electronics

MDPI

Article

Chinese Calligraphy Generation Based on Involution

Yao Song [1,2], Fang Yang [1,2,*] and Te Li [1,2]

[1] Computer Science and Technology, School of Cyberspace Security and Computer, Hebei University, Baoding 071000, China; songyao@stumail.hbu.edu.cn (Y.S.); 20208014007@stumail.hbu.edu.cn (T.L.)
[2] Hebei Machine Vision Engineering Reasearch Center, Hebei University, Baoding 071000, China
* Correspondence: yangfang@hbu.edu.cn

Abstract: The calligraphic works of particular calligraphers often contain only a limited number of characters, rather than the full set of Chinese characters required for typography, which does not meet practical needs. There is therefore a need to develop a complete set of calligraphic characters for calligraphers. Most of the recently popular methods for generating calligraphic characters are based on deep learning, using an end-to-end approach to generate the target image. Deep learning-based methods usually suffer from unsuccessful conversion of stroke structures when the printed font differs significantly from the target font structure. In this paper, we propose an involution-based calligraphic character generation model, which can realize the conversion from printed fonts to target calligraphic fonts. We improve the Pix2Pix model by using a new neural operator, involution, which focuses more on spatial feature processing and can better handle the relationship between strokes than the models using only convolution, so that the generated calligraphic characters have an accurate stroke structure. A self-attentive module and a residual block are also added to increase the depth of the network to improve the feature processing capability of the model. We evaluated our method and some baseline methods using the same dataset, and the experimental results demonstrate that our model is superior in both visual and quantitative evaluation.

Keywords: Chinese calligraphy generation; involution; image translation; generative adversarial

Citation: Song, Y.; Yang, F.; Li, T. Chinese Calligraphy Generation Based on Involution. *Electronics* **2022**, *11*, 2201. https://doi.org/10.3390/electronics11142201

Academic Editors: Xiushan Nie, Guoqiang Zhong, Yongshun Gong, Bin Fan and Xin Li

Received: 5 June 2022
Accepted: 11 July 2022
Published: 14 July 2022

1. Introduction

Chinese calligraphy is a crystallization of the wisdom of the Chinese people, an important part of national culture and a very unique visual art. The beauty of the melody, form and mood of Chinese calligraphy has been increasingly valued by the design community and is widely used in the fields of graphic design, stone carving and product design. However, none of the extant calligraphic works can contain a full set of commonly used characters to meet practical needs. Unlike the English alphabet, the number of Chinese characters is very large—three thousand even for commonly used characters—and it is not easy to rely on human effort to imitate a calligrapher's style to complete a complete font library, and it is very difficult to become proficient in a calligraphy discipline, requiring years of practice. Therefore, it was necessary to devise a method of automatically generating calligraphic fonts for calligraphers to generate a complete set of Chinese characters required for typography, making it easy for designers to use and for calligraphy enthusiasts to learn.

There are two broad approaches to automatically generating calligraphic fonts, and one is to build a stroke library by breaking down Chinese characters into strokes and then using the strokes in the library to build new calligraphic characters [1,2]. This method simply pieces together the split strokes to form calligraphic characters, ignoring the overall structure of the font and relying heavily on the extraction effect of the strokes. When the structure of the character is too complex, the stroke extraction technique does not achieve good results and relies on manual extraction, which greatly increases the workload. The other is to use deep learning to achieve the conversion of standard fonts to calligraphic

fonts. The printed font is used as input, its feature representation is extracted and feature transformation is performed in the feature space to finally complete the mapping to the calligraphic font. However, when the source fonts differ greatly from the calligraphic fonts in style, the conversion results are still unsatisfactory.

In this paper, we consider the generation of calligraphy as an image-to-image translation problem and propose a Pix2Pix-based neural network model. The model consists of a generator and a discriminator. We add residual blocks to the generator to deepen the network depth and improve the feature processing capability of the model, as well as a self-attention module to ensure the integrity of the generated font strokes. In order to achieve a better mapping from printed font to calligraphic fonts, we use a new neural operator, involution, in the generator instead of the traditional convolution operation. As our model is based on a neural network, the parameters of the model are formed by the network as it continues to learn, and our model minimizes manual manipulation compared to the stroke construction approach. Our main contributions are as follows.

We propose an involution-based calligraphy generation model that can directly convert Chinese character printed font images into target calligraphic font images without extracting strokes, and font content and style can be learned simultaneously without separate training. Compared with some baseline methods [3,4], our model can generate higher quality calligraphic images.

2. Related Work

The following subsections describe the links and differences between traditional image conversion and calligraphy generation, and discuss some traditional baseline methods as well as existing methods related to calligraphy generation.

2.1. Generative Adversarial Networks

Generative adversarial networks, proposed by [5], have revolutionized the field of deep learning and have also attracted a lot of attention in the field of computer vision, where they have been developing rapidly. The GAN is commonly used for unsupervised learning, which transforms images from one domain to another without labeling data, and it excels in image generation and processing. A number of GAN-based methods have been proposed to solve various image processing problems. In recent years, bigGAN [6] and styleGAN [7] have been proposed to design huge networks to generate higher quality and higher resolution images. These methods map noise to images and the input does not contain specific conditional information. However, in some image translation tasks it is required that the image must be used as input and transformed into the target image domain using the given image as conditional information [8]. Some new extension architectures for GANs have been proposed, such as Pix2Pix [3] and cycleGAN [4].

2.2. Image-to-Image Translation

The process of converting an image from the source domain to the target domain is called image-to-image conversion and involves areas such as medical image processing [9], semantic segmentation [10] and image style conversion [11].

The Pix2Pix [3] model is applicable to many image-to-image translation problems. The model takes images as input and uses pairs of images for training to achieve image style migration. Although Pix2Pix style migration works well, the pairing process is tedious and time-consuming due to the large amount of paired image data required for model training, and even in some special cases it is impossible to obtain paired images, and these shortcomings limit the promotion and application of Pix2Pix.

