Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186-5140
Volume 12, Number 1, pages 6–12, January 2023

An Art Font Generation Technique using Pix2Pix-based Networks

Minghao Xue, Yasuaki Ito, and Koji Nakano
Graduate School of Advanced Science and Engineering, Hiroshima University
Kagamiyama 1-4-1, Higashi-hiroshima, 739-8527, JAPAN

***Abstract*—Art font is a typeface decorated with special visual effects and is widely used in two-dimensional graphic design. Generating art fonts directly from images is a convenient way to design art fonts that reduce the need for expertise. The appearance of such decorative fonts is similar to the foreground of the reference image. In this paper, we propose an art font generation method using machine learning, which uses multiple Pix2Pix networks to quickly generate high-quality art fonts similar to the foreground of a reference image. In this method, first of all, three Pix2Pix networks are used directly. One of the networks is used for mask dataset generation and the other two networks are used for art font generation. Secondly, the parameter β can be used directly to control the effect of the network generation without changing the network structure of Pix2Pix. The experimental results show that the proposed method effectively controls shape change and texture imposition by dividing them and can generate artistic art fonts. Furthermore, we showed that the proposed method could generate artistic and complicated art fonts for input fonts with different shapes, indicating that the method has high practical utility.**

***Index Terms*—art font, Pix2Pix, Generative Adversarial Network, machine learning**

I. Introduction

Art font is a kind of font with a unique decorative effect, commonly used in the field of 2D graphic design such as advertising. It includes traditional calligraphy and fonts based on stylized effects. Usually, these art fonts are designed by professionals using professional software, which is not efficient and it is time-consuming and laborious for non-experts. On the other hand, designing art characters is a large amount of work. Since not only English letters are included, but also Japanese, Chinese, etc. need to be specially designed, it is impractical to manually design each type of art font. Because of the importance of art fonts in life, there are many algorithms related to art font generation [1]–[4].

Recently, machine learning has been well developed in the field of computer vision, especially image processing techniques have been improved [5]–[8]. Once the network is trained, it can generate a large number of high-quality images at once. Because of the use of *Generative Adversarial Networks (GANs)* [9], the image generation effect, quantity, and generalization ability have been greatly improved. Huang *et al.* [10] proposed a novel adaptive instance normalization layer that aligns the mean and variance of the content features with those of the style features. This method is fast and without the restriction of a pre-defined set of styles. Li *et al.* [11] proposed a Markovian Generative Adversarial Network, it can directly decode brown noise to realistic texture, or photos to artistic paintings. Mirza *et al.* [12] proposed conditional Generative Adversarial Networks(cGANs), the output image of the network can be controlled by the input condition data. Isola *et al.* [13] proposed Pix2Pix Network, based on cGANs, is a general-purpose solution to image-to-image translation problems.

Due to the importance of art fonts in daily life, many studies related to art fonts have been generated. Yang *et al.* introduced a traditional method of patch-based texture synthesis algorithm to map features to correlated positions on character skeleton [1]. Since traditional algorithms usually have low generation efficiency, Gao *et al.* proposed a machine learning based art font generation algorithm, AGIS-Net, to transfer both shape and texture styles in one-stage with only a few stylized samples [2]. However, since art fonts contain both shapes and textures, generating them at once will cause information loss. In order to reduce the information loss of shapes and textures of art fonts, Yuan *et al.* proposed a cGANs-based art font generation network containing two cGANs networks for changing shape and texture respectively [3]. Yang *et al.* [4] proposed a shape-matching GAN to generate real-time controllable art font with foreground features from reference images. This method works well when generating the outer outline of the font. However, it is difficult to generate good effects inside the complicated fonts when the artistic effect is high. For example, the flame font should have sticky effect inside when the artistic style is very strong. In addition, this method uses a many-to-one network training method with a very complex training process.

