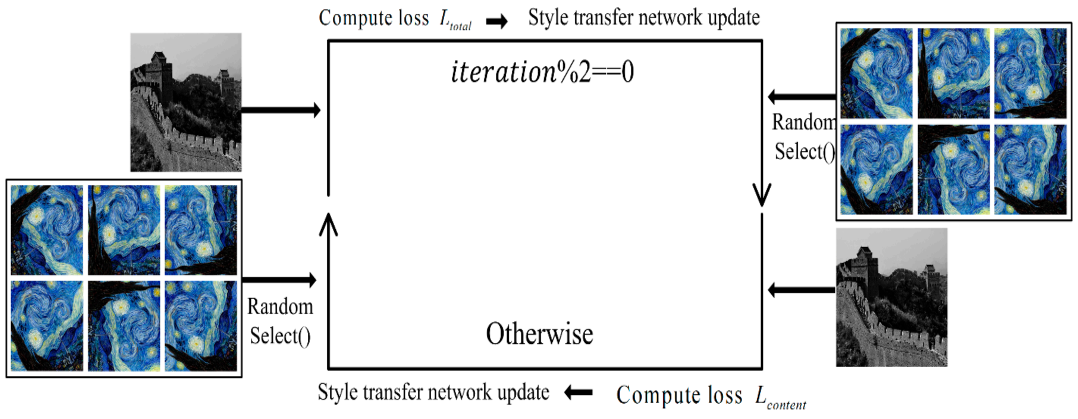
Compute loss L_{total}
Style transfer network update
iteration%2==0
Random Select()
Random Select()
Otherwise
Style transfer network update
Compute loss $L_{content}$

Figure 7. Loss network.5. Experiments and Analysis.

5.1. Experiment Environment and Parameters

In order to have an evaluation of our presented algorithm, we compare it with the classic neural style transfer algorithm proposed by Gatys et al. The experiment environment is a workstation using the Ubuntu operation system. The workstation is also equipped with a GTX 1080 Ti (10G Memory) graphics card to accelerate the training process of the style transfer network. The relevant software versions are shown in the table below.

Table 1. Relevant software versions.

Software Name	Versions
Python	3.7
TensorFlow-GPU	1.13.1
NumPy	1.14.6
SciPy	1.1.0
Matplotlib	3.02
Os	3.7

The specific information of convolution kernels in the style transfer network are illustrated in Table 2.

Table 2. Parameters of convolution kernels.

Convolution Name	Kernels Size, Stride, Number
Conv1	9 × 9, 1, 32
Conv2	3 × 3, 2, 64
Conv3	3 × 3, 2, 128
Res_block_Conv	3 × 3, 1, 128
Deconv1	3 × 3, 2, 64
Deconv2	3 × 3, 2, 32
Conv4	9 × 9, 1, 3

The relevant parameters of the network are listed in Table 3.

Table 3. Parameters for training.

Software Name	Versions
Batch_Size	4
Training Data Size	[256,512,1024]
Number of Training Data	5000
Content_layer	Conv3_1
Style_layer	Conv2_1, Conv3_1, Conv4_1, Conv5_1
Epoch	100
α, β, γ	1, [1,5,10], 1
Optimizer	Adam
Learning Rate	0.001

5.2. Activation Function in Output Layer

In our proposed algorithm, the output layer of the style transfer network can adopt tanh or sigmoid as its activation function. When adopting tanh as the activation function, the final output will adopt Formula 12.

$$y = \tanh(x) * 122.5 + 122.5 \tag{12}$$

where x is the output value of the previous layer. When adopting sigmoid as the activation function, the final output will adopt Formula 13. In both ways, the output value of the output layer can be between 0 and 255.

$$y = \mathrm{sigmoid}(x) * 255 \tag{13}$$

In order to have an evaluation of two different activation functions in the output layer, we test the performance of the network using the same content and style image. The experiment result is shown as Figure 8. We can clearly see from Figure 8 that when adopting tanh as the activation function, the performance is poor and only part of the image is stylized. Whereas, the style image can be well integrated into the content image by the method of adopting sigmoid as the activation function in the output layer.

