# Multitask Adversarial Learning for Chinese Font Style Transfer

1st Lei Wu
*School of software*
*Shandong University*
Jinan, China
i_lily@sdu.edu.cn

2nd Xi Chen
*School of software*
*Shandong University*
Jinan, China
343232674@qq.com

3rd Lei Meng✉
*NExT++*
*National University of Singapore*
Singapore
lmeng@nus.edu.sg

4th Xiangxu Meng
*School of software*
*Shandong University*
Jinan, China
mxx@sdu.edu.cn

***Abstract*—Style transfer between Chinese fonts is challenging due to both the complexity of Chinese characters and the significant difference between fonts. Existing algorithms for this task typically learn a mapping between the reference and target fonts for each character. Subsequently, this mapping is used to generate the characters that do not exist in the target font. However, the characters available for training are unlikely to cover all fine-grained parts of the missing characters, leading to the overfitting problem. As a result, the generated characters of the target font may suffer problems of incomplete or even radicals and dirty dots. To address this problem, this paper presents a multi-task adversarial learning approach, termed MTfontGAN, to generate more vivid Chinese characters. MTfontGAN learns to transfer a reference font to multiple target ones simultaneously. An alignment is imposed on the encoders of different tasks to make them focus on the important parts of the characters in general style transfer. Such cross-task interactions at the feature level effectively improve the generalization capability of MTfontGAN. The performance of MTfontGAN is evaluated on three Chinese font datasets. Experimental results show that MTfontGAN outperforms the state-of-the-art algorithms in a single-task setting. More importantly, increasing the number of tasks leads to better performance in all of them.**

***Index Terms*—style transfer, font generation, multitask, GAN**

## I. Introduction

Chinese characters represent the culture and civilization of China. They play an important role in communication. The number of Chinese font has increased rapidly in the past 20 years, but the existing fonts still cannot meet the diversified and personalized needs of the digital age. People are more eager to build a personalized font library. Chinese characters are complicated and diverse. The Chinese official character set GB2312 consists of 6763 commonly used Chinese characters. The total number is larger than 80,000. Manual font creation is time-consuming and requires professional skills. Due to that, it is necessary to present a method to automatically generate Chinese font libraries. Thus, artists only need to design a small subset of the font to generate the whole Chinese font libraries, which save time and cost.

Existing methods of this task generally follow two directions: the computer graphics-based methods and deep learning based methods. Traditional computer graphics-based methods [1]–[4] usually build Chinese characters through assembling strokes. The results of the generated Chinese characters are always limited by the effect of stroke extraction. Some process of decomposing and adjusting strokes should be finished by the staff. Existing deep learning based methods [5]–[8] usually follow adversarial learning or neural style transfer to generate Chinese font, which generally implemented as a single-task process. Figure 1 shows an example of Chinese font style transfer. Some characters in HeiTi style are transferred to JinHei style while maintaining the consistency of the contents. These methods usually have weak generalization ability and some a little more complex strokes cannot be learned completely because of the complex and various structures of Chinese characters.

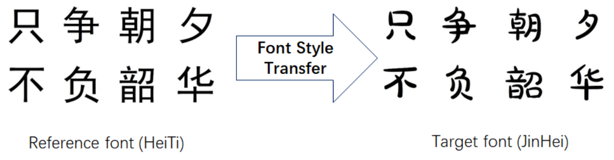

Fig. 1. Example of Chinese font style transfer. The eight Chinese characters in HeiTi style are transfered to JinHei style while maintaining the consistency of the contents. (HeiTi, JingHei are two types of Chinese fonts).

This paper aims to handle the challenging task of automatic generation of large-scale Chinese font libraries. We propose a multitask adversarial learning approach, termed MTfontGAN, for Chinese font style transfer. The model consists of one generator with multiple subnets, i.e. multi-task. The reference character is transferred into multiple high-level feature representations by using encoder parts. The multiple tasks' encoder layers share information with each other to achieve a more vivid character generation. MTfontGAN contains multiple discriminators for the multiple font styles generation. The discriminator is used to distinguish each style of the generated characters with their ground truth correspondingly. Thus, different font style results can be generated by the same MTfontGAN training model, as shown in Figure 2. Comparing with existing algorithms, MTfontGAN model has three unique characteristics, including 1) It completes the style transfer from a reference font to multiple target ones at the same time and achieves better performance in all tasks; 2) The multitask training strategy enables the feature-level

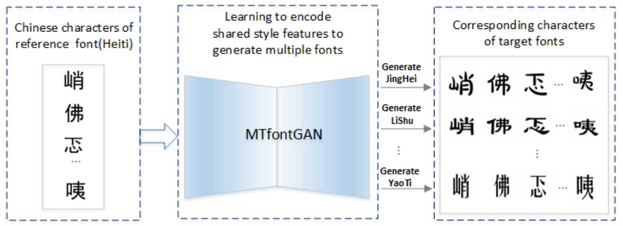
Chinese characters of reference font(Heiti)
峭
佛
忑
…
咦
Learning to encode shared style features to generate multiple fonts
MTfontGAN
Generate JingHei
Generate LiShu
Generate YaoTi
Corresponding characters of target fonts
峭 佛 忑 … 咦
峭 佛 忑 … 咦
峭 佛 忑 … 咦

Fig. 2. An overview of our method, i.e. MTfontGAN, for Chinese font transfer. It learns to attend to the important style characteristics of the reference font, i.e. HeiTi, by imposing a multitask framework. This makes the U-Net encoder encode the generalized style features required to generate different fonts.(HeiTi, JingHei, LiShu and YaoTi are the English names for the four types of Chinese fonts)

interactions across tasks, making the image encoders to focus on the important parts of characters in general style transfer; and 3) Existing algorithms typically require to be pretrained on a large external dataset of more than 20 fonts to enhance their generalization ability. In contrast, MTfontGAN does not require this procedure by taking advantage of the multitask learning paradigm. We conducted experiments on style transfer among four Chinese fonts to demonstrate the effective of our method.

