JointFontGAN: Joint Geometry-Content GAN for Font Generation via Few-Shot Learning

Yankun Xi
Wayne State University
yankun.xi@wayne.edu

Guoli Yan
Wayne State University
guoliyan@wayne.edu

Jing Hua
Wayne State University
jinghua@wayne.edu

Zichun Zhong*
Wayne State University
zichunzhong@wayne.edu

ABSTRACT

Automatic generation of font and text design in the wild is a challenging task since font and text in real world exhibit various visual effects. In this paper, we propose a novel model, *JointFontGAN*, to derive fonts, including both geometric structures and shape contents in correctness and consistency with very few font samples available. Specifically, we design an end-to-end deep learning based approach for font generation through the new multi-stream extended conditional generative adversarial network (XcGAN) models, which jointly learn and generate both font skeleton and glyph representations simultaneously. It can adapt to the geometric variability and content scalability at the neural network level. Then, we apply it, along with the developed efficient and effective one-stage model, to text generations in letters and sentences / paragraphs with both standard and artistic / handwriting styles. The extensive experiments and comparisons demonstrate that our approach outperforms the state-of-the-art methods on the collected datasets including 20K fonts (letters and punctuations) with different styles.

CCS CONCEPTS

• **Computing methodologies** → **Shape modeling**; **Computer vision**; • **Information systems** → **Multimedia content creation**.

KEYWORDS

Font generation; conditional GAN; skeleton; few-shot learning

ACM Reference Format:
Yankun Xi, Guoli Yan, Jing Hua, and Zichun Zhong. 2020. JointFontGAN: Joint Geometry-Content GAN for Font Generation via Few-Shot Learning. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20), October 12–16, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394171.3413705

*Corresponding author.

MM '20, October 12–16, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-7988-5/20/10…$15.00
https://doi.org/10.1145/3394171.3413705

1 INTRODUCTION

Texts in multimedia, e.g., newspaper, advertisement, webpage, image, video, etc., contain rich geometric and semantic information that is very useful in many applications. However, traditional calligraphic or artistic font design is a time-consuming and labor-intensive process due to the complex combination of structures and content details. Moreover, maintaining a coherent style among all characters of a font is also difficult, especially in a large-scale font design. In particular, for the design of artistic fonts, automatically generating glyph images for characters / words could be helpful. During the past few years, the researchers have applied deep learning methods to synthesize and analyze glyph images inspired by the successes of these techniques in general 2D images and 1D texts. The ability to understand and analyze this kind of 2D art design is becoming increasingly important in multimedia, computer graphics, and computer vision.

In recent decades, there are many traditional model-driven font generation methods proposed, such as font manifold in a high dimensional space [5], stroke-based geometric representation [2], statistics-based method [29], constraint-based reasoning system [28], hierarchy of character components [27], Chinese Character Radical Composition Model [31], StrokeBank [33], EasyFont [15], etc. However, these approaches suffer from tons of low-level overwhelming features which need to be handcrafted, and complicated manual parameter adjustment which is required to derive style variations.

Recently, data-driven approaches have been proposed to robustly investigate the correlations between different objects / instances without relying on hard-coded metrics. Along this direction, there is an emerging trend to automatically generate different styles of font / glyph by deep neural network (DNN), such as DNN-based method [3], modified variational autoencoder (VAE) method [26], image-to-image translation model [18], zi2zi [25], DCFont [12], Hierarchical Adversarial Network (HAN) [7], CycleGAN-based model [6], SCFont [13], PEGAN [22], MC-GAN [1], AGIS-Net [8], FontRNN [24], convolutional recurrent generative model [4], GlyphGAN [10], FontGAN [16], etc. However, most of the above methods are not able to automatically generate large-scale fonts with high-quality and high-consistency in geometry and content / style for characters and words / paragraphs from a few given samples.