CycleGAN [4] is widely used for tasks where ground truth does not exist because it does not require a paired training set. The network contains two pairs of generative adversarial networks that enable bi-directional domain transformation, and the invoked cyclic consistency loss removes the pairwise constraint between domains, which allows cycleGAN to transform image styles even without a paired dataset. However, the cyclic

consistency loss and the pixel-by-pixel difference loss of the images used by cycleGAN allow the content information of the generated images to be over-represented and do not allow for good style migration. Sanakoyue et al. [12] introduced a style-aware content loss in GAN that allows the model to better learn the target style. cycleGAN uses two pairs of generators and discriminators in order to train without pairing, which significantly increases the time cost and computational cost required to train the model, especially in one-to-many style transformation tasks. To compensate for this drawback, Choi et al. proposed starGAN [13], a model that accepts multiple target style attributes, increases domain classification loss and learns mappings between multiple domains using only one generator and one discriminator, enabling one-to-many image style conversion.

In these tasks mentioned above, the image-to-image conversion problem is usually described as a pixel-to-pixel conversion problem, where the input image and the target image have the same underlying structure without any distortion. However, in the calligraphic character generation task, the input standard image and the target calligraphic image only have similar relative layouts, the position and style of the strokes are not the same and distortion is inevitable in the implementation of the font conversion process, so the above methods are not well-suited to the calligraphic character generation task.

2.3. Chinese Character Generation

Zi2zi [14], proposed in 2017, is an early approach to using generative adversarial learning for Chinese character font generation, which adds category embedding to the Pix2Pix model for one-to-many modeling, enabling the model to handle many different styles simultaneously; to avoid the model confusing multiple styles, it draws on the multi-class in AC-GAN. Zi2zi performs very well in some conventional font generation.

Zhou et al. [15] added a series of residual blocks and a content complementary network to the cycleGAN model to implement the conversion of printed font to calligraphic fonts. Li [16] et al. proposed a F2PNet model inspired by cycleGAN to implement the conversion of printed font to flower and bird art characters. Chen et al. [17] proposed a model that can generate multiple styles of fonts simultaneously with reference to starGAN, and can fuse existing styles to generate a new style font. Zheng et al. [18] argued that each Chinese character includes a style factor and a content factor, and they mapped two kinds of Chinese character images with style features and content features, respectively, and then used the generator to perform style and content. The fusion of style and content was then performed by a generator to generate the target style of Chinese characters. The model is able to generate realistic Chinese characters with strong generalization capabilities. It is worth noting that the structure of the generated calligraphic characters, whether trained using the cycleGAN model or the starGAN model, is very dependent on the input print, and is not generated well when using certain calligraphic fonts that differ significantly from the printed font structure.

Jiang et al. [19] divided font generation into two parts: skeleton synthesis and font style rendering, and used CNN to convert the skeleton of the reference font into the skeleton of the target style, making the final generated font free from the structure of the original font. The accuracy of the fonts generated by this method is of good quality, but its network type is too complex. Lyu et al. [20] added a monitoring network to Pix2Pix to encode and decode the target calligraphic characters, and fuse the features of the target calligraphic characters obtained from the decoding with the features of the print, using the target calligraphic character features to supervise the print features, so that the generated calligraphic fonts are to a certain extent free from the print structure. In addition, this author added reconstruction loss to the monitoring network to ensure the accuracy of the calligraphic character features obtained by decoding.

3. Methodology

We propose an involution [21]-based calligraphy generation model for Chinese characters, which is based on the Pix2Pix model. The generator of the model is a U-Net network

structure [22], and we used involution for all layers except the outermost layer of the network, and added residual blocks and self-attentive modules to the middle layer of the U-Net network. The discriminator uses patchGAN, which learns the distribution of target styles and thus guides the generator to generate more realistic images. In 3.1, we introduce the principle of involution and compare it with convolution. The next subsections describe our proposed model architecture in detail.

3.1. Involution

Most generative adversarial network-based image generation models are superimposed by convolution operations, but convolution can suffer from several shortcomings in use: (1) the convolution kernel of a CNN shares parameters at all spatial locations, which will limit the ability to model local space at different spatial locations and cannot effectively capture distant relationships in space; and (2) the parameters within the convolution kernel are randomly generated and there is no relationship between them and the input features.

Involution is a complete reversal of the convolution mindset:

1. Whereas convolution shares parameters in space, involution uses different kernels at different locations in space.
2. Whereas convolution uses different kernels for different channels, involution shares parameters.
3. The parameters within the convolution kernel are randomly generated and are independent of the input feature information, whereas the involution kernel is formed by mapping the input to the features.

Taking these points together, it can be seen that involution focuses more on spatially scoped feature processing.

Involution's kernels can be set to different sizes as required, and as the kernels become larger, the field of view becomes larger, making it easier to capture relationships between features at a distance. In addition to this, channels can be grouped so that kernels are shared between channels within a group and different kernels are used for different channel groups. When the number of groups is 1, it means that the whole channel uses one kernel, and when the number of groups is equal to the number of channels, it means that each channel has its own corresponding kernel. The number of groups can be adjusted as required.

3.2. Model Design

3.2.1. Generator

The generators are mainly structured as a U-Net network [22], which is a combination of an encoder and a decoder with additional jump links to preserve the low-dimensional features and avoid the loss of valid information. However, due to the lack of depth of the network, a good result is not achieved when handling the calligraphic character generation task, and the generated calligraphic characters suffer from serious stroke errors and blurred edges. The task of generating calligraphic characters is to achieve the conversion from printed Chinese characters to calligraphic Chinese characters, in which the character structure and position need to change, and a simple coding and decoding structure cannot meet the demand. In order to increase the depth of the network and enhance its feature processing capability, we added a residual network consisting of a series of residual blocks [23] to the middle layer of the U-Net network. The use of this residual network both increases the depth of the network and avoids the problem of gradient disappearance due to the network being too deep.