In summary, this paper has two main contributions:

1) We propose a multitask adversarial learning approach that learns the mappings among multiple Chinese fonts using only a single generator and multiple discriminators, generating multiple Chinese font libraries simultaneously.

2) We propose an effective architecture to make MTfontGAN achieve stable training of multiple tasks in a unified framework. Experimental results show that MTfontGAN scalable to more than three tasks.

The rest of the paper is organized as follows. Section II briefly summarizes the related work. Section III describes the details of the multitask adversarial learning approach. Section IV describes these experiments on style transfer among four Chinese fonts. Finally, we draw conclusions in Section V.

## II. Related Work

Automatic generation of Chinese font library in a certain style is now still a challenging and ongoing problem. Some methods have been explored to generate font automatically. CG-based methods and deep learning based methods are main approaches to this task.

The traditional CG-based methods [1]–[4], [9] have to extract strokes from characters, then synthesize the strokes into target characters. Tomo et al. [10] proposed a method to extract the strokes from character images and construct characters by deploying the appropriate strokes onto skeletons that are generated automatically. Lian et al. [4] proposed a system to synthesize handwriting font libraries. Lin et al. [11] synthesizes Chinese characters with extracted components, by placing them properly according to their position and size information. They need to track every stroke, recognize and extract components from the user's handwritings first. Nevertheless, the results of the generated characters are limited by the effect of stroke extraction. Also, this kind of method only focusses on the local representation of Chinese characters rather than the overall style. Therefore, some attributes, like the size and location of the generated Chinese characters need to be adjusted manually.

In the last decade, deep neural networks [12] have been widely adopted in order to handle many challenging tasks in many areas. Deep learning based font generation methods are generally classified into two types: methods adapted from style transfer [13]–[16] and image-to-image transformation methods [6], [7], [17]. Atarsaikhan et al. [18] proposed an approach to generation fonts by using neural style transfer. Tian proposed 'rewrite'[1]method to transfer a given character from the standard font style to a target style. However, this kind of method usually has high computational complexity and the strokes of generated characters are incomplete.

Generative adversarial networks [19] are the most common methods. This kind of method usually optimizes a forward generative model through adversarial game between generator and discriminator and then transfer the reference font to the target one. The method 'pix2pix' [6] used a conditional GAN to predict image pixels from pixels. Jun-Yan Zhu et al. [17] proposed the 'BicycleGAN' to learn a multi-modal mapping between two image domains and the method 'CycleGAN' [7] solved the problem of using pairwise training dataset.

With the help of generative adversarial networks, some font generation methods based on them have been put forward. The paper [20], [21] proposed methods of English characters generation. Learning to generate English letters or numbers is relatively easy because there are just 26 letters in English, while generating Chinese fonts library is a tough job. The project 'zi2zi'[2] exploited a conditional GAN-based model which is an extension of 'pix2pix' to generate characters. Sometimes the strokes it generated are not learned completely. Jie Chang et al. [8] proposed a Hierarchical Generative Adversarial Network (HGAN) for typeface style transformation. Lyu et al. [22] proposed an auto-encoder guided GAN network (AEGN) which can synthesize calligraphy images with a specified style from standard Chinese font images. It requires 6000 paired characters as training sets, which is a weakness of this method. Bo Chang et al. [5] proposed an improved Cyclegan to transfer font styles. It replaces ResNet blocks with DenseNet blocks. Jiang Yue et al. proposed a method [23] to generate Chinese font, which first extracts strokes of characters. The SkelNet predict the skeleton stream of target characters and the StyleNet render their style.

Yue Jiang et al. [24] and Yexun Zhang et al. [25] used two separate networks to extract the content and style of characters separately. Sun et al. [26] proposed a SA-VAE framework, which incorporates the prior knowledge of Chinese characters into the network framework. However, ghosting and blurred strokes are common problems.

[1]Rewrite. https://github.com/kaonashi-tyc/Rewrite
[2]zi2zi. https://github.com/kaonashi-tyc/zi2zi

To solve the problems of generating characters mentioned above, such as incomplete strokes, images with noise, etc. And based on the inspiration of contributions in multi-task [27], we propose our MTfontGAN model that shares information among tasks, and accordingly generate more reasonable results.

## III. METHOD DESCRIPTION

As mentioned above, our goal is to address the problem of automatic generation Chinese font. To obtain better performance for this task, we specifically design the network architecture and loss functions. It can transfer the reference font into multiple styles of target fonts simultaneously. It contains multiple tasks, each task is a kind of style transformation. MTfontGAN includes one generator and multiple discriminators, and the quantity of discriminators depends on the number of generation fonts simultaneously. The generator is used to generate multiple characters with different style and discriminators are served to distinguish the generated characters from the ground truth. A soft alignment is imposed on the encoding of the reference font for different targets. This promotes the extraction of target-independent font features to improve MTfontGAN's generalization capability and reduces the risk of over-fitting. We will explicitly discuss the model in this section.