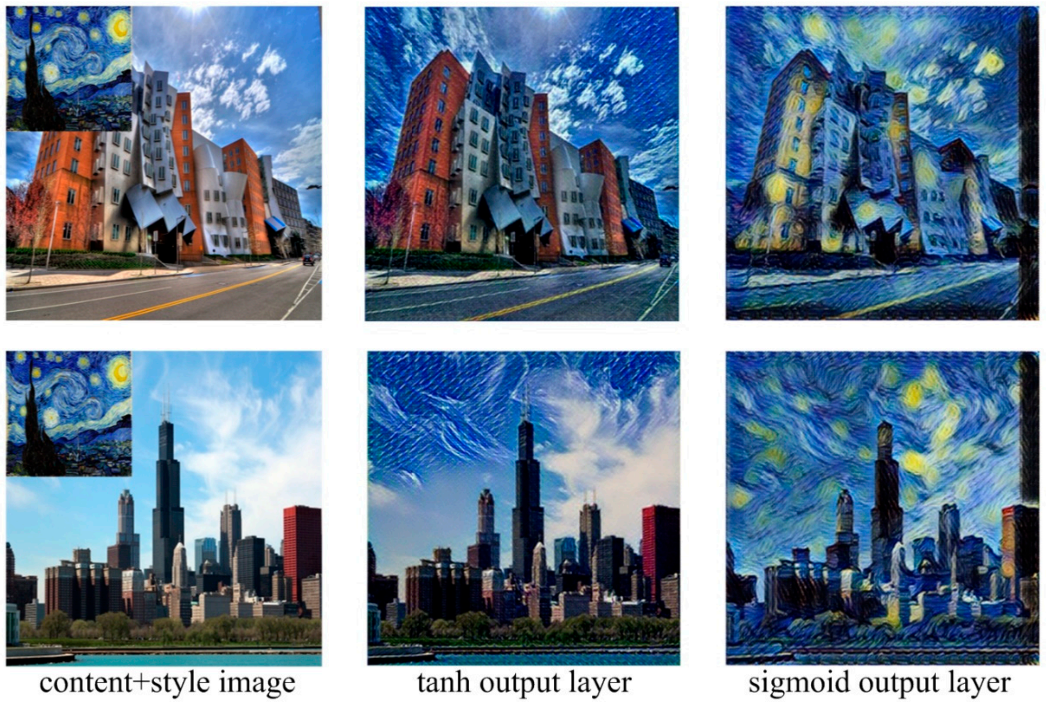

Figure 8. Different activation functions in the output layer.

5.3. Loss Control Factors Adjustment

As we have discussed in the loss network, α, β, γ are three parameters to adjust the proportion of content loss, style loss, and TV loss. The proportion of content loss and style loss has a significant influence on the levels of stylization. When the content loss accounts for a larger proportion, the stylized image will preserve more information of the content image. On the contrary, the stylized image will be better rendered with the value of β increasing. TV loss only affects the smoothness of the stylized image and it owns a small impact on the overall rendering effect. The rendering can be strengthened by decreasing the value of α or increasing the value of β. Different applications can retrofit the values of α and β based on their requirements. As Figure 9 illustrates, when β is 1, the stylization is shallow and when β is increased to 10, the stylization is obvious.

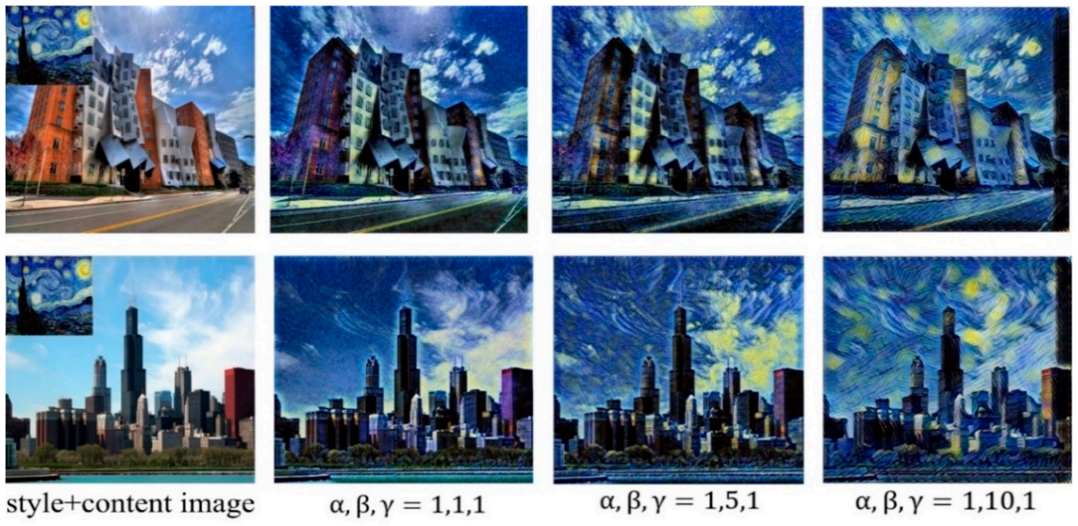
style+content image
$\alpha, \beta, \gamma = 1,1,1$
$\alpha, \beta, \gamma = 1,5,1$
$\alpha, \beta, \gamma = 1,10,1$