However, adding a residual network to the intermediate layer alone only generates roughly the target calligraphic characters, and there is still some degree of stroke loss. For this reason, we tried to continue adding a self-attentive module [24] to the middle layer to capture the internal correlation of the data features. The combination of the residual network and the self-attentive module was experimentally found to be effective in improving the stroke problem. At this point the generated calligraphic characters already had some accuracy and legibility, but the font would appear somewhat distorted and

wavy. We believe that this is a problem of insufficient spatial feature processing, where convolution shares convolution kernels at all spatial locations and cannot effectively capture distant relationships in space, whereas involution, mentioned in 3.1, is very concerned with spatial range on feature processing and can adaptively assign weights so that the most informative visual elements in the spatial domain are prioritized. Therefore, we replaced some of the convolution modules in the generator with modules containing involution.

We set the size of the involution kernel to 7×7, the number of groups G to 4 and the compression rate r to 4 according to the experimental data given in [21]. The principle of involution is shown in Figure 1. All channel features under a certain pixel point in the input features are first extracted, and then the channel compression is performed by r, with the aim of reducing a certain amount of computation. Then involution will take the compressed channels and expand them again to a size of $K \times K \times G$. These features are then reshaped to generate G kernels of size $K \times K$. Finally, these kernels are multiplied and summed with the features in the $K \times K$ range around the pixel point just selected, resulting in features of the same size and number of channels as the input features.

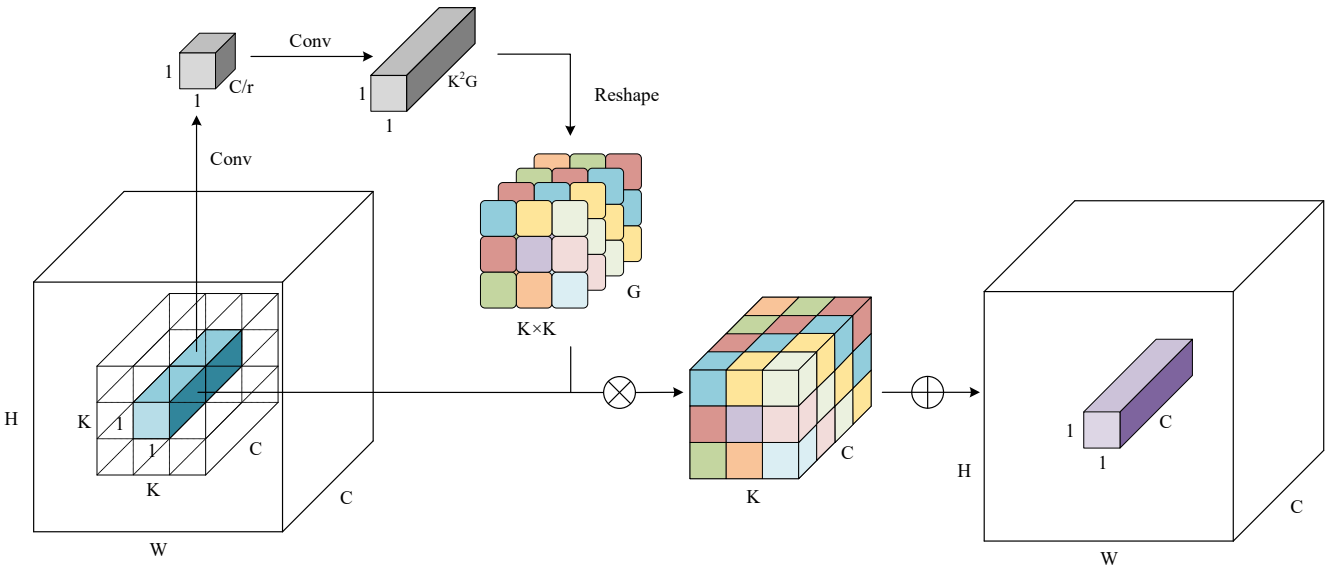

Figure 1. The blue block is the feature vector to be processed and the compressed purple block is the feature vector after involution. h and w are the height and width of the feature, respectively, C is the number of channels, r is the compression rate, $K \times K$ is the kernel size and G is the number of groups. The feature vectors to be processed are first compressed by convolution, then expanded by convolution, and then reshaped into G kernels of size $K \times K$. The generated kernels are dot product and summed with the feature vectors to be processed and the feature vectors in their surrounding $K \times K$ range to obtain the processed feature vectors.

The number of feature channels processed by involution is constant. In order to adapt to the change of channels in the U-Net network and to apply it better in the upsampling and downsampling process, we add a convolution operation after each involution operation, as shown in Figure 2, and the involution block consists of Relu-involution-conv-BN together. In addition, in the outermost layer of the network we do not use the involution block but the convolution block containing Relu-conv-BN. The grey block and the blue block together form the U-Net network, the purple block added in the middle layer is the self-attentive block and the orange block is the residual block. The blue lines indicate jump links, which place the low-dimensional features encoded by the encoder into the decoder to be merged with its feature vector and decoded together in the next step. The addition of jump links allows valid features to pass directly through the network layer, preserving low-dimensional features and avoiding spatial redundancy of information.