### *A. Problem Formulation*

We formulate the multitask font style transfer process as a mapping from a reference Chinese character $\mathbf{x}^{rs}$ to multiple target characters simultaneously. $\mathbf{x}^{rs} \to \{\mathbf{x}^{ts_1}, \mathbf{x}^{ts_2}, ..., \mathbf{x}^{ts_n}\}$, where $n$ is the number of the target fonts. Reference character images are binary glyph images in a standard font style containing little style information, for example HeiTi or DengXian font.

The paper presents MTfontGAN, which is achieved by the proposed continual multitask learning approach, learning to encode shared style features to generate multiple target fonts simultaneously. MTfontGAN adopts encoder $\mathcal{E}_p(.)$ to extract visual features of reference characters. It is a generative model that learns a mapping from reference character $\mathbf{x_i}^{rs}$ to output generated character $\hat{\mathbf{x}}_i^{ts_p}$, $\mathbf{x_i}^{rs} \to \hat{\mathbf{x}}_i^{ts_p}$,where $p \in \{1,2,...n\}$ and $i$ is the $i$-th input character. The multitask training strategy enables the feature-level interactions across tasks.

### *B. Base Model*

The conditional adversarial networks are a promising approach for image-to-image translation tasks. At first, a base model is used to generate one styled target font from the reference font. We select pix2pix [6] model as the base model due to its successful applications in image-to-image translation tasks. The generator G is trained to produce target characters that cannot be distinguished from "real" images by an adversarially trained discriminator, D, which is trained to do as well as possible at detecting the generator's "fakes".

### *C. MTfontGAN Network Achitectures*

Characters generated by the base model usually have noise and the images are blurry. Also, if we wish to generate another font, we need to retrain the network. Therefore, we propose our multi-task adversarial learning model (MTfontGAN) based on the base model, which can generate multiple types of fonts with higher quality at the same time. As shown in Figure 3, MTfontGAN model consists of one generator and $n$ discriminators, where $n$ is the number of fonts we want to generate. They are used to distinguish each style of the generated characters with their ground truth correspondingly.

*1) Multitask Generator:* The generator contains multiple subnets, namely, multiple tasks. Each task has the encoder part and decoder part. The "U-Net"-based architecture is used for each task generator [28]. The reference character is transferred into multiple high-level feature representations by using encoder parts, and decoder parts are used to reconstruct multiple target images progressively using the feature representations calculated by encoder layers.

**Encoder Network:** What's more, the corresponding encoder layers of these multiple tasks are connected with each other to share information, which is conducive to generating more vivid characters.

Encoder part of G consists of a series of Convolution-InstanceNorm-LeakyReLu blocks to encode the input image into high dimension feature. The input of encoder layers is the reference font. After passing through encoder layers, the reference character is transferred into high-level feature representation of the target style.

An alignment is imposed on the encoders of different tasks to make them focus on the important parts of the characters in general style transfer. We try to make the parameters included in corresponding layers of encoder parts as close as possible. As the generator needs to produce different styles of fonts, similar parameters increase the difficulty of the training network. Thus, the generator will have a stronger learning ability to generate more realistic images. To do this, we add losses between corresponding layers of encoder parts. So tasks can be connected to share information as well as have enough independent parameters to learn multiple styles of fonts at the same time. In this way, the generator has a stronger generative ability and the network is more robust. MTfontGAN has better generalization ability by learning multiple tasks simultaneously and it can reduce the risk of over-fitting.

**Decoder Network:** The decoder consists of a series of upsampling layers, which contain a 4×4 stride 2 deconvolution, Instance Normalization and Relu except for the last one. The last layer which only contains the deconvolution layer. The outputs are transformed into [0,1] by the sigmoid function. The decoder can reconstruct the target image with specific size and channels progressively using the feature representation calculated by encoder layers.

Given that each input character and its target one have the same contents but different styles, they should have similar structures. So we add skip connection between corresponding