Figure 9. Stylized image under different values of β.

5.4. Comparison of Details Preserving

In a stylized image, we expect the objects in it are still recognizable while the background is well rendered. Classic neural style transfer achieves a great performance in image rendering, however, it's weak to preserve the details in the content image. In order to evaluate the performance of the presented algorithm, we compare it with the classic neural style transfer in terms of details preserving. Both two algorithms adopt the same content and style image with 1024×1024 pixels. Classic neural style transfer iterates 1000 times for fully rendering while our presented algorithm iterates 100 epochs for fully training. The experiment result is shown as Figure 10. As Figure 10 illustrates, the classic neural style transfer destroyed partial details from the content image. As you can clearly see in the enlarged picture, that the pillars and the roof of the pavilion have different degrees of missing. While, in our improved algorithm, those details are preserved and the background is well rendered.

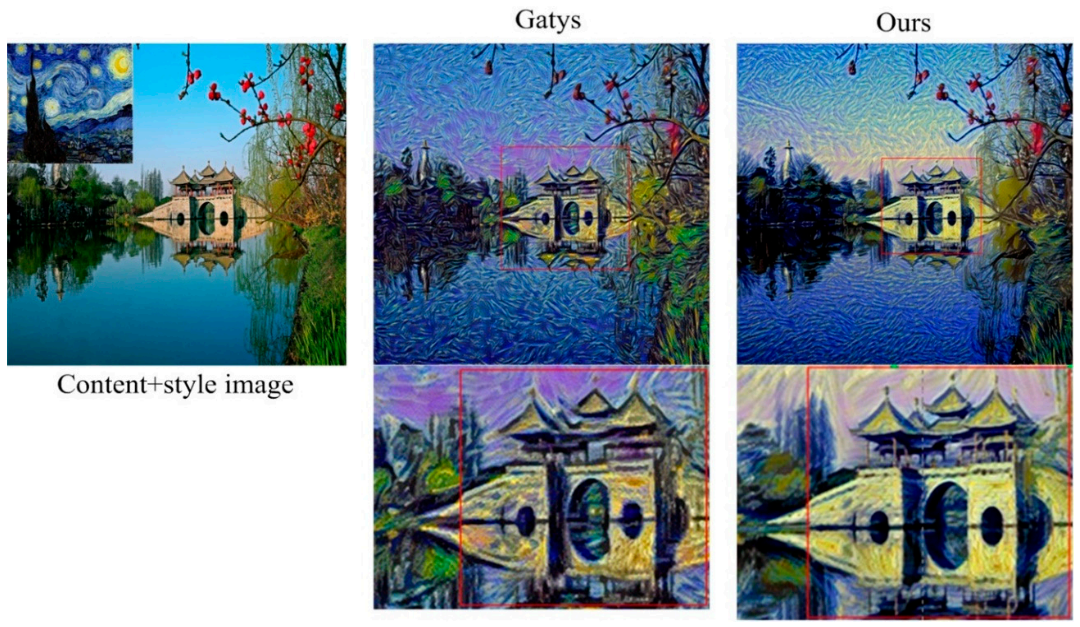

Figure 10. Comparison of details preserving.