In machine learning, Convolutional Neural Networks (CNNs) is one of the most commonly used algorithms in the field of image generation. Gatys *et al.* introduces an artificial system based on CNNs that creates artistic images of high perceptual quality [5]. [6] proposed A Neural Algorithm of Artistic Style that can separate and recombine the image content and style of natural images. It can combine the content of a photograph with the appearance of numerous well-known artworks. Li *et al.* studies a combination of generative Markov random field models and discriminatively trained deep convolutional neural networks for synthesizing 2D images [7]. Ulyanov *et al.* proposed to replace the batch normalization layer with an instance normalization layer, which is a great improvement for image generation [8]. Johnson *et al.* proposed a perceptual loss based on VGG16 and links it with

convolutional neural networks, improving the texture quality of transferred images [14]. Due to information loss, using a single network may not generate the best image. Xue *et al.* divided the generation of Chinese landscape painting into two networks: line sketching and coloring. [15]. Yuan *et al.* [3] proposed a two-step art font generation method based on cGAN, and Yang *et al.* [4] proposed a multi-step art font generation method based on GAN, proving the effectiveness of the separation network for art font generation.

This paper proposes a method to generate art fonts similar to the foreground of the reference image using Pix2Pix networks, while the shape of the generated art fonts can be controlled directly using parameters. In the task, the shape and color of the art font must be similar to the reference image, at the same time the Pix2Pix network generates images that require paired datasets. Therefore, we split this task into multiple steps to complete. Fig. 1 shows our proposed method for generating art fonts.

This method consists of three Pix2Pix networks, the shape convert network, the color convert network, and the dataset generation network. The shape convert network is used to generate art font masks. It can control the shape of the generated art font mask in real-time using the parameter β, varying between artistry and readability. The color convert network is used to generate the art font image. It takes the art font mask output from the first network and fills it with texture to create the final art font effect, such as flame, water, etc. Since the shape convert network has the function of generating controllable images in real-time, it needs to use a large number of art font mask images during training. We used the dataset generation network to generate art font masks with different hyper-parameters to make multiple art font datasets for the shape convert network training. We add the original font to training, and use hyperparameters to control the effect of the dataset generation network, which can switch between extreme deformation effects and almost completely unchanged fonts, with a wide range of font variations. We performed various scaling and cropping of the reference image, and kept the scaling of the original image to make full use of the small data set, at the same time, we used perceptual loss [14] in texture filling, so that the generated complex fonts have a good artistic effect. After the network is trained, it can quickly generate a large number of art fonts.

The other sections are presented below. Section II introduces the baseline of the proposed method: the Pix2Pix network, and the details of the proposed method. In Section III the experimental parameters, experimental results and comparison experiments are described. Finally, in Section IV we summarize the content of the paper and propose future goals.

II. Method

In this section, we first review the image-to-image translation method using Pix2Pix. After that, our proposed method will be presented.

A. Pix2Pix Network

Pix2Pix is a deep neural network model, a type of Generative Adversarial Networks (GANs). It consists of two basic networks, a generator G is used to generate images that are similar to real images and a discriminator D. The original image y is the input of the generator and $G(y)$ is the output of the generator. The image generated by the generator is called a fake image and the image x coming from the dataset is called a real image. The original image is input to both the generator and the discriminator to determine whether the image is real or fake.

In Pix2Pix, the generator uses the U-Net [16], and Fig. 2 shows the structure of this generator. The U-Net is a network consisting of two parts, the encoder, and the decoder. The encoder is composed of fully convolutional layers for downsampling, and the decoder is composed of deconvolutional and convolutional layers for upsampling, each having eight layers. The corresponding skip connection is used between each layer, which is used to keep the image information from being lost due to a large amount of downsampling and upsampling.

On the other hand, Fig. 3 shows the network structure of the discriminator. In Pix2Pix, the discriminator uses a network structure called PatchGAN [7]. The PatchGAN is designed as a fully convolutional network, where the image is not fed into the fully connected layer or activation function after convolutional layers. The input will be mapped into an $N \times N$ matrix using convolution, which is equivalent to the final evaluation value in the original GAN to evaluate the generated image by the generator. Each element of the $N \times N$ matrix represents the evaluation value of a patch in the original image. The average of these values is used as the final discriminant result.

Fig. 4 illustrates the basic network structure of Pix2Pix. The main purpose of the generator is to generate an image $G(y)$ that is similar to the feature of the target domain image. The generator and the discriminator will train each other at the same time. During the training stage, the generator will try to generate images that the discriminator cannot discriminate. The discriminator will try to discriminate between a real image and a fake image, and through this adversarial training, the generator will generate more realistic images. Pix2Pix uses the loss function $\mathcal{L}_{cGAN}$ to achieve the above training objectives.