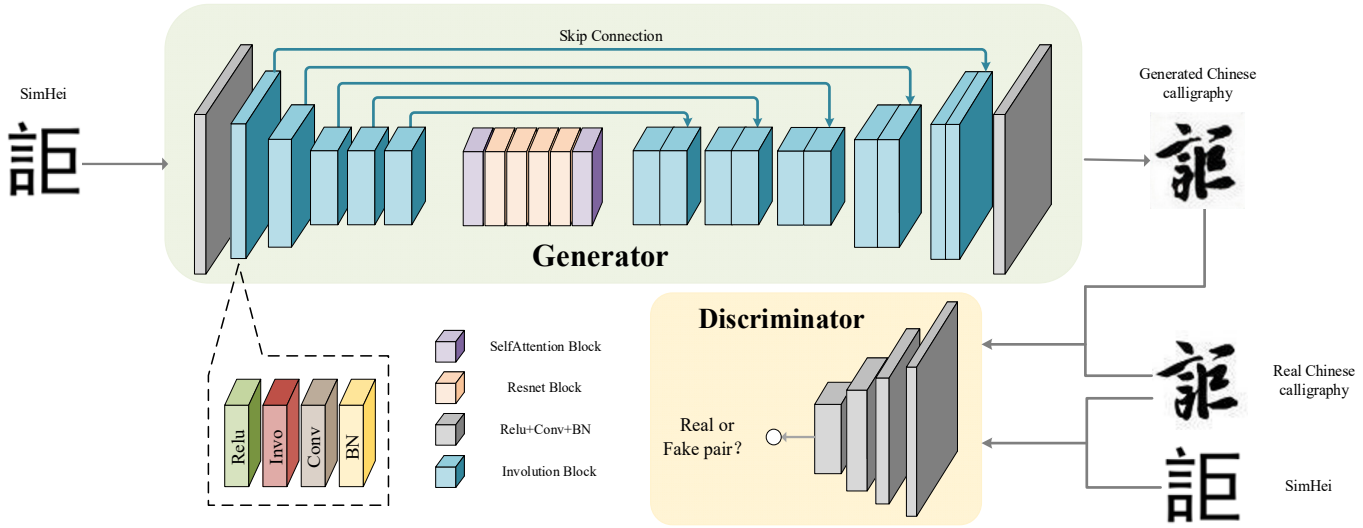

Figure 2. The generator is used to convert the printed fonts (SimHei) to the target calligraphic font. The discriminator is used to determine the authenticity of the image. The blue block is the involution block, the gray block is the convolution block, the purple block is the self-attentive block and the orange block is the residual block.

3.2.2. Discriminators

The discriminator we use is patchGAN, where the printed font image and the real calligraphic font image form a real image pair, and the print image and the generated calligraphic font image form a false image pair. These two pairs are given to the discriminator to discriminate and give predicted values. PatchGAN is able to maintain a certain high resolution and detail in the style migration task, and also ensures that there is a correspondence between the input image and the output image. There is a correspondence between the input and output images.

3.3. Loss Functions

We train the network in an end-to-end way. Given a pair of samples (x, y), x and y represent the printed and calligraphic fonts of the same Chinese character, respectively.

The loss of the generator includes both adversarial loss and pixel loss. Adversarial loss can be expressed as:

$$\mathcal{L}_{advG} = -\mathbb{E}_{x\sim P_{data(x)}}[\log \mathrm{D}(\mathrm{G}(x), x)] \tag{1}$$

The generator G aims to learn the mapping G: $x \rightarrow y$, and hand over the printed font x and the generated calligraphy font $\mathrm{G}(x)$ to the discriminator D. The output of the discriminator D is a probability value taking values from 0 to 1. When D $(\cdot) = 0$ it means that the probability is minimal and the discriminator considers $\mathrm{G}(x)$ to be a false calligraphy font; When D $(\cdot) = 1$ it indicates that the probability is maximum and the discriminator considers $\mathrm{G}(x)$ to be a true calligraphy font. The generator expects the discriminator to judge $\mathrm{G}(x)$ as true, and the output is close to 1.

In order to ensure that the generated calligraphy font $\mathrm{G}(x)$ is closer to the real calligraphy font y, a pixel loss is also added, represented as follows:

$$\mathcal{L}_{pixel} = -\mathbb{E}_{x,y}[\left|\left|y - \mathrm{G}(x)\right|\right|_1] \tag{2}$$

The loss of the discriminator is likewise divided into two parts, the cross-entropy loss between the printed font x and the real calligraphic font y, and the cross-entropy loss between the printed font and the calligraphic font $\mathrm{G}(x)$ generated by x, denoted as follows:

$$\mathcal{L}_{advD} = \mathbb{E}_{x\sim P_{data(x)},\, y\sim P_{data(y)}}[\log \mathrm{D}(x,y)] + \mathbb{E}_{x\sim P_{data(x)}}[\log(1-\mathrm{D}(x,\mathrm{G}(x)))] \quad (3)$$

The data is generated during the adversarial process between the generator and the discriminator, and the two of them play each other so that the model is constantly updated in order to generate a better calligraphic picture. Equation (1) can be merged with Equation (3) as the loss of the whole model in the confrontation. The goal of the generator is to minimize it, while the goal of the discriminator is to maximize it. The display is as follows:

$$\mathcal{L}^* = \arg\min_{\mathrm{G}}\max_{\mathrm{D}} \mathcal{L}_{advD} \quad (4)$$

The final objective is:

$$G^* = \arg\min_{\mathrm{G}}\max_{\mathrm{D}} \mathcal{L}_{advD} + \lambda\mathcal{L}_{pixel} \quad (5)$$

Here λ controls the weight of the item.

4. Experiment

4.1. Experiment Setting

4.1.1. Data Preparation

Chinese calligraphy is divided into five main types: cursive, seal script, running script, clerical script and regular script. The cursive script, with its continuous strokes and gracefulness in the midst of chaos, is ornamental, and the structure of the cursive script differs considerably from that of the printed form, making it a better test of the model's performance. Running script and regular script are more standardized and closer to printed font, and are the two calligraphic types most used in everyday writing. Therefore, these three calligraphic types were selected to build the dataset in this paper. An example of calligraphic work is shown in Figure 3. To demonstrate the generalization of the model, the four calligraphers chosen for this paper all have different styles of work. The collection of calligrapher's font images is a tedious task, and due to the limited number of characters involved in the original calligraphic works, we were able to collect a smaller number of unduplicated calligraphic characters, which led to the small size of our dataset. Each calligrapher's work contains approximately 2000 images. A random selection of 1800 images were used as the training set and the remaining 200 images were used as the test set. The images in the dataset are 64 × 64 pixels and all input images are expanded to 256 × 256 before entering our model.

Figure 3. Examples of four different calligraphic works in the dataset.