5.5. Comparison of Characters Rendering

Sometimes, characters are contained in the content image and commonly, we expect those characters can preserve their original features rather than be rendered. In order to have an evaluation of the presented algorithm in terms of characters rendering, we compare it with the classic neural style transfer. Both two algorithms use the same content and style image with 1024 × 1024 pixels. Classic neural style transfer iterates only 500 times to preserve more features of characters and our presented algorithm still iterates 100 epochs for fully training. The experiment result is shown as Figure 11. As we can clearly see from Figure 11, that both two algorithms achieve a good performance in stylization. However, the classic neural style transfer algorithm stylizes the characters the same as the background which results in the facial features, contours, etc., of the characters become blurred and distorted. This can be explained as that classic neural style transfer adopts the optimization method to convert the original image to the stylized image. Therefore, the network will treat each pixel in the picture indiscriminately, making the content image close to the style image. While in our presented algorithm, the trained deep neural network will recognize the characters in the image and separate them from the background. Thus, the characters still keep complete features and clear outlines.

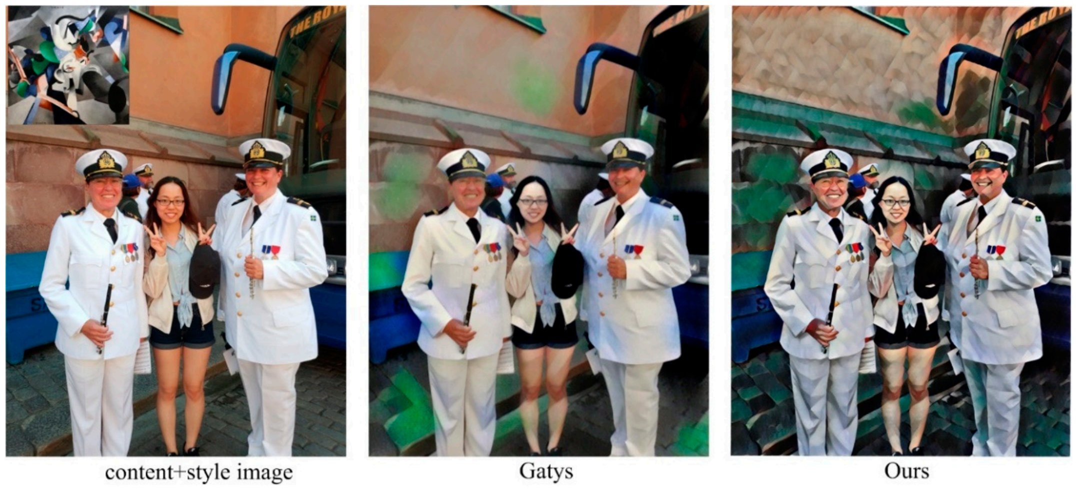

Figure 11. Comparison of characters rendering.

5.6. Comparison of Time Consuming

Finally, we have an evaluation of our presented algorithm compared with the classic neural style transfer in terms of time consuming. Images with different pixels are tested respectively. Both in the classic neural style transfer and our present algorithm, the network needs to be adjusted to fit different sizes of input data. Graphics Processing Unit (GPU) is only used when execute the two different algorithms. Images with high resolution will have a better visual effect and meanwhile, it takes more time for the network to render. Experiment result is shown in Table 4. As we can clearly see from Table 4, the time consuming of both two algorithms increases with image pixel increasing. For the image with the same pixel, our proposed algorithm achieves an enhancement of three orders of magnitude compared with the classic neural style transfer. However, in our presented algorithm, a long time is needed to train the style transfer network.

Table 4. Comparison of time consuming of different algorithms.

Algorithm	Classic Neural Style Transfer Algorithm			Ours	
Image Size	100 Iterations	500 Iterations	1000 Iterations	Training Time	1 Iteration
256 × 256	5.4 s	26.1 s	51.3 s	5 h 32 m	0.05 s
512 × 512	15.1 s	69.6 s	122.7 s	8 h 47 m	0.1 s
1024 × 1024	30.5 s	138.6 s	240.1 s	12 h 17 m	0.2 s

5.7. Other Examples Using Proposed Algorithm

Some other examples using the proposed algorithm are shown as Figure 12.