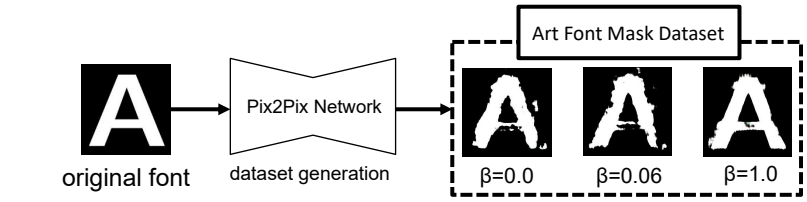
$$\begin{aligned}\mathcal{L}_{cGAN}(G, D) = E_{x,y}[\log D(x, y)] \\ + E_y[\log(1 - D(G(y), y)]\end{aligned} \tag{1}$$

At the same time, in order to make the images generated by the generator quickly approach the real images and achieve higher quality, adding loss function $\mathcal{L}_{l1}$ of paired data sets to the loss function can achieve the goal.

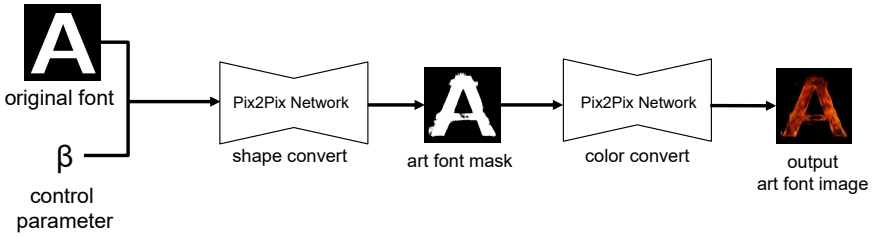
$$\mathcal{L}_{l1}(G) = E_{x,y}[\|x - G(y)\|_1] \tag{2}$$

Finally, $\mathcal{L}_{Pix2Pix}$ is the complete loss function of Pix2Pix. It consists of the two loss functions above.

$$\mathcal{L}_{Pix2Pix} = \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{l1}(G) \tag{3}$$



(a) generating art font mask dataset



(b) generating art font images

Fig. 1. Outline of the proposed method

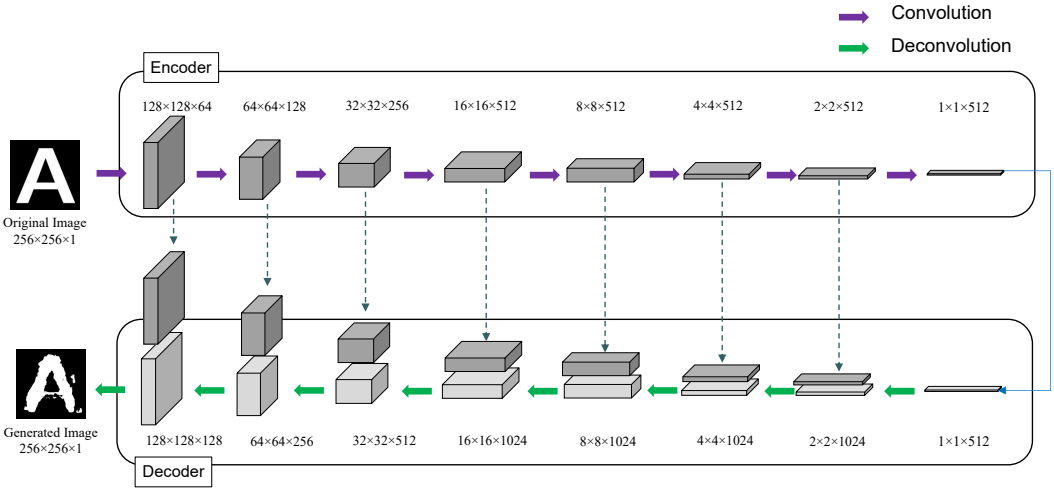


Fig. 2. Generator Network

During training, the generator and the discriminator will try to minimize and maximize the loss function, respectively, in order to obtain an optimal generator.

B. Network Architecture

Three Pix2Pix networks are included in our proposed method. The dataset generation network is used to generate mask datasets. The shape convert network, and the color convert network are used to generate art font.