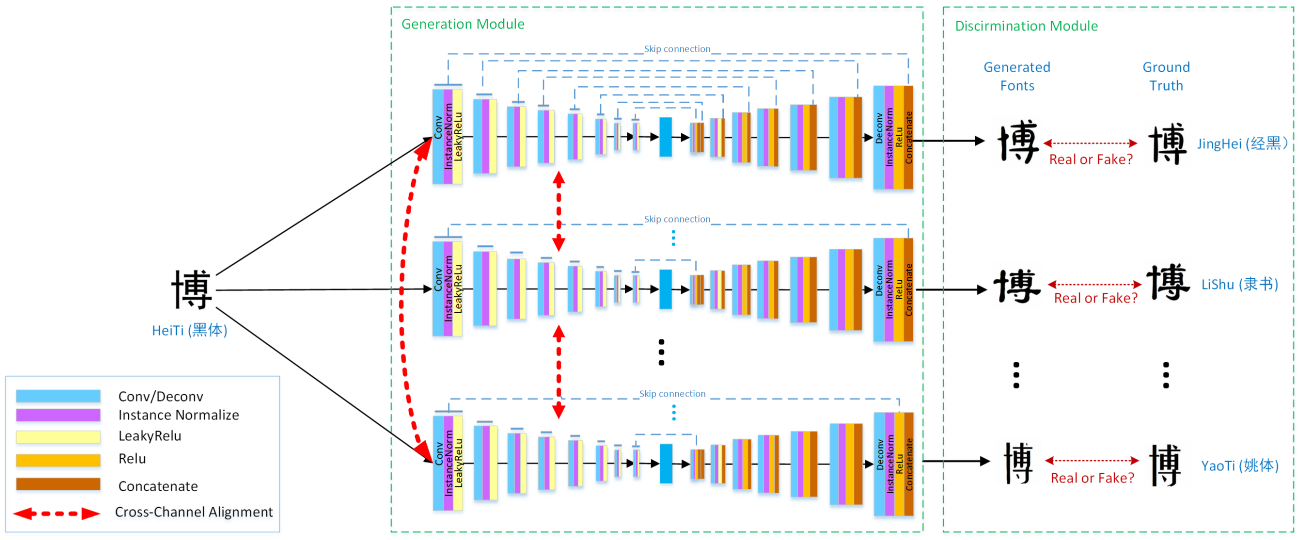

Fig. 3. Network architecture. MTfontGAN learns a mapping from one reference font to multiple styles of target fonts simultaneously. The generation module employs multiple U-Net for different font generation tasks. It uses a soft-sharing strategy for the image encoders in different channels to extract the common features of the reference font. The discrimination module adopts adversarial learning strategy to control the quality of the generated target fonts.

layers of the encoder and decoder to make full use of information on different levels.

*2) Discriminator:* The MTFontGAN consists of multiple independent discriminator modules. D= $\{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_n\}$ , where $n$ is the number of target fonts. They are used to distinguish each style of the generated characters with their ground truth correspondingly.

Each discriminator module composes of a series of Convolution-InstanceNorm-LeakyReLu blocks. And we use discriminator that concentrates its attention on local image patches other than the whole graphic. It is more accurate to judge the authenticity of the image using this structure. Meanwhile, it can also improve the generation ability of the generator.

## D. Objective Function

The objective function of our model consists of three terms: adversarial loss, L1 loss and soft alignment loss.

$$G^* = \arg\min_G \max_D \lambda_0 \mathcal{L}_{GAN}(G,D) + \\ \lambda_1 \mathcal{L}_{L_1}(G) + \lambda_2 \mathcal{L}_{L_{align}}(G) \tag{1}$$

where $\lambda_0$, $\lambda_1$, and $\lambda_2$ are the weights of the three losses correspondingly.

**Adversarial Loss:** we impose a standard adversarial game to train the generator G and discriminators D of every single task model.

$$\mathcal{L}_{GAN}(G,D) = \mathbb{E}_{\mathbf{x_i}^{rs},\mathbf{x_i}^{ts}}[logD(\mathbf{x_i}^{rs},\mathbf{x_i}^{ts})] \\ +\mathbb{E}_{\mathbf{x_i}^{rs}}[log(1-D(\mathbf{x_i}^{rs}, G(\mathbf{x_i}^{rs})))] \tag{2}$$

The adversarial loss of the multitask model is the sum of every single task.

$$\mathcal{L}_{GAN}(G,D) = \sum_{p=1}^{n} \mathcal{L}_{GAN_p}(G,D) \tag{3}$$

**L1 Loss:** to make the generated characters sharper and add more details, we use L1 loss in our objective function.

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{\mathbf{x_i}^{rs},\mathbf{x_i}^{ts}}[||\mathbf{x_i}^{ts} - G(\mathbf{x_i}^{rs})||_1] \tag{4}$$

where $\mathbf{x_i}^{rs}$ and $\mathbf{x_i}^{ts}$ represent the $i$-th reference character and ground truth correspondingly. The L1 loss of the multitask model is the sum of every single task.

$$\mathcal{L}_{GAN}(G,D) = \sum_{p=1}^{n} \mathcal{L}_{GAN_p}(G,D) \tag{5}$$

The objective of every single generation task contains two parts: adversarial loss and L1, which can be expressed as:

$$G^* = \arg\min_G \max_D \lambda_0 \mathcal{L}_{GAN}(G,D) + \lambda_1 \mathcal{L}_{L1}(G) \tag{6}$$

where $\lambda_0$ and $\lambda_1$ are the weights of the two losses correspondingly.

**Soft Alignment Loss:** A soft alignment is imposed on the encoding of the reference font for different targets. As L2 is concise and more sensitive to outliers, in the connection parts, we use L2 to achieve it. The L2 can be expressed by the following formula:

$$\mathcal{L}_{L2}(G) = \frac{2}{n(n-1)\cdot k} \sum_{k=1}^{K} \sum_{\substack{p,q=1,2,...,n \\ p\neq q}} ||T_p^{(k)} - T_q^{(k)}||_2 \tag{7}$$

where $n$ refers to the amount of tasks and $T_p$ and $T_q$ are the p-th and q-th tasks correspondingly. And $K$ represents the total layers in one encoder part.

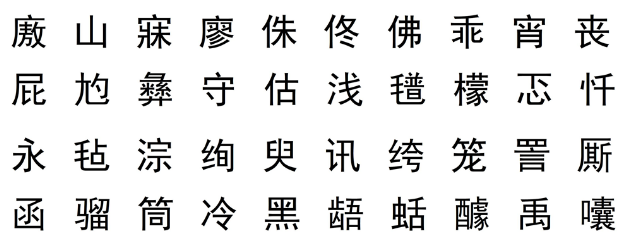
廒 山 寐 廖 侏 佟 佛 乖 宵 丧
屁 尥 彝 守 估 浅 氇 檬 忑 忏
永 毡 淙 绚 臾 讯 绔 笼 詈 厮
函 骝 筒 冷 黑 龉 蛞 醵 禹 囔

Fig. 4. Examples of content reference font images on the Chinese datasets.