4.1.2. Network Architectures and Optimizer

The number of filters in the encoder were 64, 128, 256, 256 and 256 and the decoder was structured in the reverse order of the encoder. The Relu in the encoder was leakyRelu [25], with a slope of 0.2. The decoder was normal ReLU except for the last layer, which had an activation function of Tanh [26], which mapped the resultant values between -1 and 1. The type of batch normalization was instance normalization [27]. The number of groups was G = 4 in involution, the compression ratio was r = 4, and the size of the kernel was K = 7. The discriminator was a 70×70 patchGAN [3]. The model was implemented in Pytorch. In our experiments, the initial learning rate was 0.0002, the batch size was 1 and the optimizer was Adam [28]. The value was 100.

4.1.3. Evaluation Metrics

In addition to relying on vision to determine the accuracy of the font, we also calculated the Structural Similarity Index (*SSIM*) and the Peak Signal to Noise Ratio (*PSNR*) to assess the performance of the model:

1. *SSIM* is used to measure the structural similarity between the real image and the generated image. It consists of three components: structural correlation loss, luminance distortion and contrast distortion. *SSIM* is expressed as:

$$SSIM(R,F) = \sum_{r,f} \frac{2\mu_r\mu_f + C_1}{\mu_r^2 + \mu_f^2 + C_1} \times \frac{2\sigma_r\sigma_f + C_2}{\sigma_r^2 + \sigma_f^2 + C_2} \times \frac{\sigma_{rf} + C_3}{\sigma_r\sigma_f + C_3} \tag{6}$$

where R and F indicate the real image and the generated image, respectively. r and f denote the image patch of R and F in a local window of size h $\times$ w. μ_r and μ_f present the mean values of corresponding image patches. σ_r and σ_f indicate the standard deviation of real and generated image patches, respectively. The range of *SSIM* is [0, 1]. The higher the similarity between the real image and the generated image, the closer the value of *SSIM* is to 1.

2. *PSNR* is the most common and widely used objective measure of image quality. It is the logarithm of the mean squared error *MSE* between the real image and the generated image relative to $(2^n - 1)^2$. The formula is as follows:

$$PSNR = 10 \times log_{10}(\frac{(2^n - 1)^2}{\text{MSE}}) \tag{7}$$

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} I(i,j) - K(i,j)^2 \tag{8}$$

where I and K denote the reference image and target image, respectively, representing pixel points, and m and n are the length and width of the image.

4.2. Ablation Study

4.2.1. Effect of Involution

We compared the models with and without involution, and the results are shown in Figure 4. The calligraphic characters generated by the model without involution show distorted strokes and a few missing strokes. The addition of involution effectively improved the distortion of the strokes and improved the accuracy and readability of the font. The results of the quantitative analysis are shown in Table 1, with both SSIM and PSNR improving with the addition of involution.

Figure 4. The results of our model with/without involution block evaluated on Wang Xizhi subset.

Table 1. Quantitative evaluation of the performance of the model with/without involution block.

	SSIM	PSNR
Without Involution Block	0.64	13.7
With Involution Block	0.70	17.1

4.2.2. Effect of Res-SA Block

We considered the residual block and the self-attentive block as a whole, called the Res-SA Block. The model with the Res-SA block removed was compared with our model, and the results are shown in Figure 5. The calligraphic characters generated by the model without the addition of the Res-SA block had some incorrect connections between strokes that should not occur during correct writing. In contrast, the addition of the Res-SA block improved these problems. The data in Table 2 also illustrate the ability of the Res-SA block to improve the quality of the generated images.

Figure 5. The results of our model with/without Res-SA block evaluated on Wang Xizhi subset.

Table 2. Quantitative evaluation of the performance of the model with/without Res-SA block.

	SSIM	PSNR
Without Res-SA Block	0.67	16.8
With Res-SA Block	0.70	17.1

4.2.3. Effect of Input Fonts

KaiTi has its own stroke style, which is closer to Wang Xizhi's calligraphic style than SimHei. In order to test the effect of the input font on the model generation effect, we used KaiTi and SimHei as input fonts and Wang Xizhi's calligraphic characters as target fonts, respectively, to train the model. The test results are shown in Figure 6. The results of the quantitative analysis are shown in Table 3. The experiments showed that when the input font is KaiTi, all three models produce calligraphic images with improved accuracy of strokes, the most obvious performance being cycleGAN. The results of the quantitative analysis also show that using KaiTi as the input font produces higher quality calligraphic characters. It can be concluded that the input font has an impact on the results, and the closer the input font is to the target font style, the better the model generates images, but the degree of impact varies between models.