1) Generate Mask Datasets: The Pix2Pix requires a paired dataset, and the original font does not have a corresponding art font mask as a paired dataset, so it cannot be trained directly. Therefore, we use the dataset generation network for generating the art font mask dataset. The dataset generation network is used where the input is the original font and the output is the transformed art font mask. When training the network, the rough mask and the detailed mask are used as paired datasets to try to add details of the detailed mask to the rough mask. The input is the rough mask and the output is the detailed mask. Use $\mathcal{L}_{mask}$ to make the fake detailed mask close to the real detailed mask.

$$\mathcal{L}_{mask} = \mathcal{L}_{l1}(G) = E_{x,y}[\|x - G(y)\|_1] \tag{4}$$

We use Adobe Photoshop to obtain foreground masks of a reference images, which is an image with a white foreground and a black background. For a detailed foreground mask, use the lasso tool to select the content as carefully as possible to ensure that its details are closer to the reference image. For a rough foreground mask, use the lasso tool to get only the general content of the foreground, with only the foreground outline of the reference image and no detail parts.

When testing, the original font is fed into the network as a rough mask so that it gets the foreground detail of the reference image. Usually, this would make the font shape completely

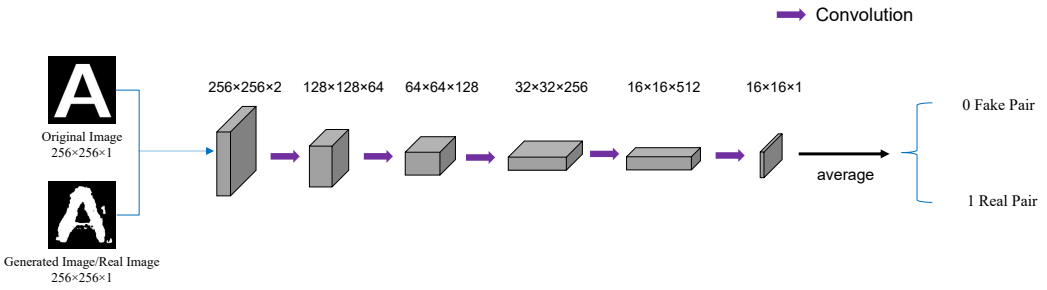


Fig. 3. Discriminator Network

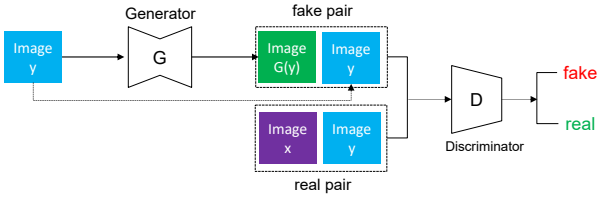


Fig. 4. Pix2Pix Network

collapsed and unreadable. Therefore the original font is also used as part of the network training, using $\mathcal{L}_{font}$ to make the output art font mask close to the shape of the original font. At the same time, the hyperparameter β will control the weight of this loss function. When β is 0, this loss function is disabled, so the output art font mask will be completely collapsed and almost unreadable. While, when β is 1, the art font mask will extremely close to the original input font.

$$\mathcal{L}_{font} = \mathcal{L}_{l1}(G) = E_z[\|z - G(z)\|_1] \tag{5}$$

Fig. 5 illustrates the dataset generation network. $\mathcal{L}_{dg}$ is the complete loss function.

$$\mathcal{L}_{dg} = \mathcal{L}_{cGAN}(G, D) + \lambda\mathcal{L}_{mask} + \beta\mathcal{L}_{font} \tag{6}$$

2) Generate Art Fonts: Two Pix2Pix networks are needed to generate art fonts, the shape convert network and the color convert network. Input the original font into the shape convert network and the output will be an art font mask. Input the output art mask into the color convert network and we will get the final art font.

First, we start with the shape convert network. The shape convert network is used to generate controlled art font masks. It uses the art font masks generated by the dataset generation network and the original fonts as paired datasets. During training, we use parameter β to control the shape of the generated art font masks. The parameter β will correspond to the hyper-parameter β of the dataset generation network, and we have six sets of art font masks datasets. We feed parameter β into the generator along with the original font image, which outputs a art font masks corresponding to parameter β. After the parameter β is input into the network, it will be converted into a 256×256 matrix, and each parameter is filled with β. This matrix will be concatenated with the original font and input into the convolutional layer. We use $\mathcal{L}_{ctrl}$ to make the generated art font masks quickly approach the mask label.

$$\mathcal{L}_{ctrl} = \mathcal{L}_{l1}(G) = E_{x,y}[\|x - G(y, \beta)\|_1] \tag{7}$$