The total objective of our network contains three parts: the adversarial loss loss,the L1 loss and the soft alignment loss, which can be expressed as:

$$G^* = \arg\min_G \max_D \lambda_0 \mathcal{L}_{GAN}(G,D) + \lambda_1 \mathcal{L}_{L1}(G) + \lambda_2 \mathcal{L}_{L2}(G) \tag{8}$$

where $\lambda_0$, $\lambda_1$, and $\lambda_2$ are the weights of the three losses correspondingly.

## *E. Training Process*

MTfontGAN model is optimized using three loss terms: adversarial loss, L1 loss and soft alignment loss. The specific training strategy has the following two steps:

1) Independently training base models first: Before training MTfontGAN, we first train the base model for each style of fonts that we want to generate simultaneously in MTfontGAN. The base model is optimized using two loss terms: $\mathcal{L}_{GAN}$ and $\mathcal{L}_{L1}$. The pre-trained parameters are used to initialize every task of MTfontGAN.

2) Training MTfontGAN model: After the base model has been trained well, we train the MTfontGAN model. Given the set of images $\{\mathbf{x_i}^{rs}, \mathbf{x_i}^{ts_1}, \mathbf{x_i}^{ts_2}, ....\mathbf{x_i}^{ts_n}\}$, where $i \in \{1,2,...m\}$ and $m$ is the number of training characters, $n$ is the number of tasks. We load the optimal parameter sets of each font firstly. Then we train the MTfontGAN model. We need to adjust the weight of $\mathcal{L}_{L2}$ loss.

After the MTfontGAN network is trained fully, we can use the rest of the reference characters to generate the whole target font libraries.

## IV. Experiments

In this section, we first introduce the datasets and model details. Then, we analyze sensitivity of MTfontGAN to hyperparameters. Finally, we compare the proposed method with baselines and state-of-the-art font style transfer algorithms to verify the effectiveness of our method.

## *A. Experimental Settings*

*1) Datasets:* To evaluate the proposed MTfontGAN model with Chinese font generation tasks, we construct a dataset which contains four Chinese fonts: HeiTi, JingHei, LiShu and YaoTi. Each font has 5000 Chinese characters selected randomly. The data split includes 4000 images for training, 800 for validation, and 200 for testing. To prepare the input datasets, we downloaded the source files of the four fonts from the network, transferred them into 256*256 images. Finally, the training sets got converted to .npy files for better performance of MTfontGAN.

We adopt HeiTi as the input content reference font. HeiTi is part of the commonly used fonts in printing. As shown in Figure 4, it has the advantages of concise strokes and rigorous structures. As the experimental MTfontGAN contains three tasks and it can generate three styles of fonts at the same time, we choose JingHei, LiShu and YaoTi fonts as our target fonts. For time efficiency, We randomly picked 400 characters subset in each style of fonts as our training sets. We give the experiment result on the whole 5000 dataset in the later section.

*2) Implementation Details:* The multiple tasks of MTfontGAN have the same network layers in every single model. The encoder has 8 stacked Convolution-InstanceNorm-LeakyReLu blocks. The output channels of each convolutional layer are 64, 128, 256, 512, 512, 512, 512 and 512 respectively. The decoder has 7 stacked Deconvolution-InstanceNorm-ReLu blocks and a final Deconvolution layer. Output channels of each deconvolutional layer are 512, 512, 512, 512, 256, 128 and 64 respectively. The size of convolution kernels is 4*4 and the stride is 2*2. The method was implemented in Pytorch. In our experiment, We set batch size to 16 and set $\lambda_0$ 10, $\lambda_1$ 1, $\lambda_2$ 25. We choose initial rate of 0.001 and train the proposed model end-to-end with Adam optimizer.

*3) Evaluation metrics:* For quantitative evaluation, we adopt four commonly-used metrics in many image generation tasks:

L1: the L1 of corresponding pixels between the generated image and ground truth;

RMSE (Root Mean Square Error): the RMSE loss of corresponding pixels between the generated image and ground truth;

PDAR (Pixel Disagreement Ratio): the ratio of the same pixels to the total number of pixels between the generated image and ground truth;

IOU (Intersection Over Union): the ratio of intersection of pixels to union of pixels between generated image and ground truth.

The formulas are described as follows (the $L1$ is mentioned in equation (2):

$$RMSE = \sqrt{\frac{1}{n}\sum_j (\hat{x}_{i,j}^{ts} - \ {x_{i,j}}^{ts})^2} \tag{9}$$

$$PDAR = \frac{\sum_j (\hat{x}_{i,j}^{ts} \wedge \ {x_{i,j}}^{ts})}{\sum_j \ {x_{i,j}}^{ts}} \tag{10}$$

$$IOU = \frac{\sum_j (\hat{x}_{i,j}^{ts} \wedge \ {x_{i,j}}^{ts})}{\sum_j (\hat{x}_{i,j}^{ts} \vee \ {x_{i,j}}^{ts})} \tag{11}$$

where $\hat{x}_{i,j}^{ts}$ is the $j$-th element of the generated character $\hat{\mathbf{x}}_i^{ts}$and ${x_{i,j}}^{ts}$ is the $j$-th element of the ground truth $\mathbf{x_i}^{ts}$correspondingly, $p \wedge q = min(p,q)$ is the fuzzy and operator, $p \vee q = max(p,q)$ is the fuzzy and operator.