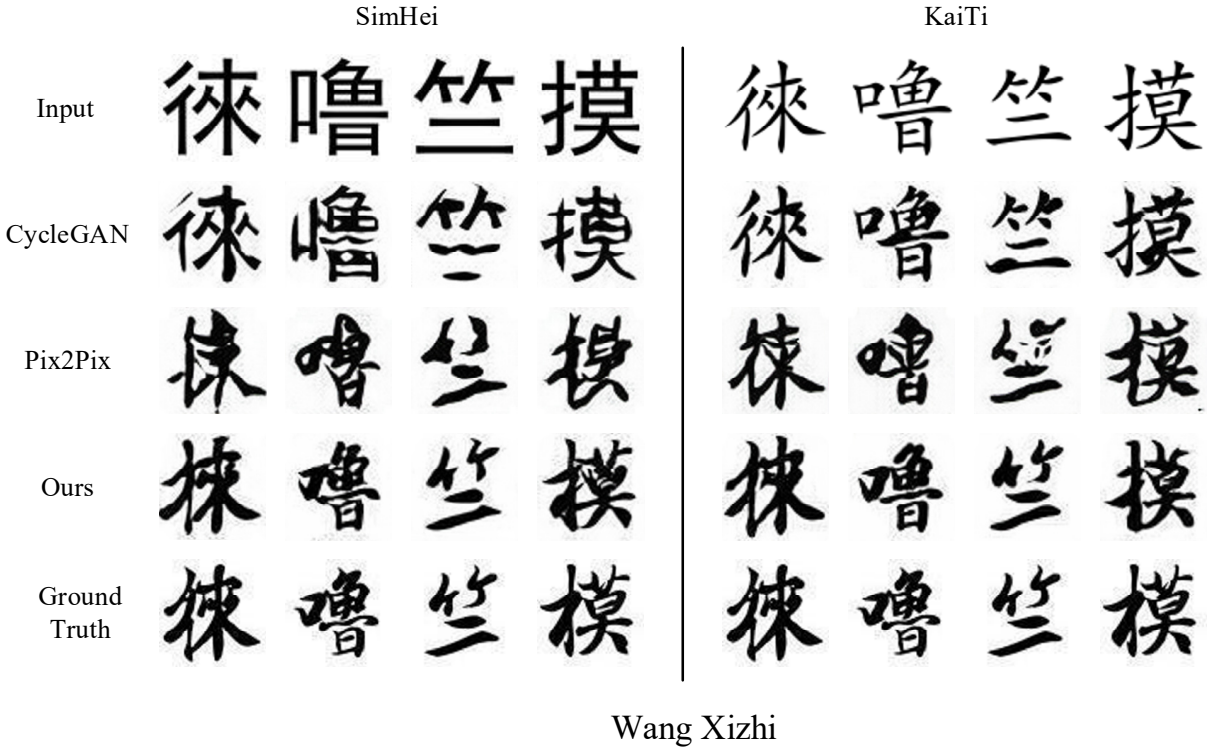

Figure 6. Generated results on the Wang Xizhi subset when using KaiTi and SimHei characters as input.

Table 3. Quantitative evaluation of the impact of input font on the model.

Input Font	CycleGAN		Pix2Pix		Ours	
	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR
SimHei	0.51	9.6	0.62	13.4	0.70	17.1
KaiTi	0.58	11.0	0.63	14.1	0.73	18.5

4.3. Comparison with Existing Methods

We chose two baseline methods to compare with our method, one is Pix2Pix and the other is cycleGAN, both of which are representative in the area of style conversion. We

chose SimHei as the input font because it has consistent stroke thickness and does not contain any font style. Experimental data in Section 4.2.3 can show that when the input font is stylistically similar to the target font, it affects the quality of the generated font and hence the judgement of the validity of the model. Therefore, choosing SimHei as the input font will prevent the input font style from affecting the generated calligraphic characters and thus test the performance of the model more accurately.

4.3.1. Qualitative Comparison

The images of calligraphic characters generated by the baseline method and our proposed method are shown in Figure 8. The performance on the four different datasets shows that our method generates a higher quality image. The specific analysis is as follows:

1. CycleGAN: CycleGAN is able to effectively capture the style of the target calligraphic fonts, for example, Liang Qiusheng's font has sharp strokes at one end and rounded strokes at the other, and Guan Jun's font has thin strokes, which are reflected in the generated calligraphic fonts. However, the mapping of the font structure is poor, unable to escape the constraints of the printed font structure, and there is a certain degree of missing stroke problem.
2. Pix2Pix: Compared to cycleGAN, Pix2Pix leans more towards the target calligraphic font in terms of stroke structure, and the stylistic features are better represented. However, the edges of the strokes are blurred. Guan Jun's results show that the generated fonts have more stroke errors. Wang Xizhi's results show that the generated fonts have more serious problems such as strokes sticking together and blurred edges.
3. Ours: Compared to cycleGAN, our model produces calligraphic characters with more accurate font structure and a more pronounced calligraphic style; compared to Pix2Pix, the edges are clear and free of adhesion, which is more effective. Overall, our model has an advantage over the baseline model in terms of both style and font structure. As Sun Guiting's stylistic features are most different from those of printed scripts, and there are cases where the strokes are contiguous, the model is relatively less effective in learning this type of calligraphy, resulting in a small number of stroke errors in the resulting script, but this does not affect legibility.

4.3.2. Quantitative Comparison

We calculated the corresponding SSIM and PSNR values derived from the performance of the different models under the four datasets and the results are shown in Table 4. The bolded numbers are the best generation results achieved by the three models for different styles. It can be seen that our model outperforms cycleGAN model and Pix2Pix model.

Table 4. Quantitative evaluations of our method and other two methods.

Method	Sun Guoting		Liang Qiusheng		Guan Jun		Wang Xizhi	
	SSIM	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**
CycleGAN	0.53	9.4	0.53	10.1	0.57	11.1	0.51	9.6
Pix2Pix	0.63	11.3	0.64	13.5	0.67	14.6	0.62	13.4
Ours	**0.65**	**11.4**	**0.73**	**18.2**	**0.78**	**19.9**	**0.70**	**17.1**

In addition, we selected partial test results to generate line graphs of their SSIM and PSNR, and the results are shown in Figure 7. In Guanjun, our model performs best, significantly better than the other two. Additionally, in Sun Overdrive, the difference with Pix2Pix is smaller, with the two generating essentially the same results for certain characters.