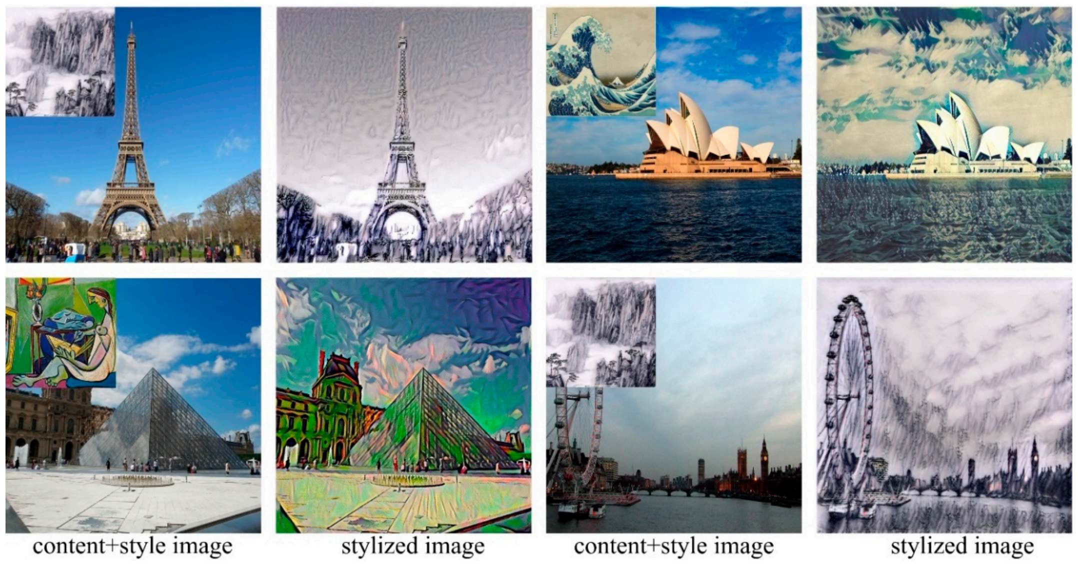

Figure 12. Examples using proposed algorithm.

6. Discussion

The convolution neural network owns an excellent ability for features extraction and it provides an alternative way for the feature comparison between different images. The classic neural style transfer algorithm regards the stylization task as an image-optimization-based online processing problem. While, our presented algorithm regards it as a model-optimization-based offline processing problem. The most prominent advantage of the presented algorithm is the short running time for stylization. Since the network model can be trained in advance, it's suitable for those delay sensitive application especially real-time style transfer. Another advantage is that it can separate important targets from the background to avoid the content loss. Contrary to image-optimization, model-optimization aims to train a model to directly map the content image to the stylized image. The training process makes the model capable to recognize different objects such as characters and buildings, therefore, it can better preserve the details of those objects and separate them from the background.

Meanwhile, there are also some demerits of the proposed algorithm. It's inflexible to switch the style. Once we want to change the style, we need to train a brand-new model which may take a lot of time. However, the image-optimization-based method only needs to change the style image and then iterates to the final solution. Additionally, our presented algorithm needs to run on the device with better performance such as computation and memory which increases the cost.

7. Conclusions

Classic neural style transfer has the demerits of time consuming and details losing. In order to accelerate the speed of stylization and improve the quality of the stylized image, in this paper, we present an improved style transfer algorithm based on a deep feedforward neural network. A style transfer network stacked by multiple convolution layers and a loss network work based on VGG19 are respectively constructed. Three different losses which represent the content, style, and smoothness are defined in the loss network. Then, the style transfer network is trained in advance, adopting the training set, and the loss is calculated by the loss network to update the weight of the style transfer network. Meanwhile, a cross training strategy is adopted during the training process. Our feature work will mainly focus on single model based multi-style transfer and special style transfer combined with Generative Adversarial Networks (GAN).

Author Contributions: Z.G. conceived and designed the experiments; C.Z. and Y.G. performed the experiments and analyzed the data. J.W. wrote this paper.

Acknowledgments: The authors are thankful to the editor and reviewers for their hard work which largely improve the quality of this paper.

Conflicts of Interest: The authors declare no conflict of interest.