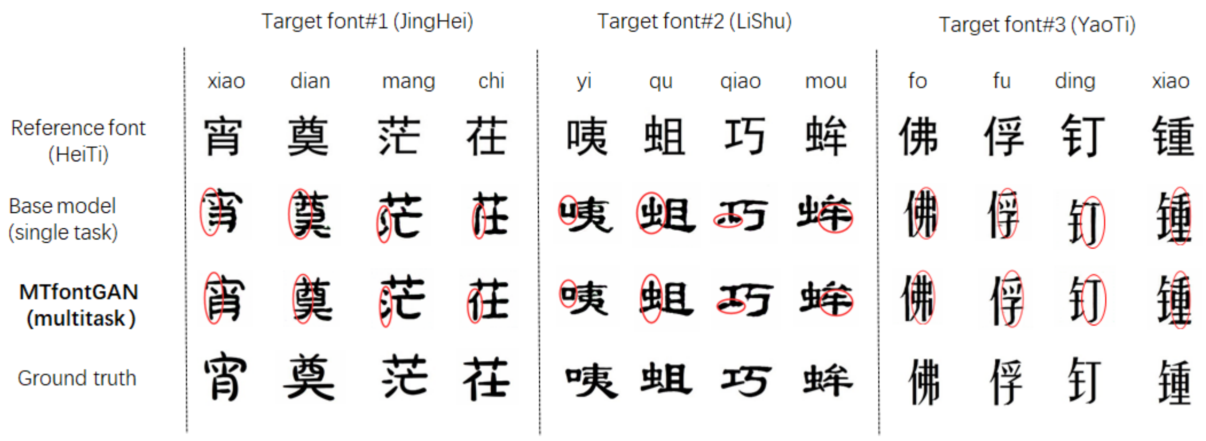

Fig. 5. Performance comparison in font transfer between base model performing single task font transfer and MTfontGAN performing multitask font transfer for the three target fonts simultaneously.

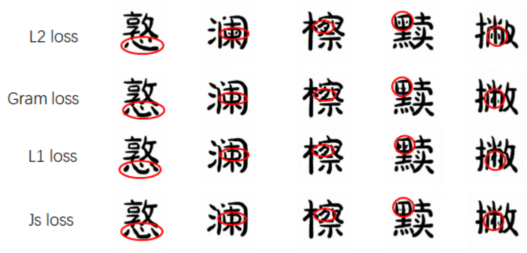

Fig. 6. Performance comparison between MTfontGAN using different alignment strategies for image encoders.

## B. *Effectiveness of Multi-Task style encoding*

As shown in Figure 5, we compare the results of our MTfontGAN with the base model. It can be seen that the results generated by base model do have serious strokes missing problems, such as the vertical stroke of the character ”Xiao” (the first character in the first row) and the left-falling stroke of the character ”chi” (the fourth character in the first row). And the strokes of generated characters are incomplete, such as the horizontal stroke of the character ’kou’ (the left part of the fifth character in the first row), the rising stroke of the character ’Qiao’ (the seventh character in the first row). Also, there are some unreasonable connection between strokes, for instance, the triple-dots of the generated character ’mang’ (the third character in the first row). And the character ’yi’ (the fifth character in the first row) has redundancy in the right-falling stroke. What’s more, the characters generated by the base model have noise, such as the character ’fo’ (the ninth character in the first row) and the character ’fu’ (the tenth character in the first row). While our model performs better in all these aspects mentioned above. It demonstrates the significance of multi-task style encoding. Our model can not only generate multiple styles of fonts without retraining but also generate images with clearer and more complete strokes that are consistent with the style of the target font.

## C. *Effects of different losses on network sharing*

To find a better way to connect between multiple tasks, we tried different methods, like L2, JS loss, Gram loss, and L1. As shown in Figure 6, we show the generated results using different losses. Although the styles of characters generated by these methods are consistent with the target font, all methods except L2 have the same problem. That is, strokes of generated characters are incomplete. Also, the method using L1 generates images with noise. This figure indicates that L2 is more appropriate.

## D. *Comparison with the State of the Art*

In this section, we compare the performance of our model with other existing methods. We compare the results of our method with other four recently proposed approaches: zi2zi, BicycleGAN [17], CycleGAN [7], DenseNet CycleGAN [5]. We train a model for every above mentioned methods as well as our method on the same dataset. For time efficiency, We randomly picked 400 characters subset in each style of fonts as our training sets.

1) *Visual comparison:* As shown in Figure 7, the characters generated by BicycleGAN, CycleGAN and DenseNet CycleGAN have a serious problem of missing strokes. And strokes produced by zi2zi method are incomplete. The styles of characters generated by CycleGAN and DenseNet CycleGAN are quite different from the target font. From comparisons among these methods, we can see that our MTfontGAN model is a better one for generating stylized fonts.

2) *Qantitative evaluation:* Although the visual appearance is much more intuitive to reflect the quality of style transfer results in the font generation task, quantitative evaluation metrics can give a higher-level indication of performance on the whole dataset. Table 1 shows the quantitative comparison of our method and other four approaches. Our method can achieve the lowest L1, RMSE, PDAR and the highest IOU. These accurate values demonstrate that the MTfontGAN model is superior to other methods.