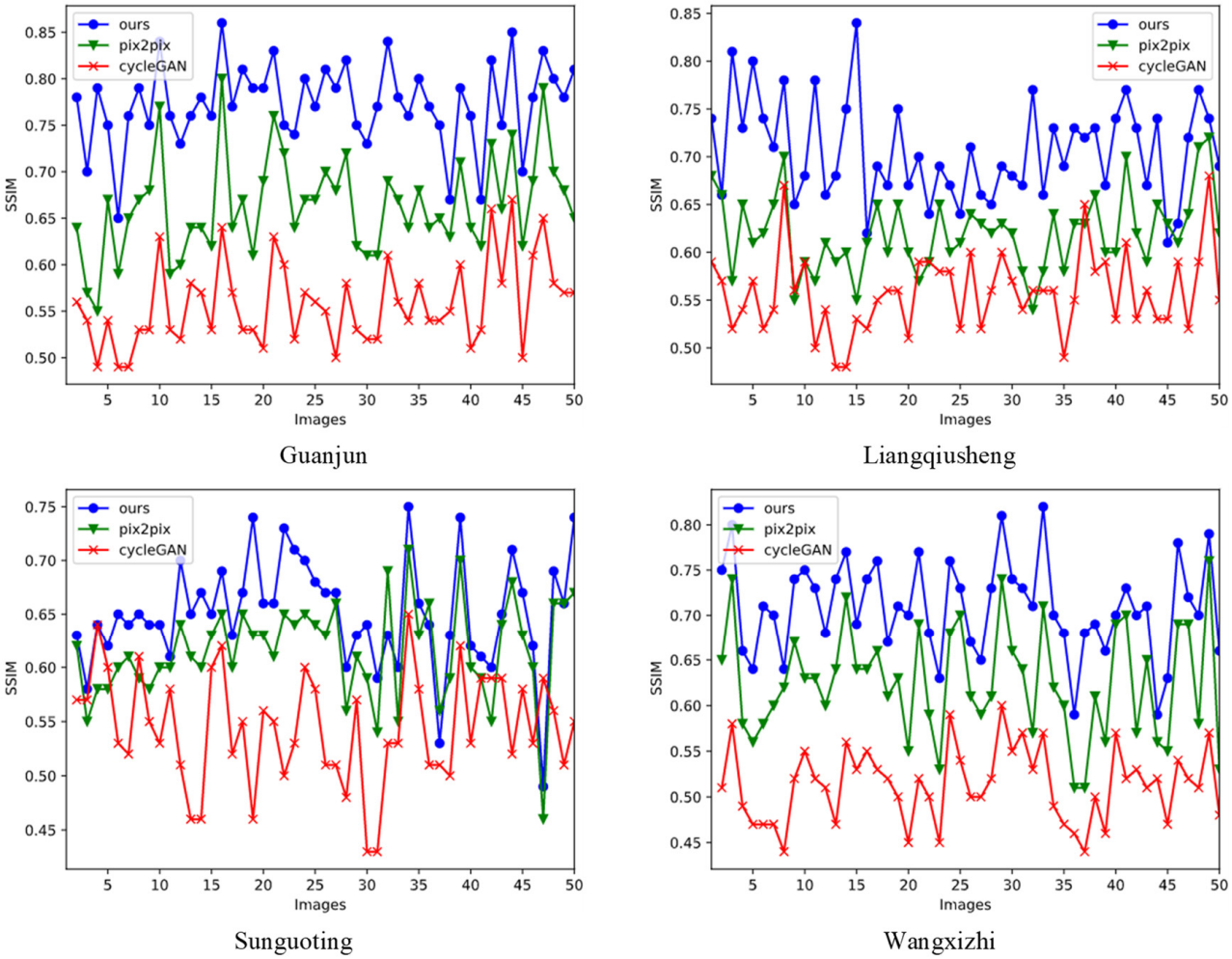

Figure 7. Partially generated image SSIM value line graph.

The model behaves differently in different styles of calligraphic characters, not only in relation to the stylistic similarity between the target calligraphic font and the input font, but also in relation to its font structure. In general, our model can be applied to different styles of calligraphic characters and outperforms the other two classical models.

4.4. Model Analysis

The GAN loss and L1 loss curves for the training process are given in Figure 9. It can be seen that the GAN loss was constantly fluctuating, indicating that our network was well-trained, whereas the L1 loss was continuously decreasing until it was stable, indicating that our reconstructed image continued to be close to the real image as the training cycle increased.

We give the results obtained from testing at different epochs using the Wang Xizhi dataset, as shown in Table 5. As the epoch increased, the quality of the calligraphic fonts generated by the model increased, reaching the best when epoch = 250.

Table 5. Performance of the model under different epochs.

Epoch	SSIM	PSNR
50	0.638	14.42
100	0.671	16.07
150	0.687	16.45
200	0.689	16.61
250	**0.702**	**17.13**
300	0.702	17.12

Figure 8. Example results of the baseline methods and our method on the same dataset.

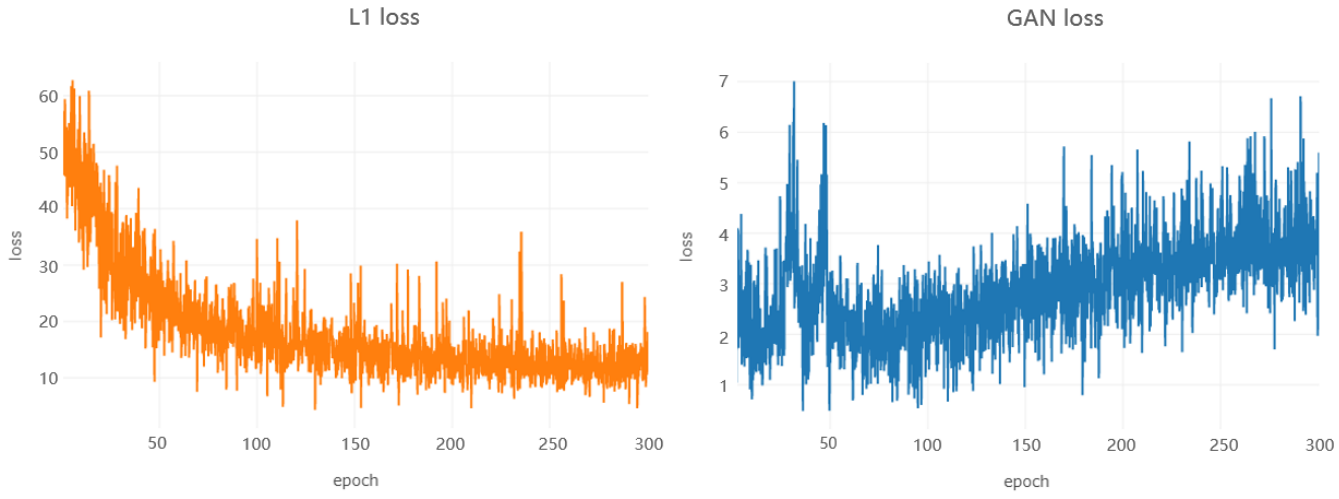

Figure 9. The curve of the L1 loss and GAN loss as epoch increased.