The parameter β will also input into the discriminator with the generated image or the real image. We use the loss $\mathcal{L}_{cGAN}$ to train the network.

$$\begin{aligned}\mathcal{L}_{cGAN}(G, D) = E_{x,y}[\log D(x, y, \beta)] \\ + E_y[\log(1 - D(G(y, \beta), y, \beta)]\end{aligned} \tag{8}$$

The total loss function of the shape convert network is $\mathcal{L}_{sc}$.

$$\mathcal{L}_{sc} = \mathcal{L}_{cGAN}(G, D) + \lambda\mathcal{L}_{ctrl} \tag{9}$$

The color convert network use the reference image and the detailed foreground mask as the paired dataset. During training, its input is the foreground mask and the output is the colored image. During test, the art font mask generated by the shape convert network is used as input, and the output is the colorized art font.

For discrete images, such as leaves, perceptual loss [14] called $\mathcal{L}_{vgg}$ is needed to give the output art fonts a more significant texture. The perceptual loss is a VGG16 [17] network without linear layers and pre-trained with ImageNet, it contains two parts, the $\mathcal{L}_{style}$ and the $\mathcal{L}_{content}$. The total loss of the color convert network is $\mathcal{L}_{color}$. Fig. 6 shows the details of the shape convert network and the color convert network.

$$\begin{aligned}\mathcal{L}_{color} = \mathcal{L}_{cGAN}(G, D) + \lambda\mathcal{L}_{l1}(G) \\ + \alpha\mathcal{L}_{style} + \kappa\mathcal{L}_{content}\end{aligned} \tag{10}$$

III. Experiments

In this section, we will describe about the experimental implementation. After that, we will show the art fonts generated by the experiment and describe them in detail.

In this work, we use the proposed method to convert the original font into an art font with the style of the reference image and able to control the shape of the generated font using the parameter β.

We selected 1107 characters as the original input font dataset. It contains numbers from 0 to 9, English alphabets, kana, and common Chinese characters from *Thousand-Character Classic* [18]. The *Thousand-Character Classic* is an ancient Chinese poem containing one thousand common Chinese characters, each of them is not repeated. We used

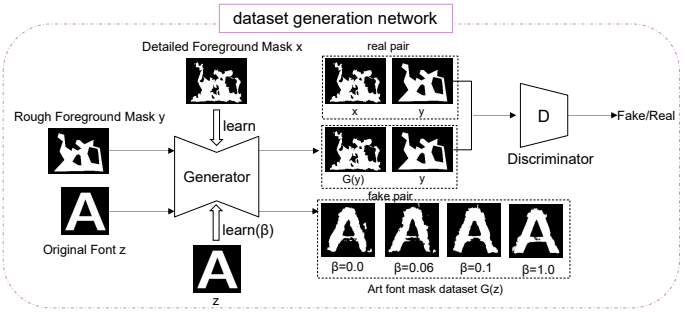


Fig. 5. dataset generation network

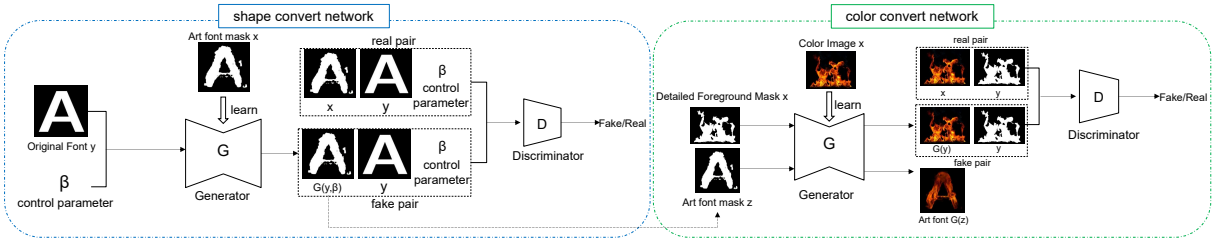


Fig. 6. shape convert network and the color convert network

Adobe Photoshop [19] to render it as 256×256 size images. We collected and photographed different kinds of reference images from the web and manually created a rough foreground mask and a detailed foreground mask for them using Adobe Photoshop. Fig. 7 shows the dataset of fire. For training, we