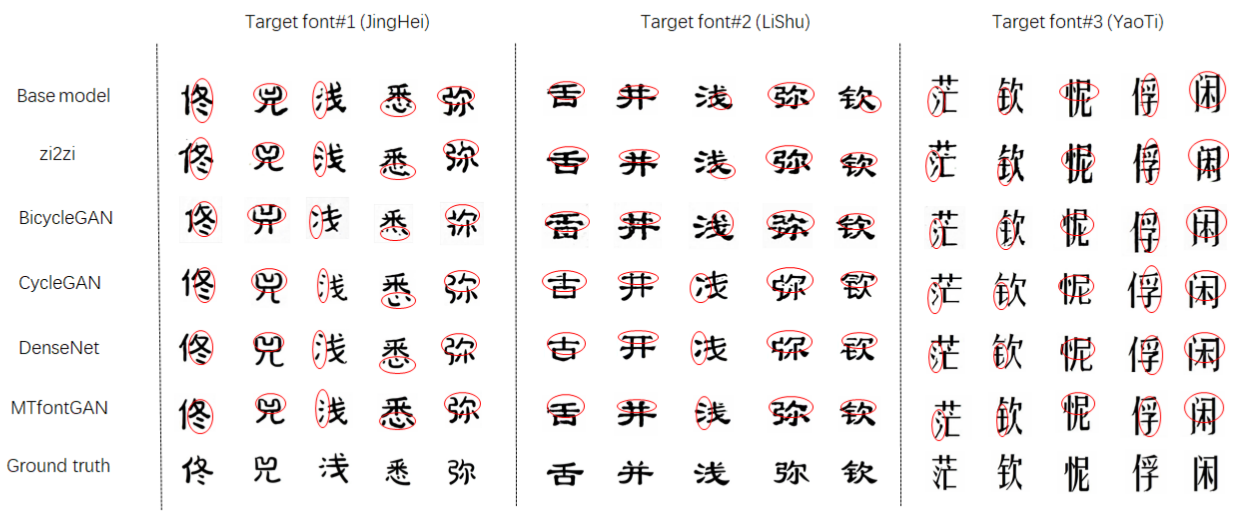

Fig. 7. Performance comparison between MTfontGAN in and existing methods in terms of font transfer from HeiTi to JingHei, LiShu, and YaoTi.

TABLE I
QUANTITATIVE COMPARISON OF MTFONTGAN AND OTHER FIVE EXISTING METHODS. (DENSENET REPRESENTS DENSENET CYCLEGAN.)

| Method | JingHei | | | | LiShu | | | | YaoTi | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | L1 | RMSE | PDAR | IOU | L1 | RMSE | PDAR | IOU | L1 | RMSE | PDAR | IOU |
| Base model | 0.2526 | 0.6688 | 0.3901 | 0.4312 | 0.1502 | 0.4949 | 0.3296 | 0.5045 | 0.1493 | 0.4776 | 0.3967 | 0.4208 |
| zi2zi | 0.2233 | 0.6202 | 0.3867 | 0.4406 | 0.1555 | 0.4906 | 0.2688 | 0.5773 | 0.1715 | 0.5086 | 0.3963 | 0.4348 |
| BicycleGAN | 0.3246 | 0.7233 | 0.8088 | 0.1101 | 0.2673 | 0.5147 | 0.8929 | 0.0597 | 0.1698 | 0.4758 | 0.8207 | 0.1036 |
| CycleGAN | 0.4694 | 0.9004 | 0.9435 | 0.0292 | 0.3786 | 0.8393 | 0.3650 | 0.4655 | 0.4728 | 0.9106 | 0.6649 | 0.2023 |
| DensenNet | 0.5189 | 0.9546 | 0.6992 | 0.1788 | 0.3559 | 0.8139 | 0.3374 | 0.4959 | 0.4734 | 0.9027 | 0.8427 | 0.0854 |
| **MTfontGAN** | **0.2219** | **0.6176** | **0.3839** | **0.4458** | **0.1485** | **0.4877** | **0.2602** | **0.5876** | **0.1463** | **0.4741** | **0.3852** | **0.4450** |

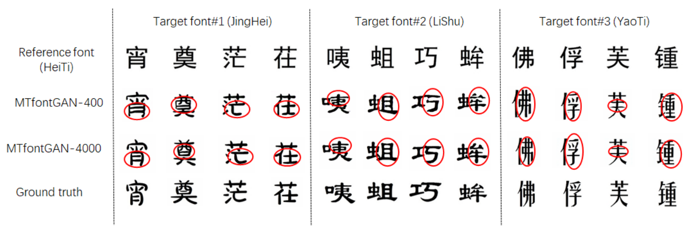

Fig. 8. The influence of the size of datasets. We compare the MTfonGAN models that trained by 400 and 4000 data correspondingly.

### *E. Influence of training data size to font generation*

In this part, we evaluate the effect of the size of datasets on the font generation. We perform an additional experiment with datasets that contain 4000 characters in each style of fonts. Except for the size of the datasets, all the experimental conditions are the same as those mentioned at the beginning of section 4. As shown in Figure 8, it can be seen that with the increase of datasets, the effect of generated characters has been significantly improved. The characters have more complete strokes and less noise and the style is more consistent with the target fonts. Therefore, the larger the datasets, the better the results.

## V. CONCLUSION

In this paper, we propose a multi-task generative adversarial network, termed MTfontGAN, to learn the generalized font patterns for Chinese font style transfer. A network sharing strategy is designed to maximize the propagation of positive feature-level information across tasks and retain the task-specific features into their own network channels. This enables the stable training of MTfontGAN. Our method can produce multiple types of fonts at one time without retraining. Experimental results demonstrated that our method can be used to automatically generate high-quality Chinese font libraries. Characters generated using our model have more complete structures and are closer to the ground truth.