Data Availability: The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

1. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. *Proc. Ieee Conf. Comput. Vis. Pattern Recognition* **2015**, arXiv:1512.03385.
2. Zou, W.; Li, X.; Li, S. Chinese painting rendering by adaptive style transfer. *Chin. Conf. Pattern Recognit. Comput. Vision* **2018**, 3–14, doi:10.1007/978-3-030-03338-5_1.
3. Zheng, C.; Zhang, Y. Two-stage color ink painting style transfer via convolution neural network. *In 2018 15th International Symposium on Pervasive Systems, Algorithms and Networks.* **2018**, doi:10.1109/i-span.2018.00039.
4. Liu, S.; Guo, C.; Sheridan, J.T. A review of optical image encryption techniques. *Opt. Laser Technol.* **2014**, *57*, 327–342.
5. Wu, C.; Ko, J.; Davis, C.C. Imaging through strong turbulence with a light field approach. *Opt. Express.* **2016**, *24*, 11975–11986.
6. Gatys, L.A.; Ecker, A.S.; Bethge, M.A. Neural algorithm of artistic style. *Proc. Ieee Conf. Comput. Vis. Pattern Recognition* **2015**, arXiv:1508.06576
7. Karen, S.; Andrew, Z. Very deep convolutional networks for large-scale image recognition. *Proc. Int. Conf. Learn. Representations* **2015**, arXiv:1409.1556
8. Wang, J.; Gao, Y.; Liu, W.; Sangaiah, A.K.; Kim, H.J. An intelligent data gathering schema with data fusion supported for mobile sink in wireless sensor networks. *Int. J. Distrib. Sens. Networks* **2019**, *15*, doi:10.1177/1550147719839581.
9. Qiu, H.; Huang, X. An Improved image transformation network for neural style transfer. *Proc. Int. Conf. Inf. Syst.* **2017**, doi:10.1007/978-3-319-68121-4_28.
10. Wang, J.; Gu, X.; Liu, W.; Sangaiah, A.K.; Kim, H. An empower hamilton loop based data collection algorithm with mobile agent for WSNs. *Human-centric Computing and Information Sciences*. **2019**, *9*, 18.
11. Zeng, H.; Liu, Y.; Li, S.; Che, J.; Wang, X. Convolutional neural network based multi-feature fusion for non-rigid 3D model retrieval. *J. Inf. Process. Systems* **2018**, *14*, 176–190.
12. Daru, P.; Gada, S.; Chheda, M.; Raut, P. Neural style transfer to design drapes. *Proc. Ieee Conf. Comput. Intell. Comput. Res.* **2017**, 1–6, arXiv:1707.09899.
13. Pan, J.S.; Kong, L.P.; Sung, T.W.; Tsai, P.W.; Snasel, V. Alpha-fraction first strategy for hierarchical wireless sensor networks. *J. Internet Technol.* **2018**, *19*, 1717–1726.
14. Johnson, J.; Alahi, A.; Li, F.-F. Perceptual losses for real-time style transfer and super-resolution. *Proceedings of European Conference on Computer Vision.* 2016; pp. 694–711.
15. Qiu, X.; Jia, W.; Li, H. A font style learning and transferring method based on strokes and structure of Chinese characters. *Proc. Int. Conf. Comput. Sci. Serv. System* **2012**, 1836–1839, doi:10.1109/CSSS.2012.457.
16. Pan, J.S.; Lee, C.Y.; Sghaier, A.; Zeghid, M.; Xie, J.F. Novel systolization of subquadratic space complexity multipliers based on toeplitz matrix–vector product approach. *IEEE Trans. Very Large Scale Integr.* **2019**, *27*, 1614–1622.
17. Azadi, S.; Fisher, M.; Kim, V.G.; Wang, Z.; Shechtman, E.; Darrell, T. Multi-content gan for few-shot font style transfer. *Proceedings of the IEEE conference on computer vision and pattern recognition.* **2018**, 7564–7573, arXiv:1712.00516.
18. Wang, J.; Gao, Y.; Liu, W.; Sangaiah, A.K.; Kim, H.J. Energy efficient routing algorithm with mobile sink support for wireless sensor networks. *Sensors* **2019**, *19*, 1494.
19. Nguyen, T.T.; Pan, J.S.; Dao, T.K. An improved flower pollination algorithm for optimizing layouts of nodes in wireless sensor network. *Ieee Access* **2019**, *7*, 75985–75998.
20. Meng, Z.Y.; Pan, J.S.; Tseng, K.K. PaDE: An enhanced differential evolution algorithm with novel control parameter adaptstion schemes for numerical optimization. *Knowl.-Based Systems* **2019**, *168*, 80–99.
21. Pan, J.S.; Kong, L.P.; Sung, T.W.; Tsai, P.W.; Snasel, V. A clustering scheme for wireless sensor networks based on genetic algorithm and dominating Set. *J. Internet Technol.* **2018**, *19*, 1111–1118.