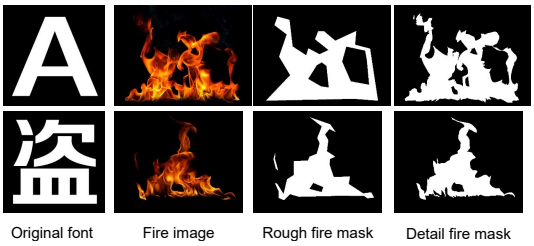


Fig. 7. Original font and reference image dataset

generated the art font mask dataset in Fig. 8 by controlling the hyper-parameter β. The art font mask dataset is used as

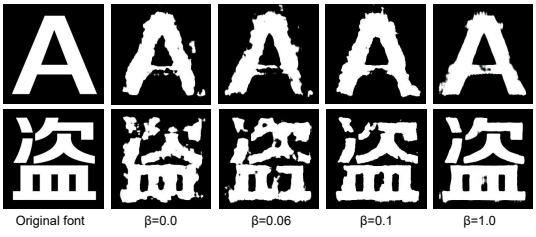


Fig. 8. Art font mask dataset

a paired dataset in the shape convert network training. The detailed foreground mask and fire images are used as the dataset for the color convert network. All the reference images, foreground masks were cropped to 256x256 size images at 128 pixel intervals. We trained the network using PyTorch 1.7.1 [20] with a learning rate of 0.002 and 200 epochs using Adam [21] as the optimize, and the dropout rate is 0.5. The ratio of training set and test set is 9:1.

Fig. 9 shows the final result of the fire art font. Most of the results of the fire art font generation are good. This figure consists of the original font, the generated art font mask with different parameters and the art font.The networks successfully generates masks of the target font and the art font effect. The fire font with different shapes also can be successfully generated under the control of parameter β. This proves the effectiveness of the β as a control parameter.

Our proposed method divides the art font generation into three implementation steps, one for dataset generation and two of which are used for art font generation. To verify the effectiveness of the separation network, we chose one Cycle GAN [22] to directly generate the art fonts. This is due to the fact that Cycle GAN does not require paired datasets and is able to generate target images of different domains in one step. Fig. 10 generates art fonts with clouds as the reference image, and compares the effect of Cycle GAN with that of our proposed method. The results of Cycle GAN are difficult to read. The Cycle GAN does not find which part of the original font needs to be transformed into art fonts, so it is difficult to control the result. Our proposed method can generate easy to read art fonts.

In addition, Fig. 11 was compared for the effect of with or without $\mathcal{L}_{vgg}$ for discrete images, such as leaves, cherry blossom, etc. It can be seen that without using $\mathcal{L}_{vgg}$, although the art font has the color and part of the texture of the reference image, it is not significant, especially the leaf art font. After

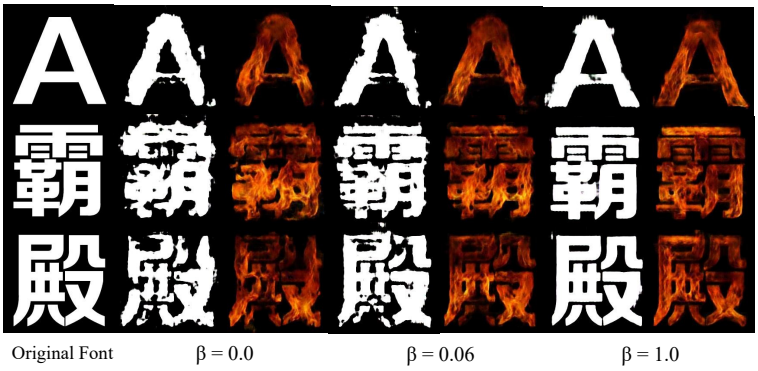


Fig. 9. Experimental results of generating different shapes of fire art fonts using different parameters β