Despite the achievements of MTfontGAN, future work can be further explored in two directions. First, we will do more experiments using various fonts to verify MTfontGAN's robustness on more Chinese fonts. Second, we will investigate its scalability on four or more tasks.

## ACKNOWLEDGMENT

Our work is supported by the National Key R&D Program of China under grant No.2017YFB0203000; and in part, by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@SG Funding Initiative.

## References

[1] S. Xu, T. Jin, H. Jiang, and F. C. M. Lau, "Automatic generation of personal chinese handwriting by capturing the characteristics of personal handwriting," in *Proceedings of the Twenty-First Innovative Applications of Artificial Intelligence Conference*, 2009, pp. 191–196.

[2] B. Zhou, Weihong Wang, and Zhanghui Chen, "Easy generation of personal chinese handwritten fonts," in *IEEE International Conference on Multimedia and Expo*, 2011, pp. 1–6.

[3] A. Zong and Y. Zhu, "Strokebank: automating personalized chinese handwriting generation," in *Proceedings of the National Conference On Artificial Intelligence*, 2014, pp. 3024–3029.

[4] Z. Lian, B. Zhao, and J. Xiao, "Automatic generation of large-scale handwriting fonts via style learning," in *SIGGRAPH Asia*, 2016, pp. 1–4.

[5] B. Chang, Q. Zhang, S. Pan, and L. Meng, "Generating handwritten chinese characters using cyclegan," in *IEEE Winter Conference on Applications of Computer Vision*, 2018, pp. 199–207.

[6] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5967–5976.

[7] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2242–2251.

[8] J. Chang, Y. Gu, Y. Zhang, and Y. Wang, "Chinese handwriting imitation with hierarchical generative adversarial network," in *the British Machine Vision Conference*, 2018, pp. 1–12.

[9] T. Miyazaki, T. Tsuchiya, Y. Sugaya, S. Omachi, M. Iwamura, S. Uchida, and K. Kise, "Automatic generation of typographic font from small font subset," *IEEE Computer Graphics and Applications*, vol. 40, no. 1, pp. 99–111, Jan 2020.

[10] M. Tomo, T. Tatsunori, S. Yoshihiro, O. Shinichiro, I. Masakazu, U. Seiichi, and K. Koichi, "Automatic generation of typographic font from small font subset," *IEEE Computer Graphics and Applications*, vol. 40, no. 1, pp. 99–111, Jan 2020.

[11] J. W. Lin, C. Y. Hong, R. Chang, Y. C. Wang, S. Y. Lin, and J. M. Ho, "Complete font generation of chinese characters in personal handwriting style," in *IEEE 34th International Performance Computing and Communications Conference (IPCCC)*, 2015, pp. 1–9.

[12] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[13] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2414–2423.

[14] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *IEEE International Conference on Computer Vision*, 2017, pp. 1510–1519.

[15] T. Q. Chen and M. W. Schmidt, "Fast patch-based style transfer of arbitrary style," in *the 30th Conference on Neural Information Processing Systems*, 2016, pp. 1–5.

[16] D. Chen, L.Yuan, J. Liao, N.Yu, and G. Hua, "Stylebank: An explicit representation for neural image style transfer," in *In Pro- ceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1–10.

[17] J. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman, "Toward multimodal image-to-image translation," in *Conference on Neural Information Processing Systems (NIPS)*, 2017, pp. 465–476.

[18] G. Atarsaikhan, B. K. Iwana, A. Narusawa, K. Yanai, and S. Uchida, "Neural font style transfer," in *14th IAPR International Conference on Document Analysis and Recognition*, 2017, pp. 51–56.

[19] I. J. Goodfellow, J. Pougetabadie, M. Mirza, B. Xu, D. Wardefarley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *Conference on Neural Information Processing Systems*, 2014, pp. 2672–2680.

[20] S. Azadi, M. Fisher, V. Kim, ZhaowenWang, E. Shechtman, and T. Darrell, "Multi-content gan for few-shot font style transfer," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*, 2018, pp. 7564–7573.

[21] S. U. Hideaki Hayashi, Kohtaro Abe, "Glyphgan: Style-consistent font generation based on generative adversarial networks," *Knowledge-Based Systems*, vol. 186, 2019.

[22] P. Lyu, X. Bai, C. Yao, Z. Zhu, T. Huang, and W. Liu, "Auto-encoder guided gan for chinese calligraphy synthesis," in *the 14th IAPR International Conference on Document Analysis and Recognition*, 2017, pp. 1095–1100.

[23] J. Yue, L. Zhouhui, T. Yingmin, and X. Jianguo, "Scfont: Structure-guided chinese font generation via deep stacked networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, pp. 4015–4022.

[24] Y. Jiang, Z. Lian, Y. Tang, and J. Xiao, "Dcfont: an end-to-end deep chinese font generation system," in *SIGGRAPH Asia*, 2017, pp. 1–4.

[25] Y. Zhang, Y. Zhang, and W. Cai, "Separating style and content for generalized style transfer," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8447–8455.

[26] D. Sun, T. Ren, C. Li, H. Su, and J. Zhu, "Learning to write stylized chinese characters by reading a handful of examples," in *the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 920–927.

[27] S. Ruder, "An overview of multi-task learning in deep neural networks," *CoRR*, vol. abs/1706.05098, 2017. [Online]. Available: http://arxiv.org/abs/1706.05098

[28] P. F. O. Ronneberger and T. Brox, "U-net: Convolu- tional networks for biomedical image segmentation," in *MIC-CAI*, 2015, p. 234–241.