22. Wu, T.Y.; Chen, C.M.; Wang, K.H.; Meng, C.; Wang, E.K. A provably secure certificateless public key encryption with keyword search. *J. Chin. Inst. Eng.* **2019**, *42*, 20–28.
23. Liu, J.; Yang, W.; Sun, X.; Zeng, W. Photo stylistic brush: Robust style transfer via superpixel-based bipartite graph. *Ieee Trans. Multimed.* **2017**, *20*, 1724–1737.
24. Wang, J.; Gao, Y.; Wang, K.; Sangaiah, A.K.; Lim, S.J. An affinity propagation-based self-adaptive clustering method for wireless sensor networks. *Sensors* **2019**, *19*, 2579.
25. Wang, J.; Gao, Y.; Yin, X.; Li, F.; Kim, H.J. An enhanced PEGASIS algorithm with mobile sink support for wireless sensor networks. *Wirel. Commun. Mob. Computing* **2018**, *2018*, 9472075.
26. Ghrabat, M.J.J.; Ma, G.; Maolood, I.Y.; Alresheedi, S.S.; Abduljabbar, Z.A. An effective image retrieval based on optimized genetic algorithm utilized a novel SVM-based convolutional neural network classifier. *Hum. -Cent. Comput. Inf. Sci.* **2019**, *9*, 31.
27. Zeng, D.; Dai, Y.; Li, F.; Wang, J.; Sangaiah, A.K. Aspect based sentiment analysis by a linguistically regularized CNN with gated mechanism. *J. Intell. Fuzzy Systems* **2019**, *36*, 3971–3980.
28. Zhang, L.; Wang, Y. Stable and refned style transfer using zigzag learning algorithm. *Neural Process. Lett.* **2019**, doi:10.1007/s11063-019-10024-w.
29. Tu, Y.; Lin, Y.; Wang, J.; Kim, J.U. Semi-supervised learning with generative adversarial networks on digital signal modulation classification. *Comput. Mater. Continua.* **2018**, *55*, 243–254.
30. Li, C.; Liang, M.; Song, W.; Xiao, K. A multi-scale parallel convolutional neural network based intelligent human identification using face information. *J. Inf. Process. Systems* **2018**, *14*, 1494–1507.
31. Liu, D.; Yu, W.; Yao, H. Style transfer with content preservation from multiple images. *Proc. Pac. Rim Conf. Multimed.* **2017**, doi:10.1007/978-3-319-77380-3_75.
32. Hu, J.; He, K.; Hopcroft, J.E.; Zhang, Y. Deep compression on convolutional neural network for artistic style transfer. *Proc. Natl. Conf. Theor. Comput. Sci.* **2017**, doi:10.1007/978-981-10-6893-5_12.
33. Wang, L.; Wang, Z.; Yang, X.; Hu, S.; Zhang, J. Photographic style transfer. *Vis. Computer.* **2018**, doi:10.1007/s00371-018-1609-4.
34. Zhang, W.; Li, G.; Ma, H.; Yu, Y. Automatic color sketch generation using deep style transfer. *Ieee Comput. Graph. Applicat.* **2019**, *39*, 26–37.
35. Zhao, H. H.; Rosin, P.L.; Lai, Y.K.; Lin, M.G.; Liu, Q.Y. Image neural style transfer with global and local optimization fusion. *Ieee Access* **2019**, *7*, 85573–85580.
36. Yoon, Y.B.; Kim, M.S.; Choi, H.C. End-to-end learning for arbitrary image style transfer. *Electron. Lett.* **2018**, *54*, 1276–1278.
37. Liu, Y.; Xu, Z.; Ye, W.; Zhang, Z.; Weng, S.; Chang, C.C.; Tang, H. Image neural style transfer with preserving the salient regions. *Ieee Access* **2019**, *7*, 40027–40037.
38. Chen, Y.; Lai, Y.; Liu, Y. CartoonGAN: Generative adversarial networks for photo cartoonization. *Proc. Ieee Conf. Comput. Vis. Pattern Recognition* **2018**, doi: 10.1109/CVPR.2018.00986
39. Chen, X.; Xu, C.; Yang, X.; Song, L.; Tao, D. Gated-gan: Adversarial gated networks for multi-collection style transfer. *Ieee Trans. Image Process.* **2018**, *28*, 546–560.
40. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. *Proc. Ieee Conf. Comput. Vis. Pattern Recognition* **2016**, 770–778, arXiv:1512.03385