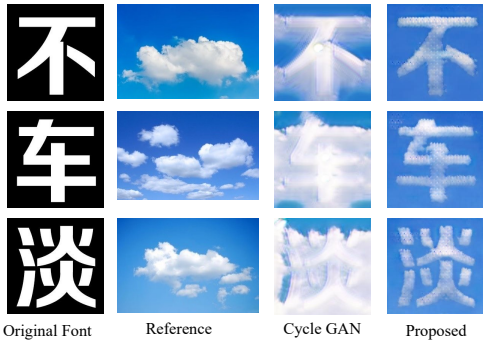


Fig. 10. The comparison of the generated images

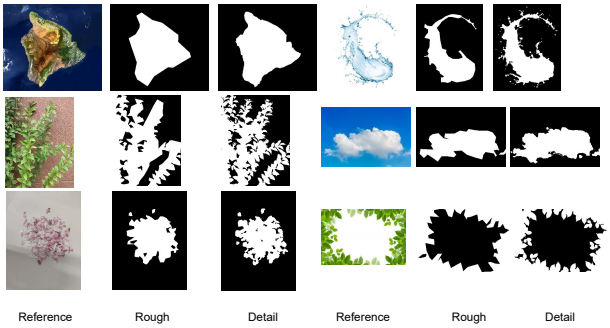


Fig. 12. Reference image dataset

using $\mathcal{L}_{vgg}$, the cherry blossom art font and the leaf art font appear with obvious cherry blossom texture and leaf texture.

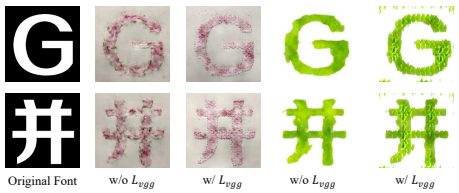


Fig. 11. Compare the effect of with and without $\mathcal{L}_{vgg}$

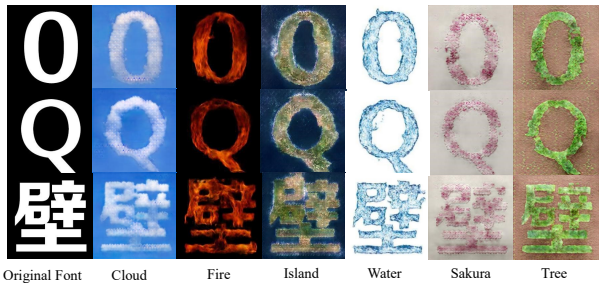


Fig. 13. Art fonts made with multiple reference images

Finally, Fig. 12 shows a variety of reference images we used as datasets, such as water and clouds found from the Internet, as well as shots of cherry blossom and trees on the wall, with multiple images of each type. Fig. 13 shows various art fonts created based on these reference images. The results show that good results were obtained in all these reference images. The features of the reference images can be clearly identified from the generated art fonts while they can be easily read.

IV. Conclusions

We propose a method for generating art fonts from images based on three Pix2Pix networks. First, in order to obtain the paired dataset needed for Pix2Pix, we successfully generated different art font masks using the dataset generation network. Then, we successfully used the shape convert network to generate art font masks, at the same time, we can use β to control the shape of the results. Finally, we successfully used the color convert network to generate art fonts. The network can generate a large number of fire art fonts at one time after training. Most of art fonts have high quality. As the future research topic, we consider adding more functions into this network. For example, converting handwritten fonts to art fonts.

References

[1] S. Yang, J. Liu, Z. Lian, and Z. Guo, "Awesome typography: Statistics-based text effects transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7464–7473.

[2] Y. Gao, Y. Guo, Z. Lian, Y. Tang, and J. Xiao, "Artistic glyph image synthesis via one-stage few-shot learning," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–12, 2019.

[3] Y. Yuan, Y. Ito, and K. Nakano, "Art font image generation with conditional generative adversarial networks," in *2020 Eighth International Symposium on Computing and Networking Workshops (CANDARW)*. IEEE, 2020, pp. 151–156.
[4] S. Yang, Z. Wang, Z. Wang, N. Xu, J. Liu, and Z. Guo, "Controllable artistic text style transfer via shape-matching gan," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4442–4451.
[5] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.
[6] ——, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2414–2423.
[7] C. Li and M. Wand, "Combining markov random fields and convolutional neural networks for image synthesis," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2479–2486.
[8] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.
[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014.
[10] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1501–1510.
[11] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," in *European conference on computer vision*. Springer, 2016, pp. 702–716.
[12] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.
[13] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.
[14] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*. Springer, 2016, pp. 694–711.
[15] A. Xue, "End-to-end chinese landscape painting creation using generative adversarial networks," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 3863–3871.
[16] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.
[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[18] J. T. C. Liu, "The classical chinese primer: Its three-character style and authorship," *Journal of the American Oriental Society*, vol. 105, no. 2, pp. 191–196, 1985.
[19] Adobe, "Photoshop," https://www.adobe.com/products/photoshop.html.
[20] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.
[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[22] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.