5. Conclusions

In this paper, we propose a calligraphic font generation model that can directly transform printed font images into calligraphic font images in the target style. The model is based on Pix2Pix, and in the generator we use a new neural operator, involution, instead of traditional convolution. Involution is able to focus on the relationship between spatial

distances, ensuring that the generated font strokes are structurally correct while enhancing the stroke details. In addition, we added a self-attentive module and a series of residual blocks to the middle layer of the generator to increase the network depth of the model and give it better feature processing capabilities. We tested the model on four different styles of calligraphy datasets. The experimental results show that the method has advantages over other classical models in both visual perception and quantitative evaluation. In future work, we will consider imbuing the generated calligraphic characters with brush and ink textures to make the strokes more realistic and vivid.

Author Contributions: All authors contributed to this work. Writing—original draft preparation, Y.S.; writing—review and editing, F.Y.; supervision, T.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported and funded by the Science and Technology Project of Hebei Education Department (No. ZD2019131).

Institutional Review Board Statement: Not applicable.

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

References

1. Zong, A.; Zhu, Y. Strokebank: Automating personalized chinese handwriting generation. In Proceedings of the AAAI Conference on Artificial Intelligence, Québec City, QC, Canada, 27–31 July 2014; pp. 3024–3029.
2. Li, W.; Chen, Y.; Tang, C.; Yu, S. Example-based chinese calligraphy synthesis. In Proceedings of the 2018 International Conference on Advanced Control, Automation and Artificial Intelligence, Shenzhen, China, 21–22 January 2018.
3. Isola, P.; Zhu, J.-Y.; Zhou, T.; Efros, A.A. Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1125–1134.
4. Jun-Yan, Z.; Taesung, P.; Phillip, I.; Alexei, A.E. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2223–2232.
5. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 2672–2680.
6. Brock, A.; Donahue, J.; Simonyan, K. Large scale gan training for high fidelity natural image synthesis. *arXiv* **2018**, arXiv:1809.11096.
7. Karras, T.; Laine, S.; Aila, T. A Style-Based Generator Architecture for Generative Adversarial Networks. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 12–20 June 2019.
8. Mehdi, M.; Simon, O. Conditional generative adversarial nets. *arXiv* **2014**, arXiv:1411.1784.
9. Kaji, S.; Kida, S. Overview of Image-to-Image Translation Using Deep Neural Networks: Denoising, Super-Resolution, Modality-Conversion, and Reconstruction in Medical Imaging. 2019. Available online: https://kyushu-u.pure.elsevier.com/ja/publications/overview-of-image-to-image-translation-by-use-of-deep-neural-netw (accessed on 10 June 2019).
10. Souly, N.; Spampinato, C.; Shah, M. Semi supervised semantic segmentation using generative adversarial network. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 5688–5696.
11. Zhang, F.; Gao, H.; Lai, Y. Detail-preserving cyclegan-adain framework for image-to-ink painting translation. *IEEE Access* **2020**, *8*, 132002–132011. [CrossRef]
12. Sanakoyeu, A.; Kotovenko, D.; Lang, S.; Ommer, B. A style-aware content loss for real-time HD style transfer. In Proceedings of the European Conference on Computer Vision, Munich, Germany, 8–14 September 2018; pp. 698–714.
13. Choi, Y.; Choi, M.; Kim, M.; Ha, J.W.; Choo, J. StarGAN: Unified Generative Adversarial Networks for Multi-domain Image-to-Image Translation. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018.
14. Tian, Y. zi2zi: Master Chinese Calligraphy with Conditional Adversarial Networks. Available online: https://github.com/kaonashi-tyc/zi2zi/ (accessed on 6 April 2017).
15. Zhou, P.; Zhao, Z.; Zhang, K.; Li, C.; Wang, C. An end-to-end model for chinese calligraphy generation. *Multimed. Tools Appl.* **2020**, *80*, 6737–6754. [CrossRef]
16. Li, G.; Zhang, J.; Chen, D. F2PNet: Font-to-painting translation by adversarial learning. *Inst. Eng. Technol.* **2020**, *14*, 3243–3253. [CrossRef]
17. Chen, J.F.; Chen, H.; Xing, X.U.; Yan-Li, J.I.; Chen, L.J. Learning to write multi-stylized chinese characters by generative adversarial networks. *J. Univ. Electron. Sci. Technol. China* **2019**, *13*, 2680–2686.
18. Mirza, M.; Osindero, S. Coconditional autoencoding adversarial networks for chinese font feature learning. *arXiv* **2018**, arXiv:1812.04451.

19. Jiang, Y.; Lian, Z.; Tang, Y.; Xiao, J. scfont: Structure-guided chinese font generation via deep stacked networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 4015–4022.
20. Lyu, P.; Bai, X.; Yao, C.; Zhu, Z.; Huang, T.; Liu, W. Auto-encoder guided gan for chinese calligraphy synthesis. In Proceedings of the 2017 14th IAPR International Conference on Document Analysis and Recognition, Kyoto, Japan, 9–15 November 2017; Volume 1, pp. 1095–1100.
21. Li, D.; Hu, J.; Wang, C.; Li, X.; She, Q.; Zhu, L.; Zhang, T.; Chen, Q. Involution: Inverting the inherence of convolution for visual recognition. *arXiv* **2021**, arXiv:2103.06255.
22. Badrinarayanan, V.; Handa, A.; Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *arXiv* **2015**, arXiv:1505.07293.
23. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
24. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. *arXiv* **2017**, arXiv:1706.03762.
25. Radford, A.; Metz, L.; Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. In Proceedings of the 4th International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
26. Agostinelli, F.; Hoffman, M.; Sadowski, P.; Baldi, P. Learning activation functions to improve deep neural networks. In Proceedings of the 3rd International Conference on Learning Representations, San Juan, Puerto Rico, 2–4 May 2016.
27. Dmitry, U.; Andrea, V.; Victor, L. Instance normalization: The missing ingredient for fast stylization. *arXiv* **2016**, arXiv:1607.08022.
28. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. In Proceedings of the 3rd International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2014.