In this work, we present a new method, *JointFontGAN*, to transfer both geometric structures and shape styles in correctness and consistency with very few font samples. This new multi-stream generative adversarial network (GAN) model can better capture and synthesize local font geometry and global style consistency. It

can adapt to the geometric variability and scalability at the neural network level. Furthermore, we apply it, along with the developed efficient and effective one-stage model, to font and sentence generations with both standard and artistic / handwriting styles. The key *contributions* of our work are as follows:

- It proposes an end-to-end deep learning based approach for font generation by synergically utilizing multi-stream GAN models to jointly learn and generate both font skeleton and glyph representations simultaneously.
- The extended conditional GAN (XcGAN) model in each stream is developed to effectively learn the font structure and content consistency from very few samples.
- Our proposed network models present new state-of-the-art performance using the collected large-scale datasets, including one 10K gray-scale Latin fonts each with 26 uppercase letters and the other 10K gray-scale Latin fonts each with 26 uppercase and lowercase letters, and 8 punctuations.

2 RELATED WORK

In this section, we focus only on recently related font generations, which are categorized into model-driven and data-driven methods.

2.1 Model-Driven Font Generation

Traditional calligraphic or artistic font design is a time-consuming and labor-intensive process. Therefore, it is important to develop automatic font generation methods. In recent decades, there are many model-driven font generation approaches proposed. For instance, Campbell and Kautz [5] built a font manifold to generate new fonts by interpolation in a high dimensional space. Balashova et al. [2] developed a stroke-based geometric model for glyph synthesis, embedding fonts on a manifold through purely geometric features, by using a fitting procedure to re-parametrize arbitrary fonts. Yang et al. [29] proposed a novel statistics-based method to solve the text effects transfer problem by exploiting the analytics on the high regularity of the spatial distribution for text effects to guide the synthesis process. There are also many works specifically designed for Chinese characters. Xu et al. [28] proposed the shape grammar to represent each character in a multi-level manner and generate calligraphy via a constraint-based reasoning system. Xu et al. [27] then proposed an intelligent algorithm which can generate Chinese handwriting by a person requiring only a very small set of input characters. Zhou et al. [31] proposed an efficient solution to generate Chinese handwritten fonts by using a Chinese Character Radical Composition Model. StrokeBank [33] is an approach to automating personalized Chinese handwriting generation by using a semi-supervised algorithm to construct a dictionary of component mappings from a small seeding set. Lian et al. [15] proposed a system, i.e., EasyFont, to automatically synthesize large-scale Chinese handwriting fonts by learning styles from a small number of selected written samples. However, these model-driven based approaches are easily overwhelmed by tons of low-level handcrafted (e.g., graphic, geometric, statistic) features and different font style variations, e.g., Chinese, English, etc.

2.2 Data-Driven Font Generation

With the increasing development of imaging and data technology, data-driven approaches become popularly applicable and useful to image processing and analysis. Recently, there is an emerging trend to automatically synthesize different styles of font / glyph by DNN in computer vision, multimedia, and computer graphics. Baluja [3] proposed one of the earliest works to use DNN to generate English glyph images (style of a font). Upchurch et al. [26] considered glyph image synthesis as an image analogy task and proposed a modified variational autoencoder (VAE) to separate image style from content. Lyu et al. [18] proposed to apply an image-to-image translation model to learn mappings from glyph images in the standard font style to those with desired styles to synthesize Chinese calligraphy. zi2zi [25] is the first attempt to generate fonts using GAN, which is directly derived and extended from the pix2pix model [11] to adopt a style transfer method using condition GAN to achieve the goal of Chinese font generation. Then, there are many GAN-based methods (variants) for font generation. For instance, DCFont [12] is an end-to-end learning system which can automatically generate the whole Chinese font library from a small number of written characters. Hierarchical Adversarial Network (HAN) [7] is proposed for the typeface transformation by a transfer network and a hierarchical adversarial discriminator. Chang et al. [6] formulated the Chinese handwritten character generation as a mapping from an existing printed font to a personalized handwritten style using CycleGAN [32]. In order to learn and generate the high-fidelity and detailed contents from the fonts, recently, Jiang et al. [13] designed a deep stacked neural network, i.e., SCFont, with the guidance of glyph structure information to synthesize high-quality Chinese fonts using multi-stage architecture design. PEGAN [22] consists of one generator and one discriminator, where the generator is built using one encoder-decoder structure with cascaded refinement connections and mirror skip connections. MC-GAN [1] presents a stacked conditional GAN (cGAN) architecture to predict the coarse glyph shapes, and an ornamentation network to predict color and texture of the final glyphs. Unlike two-stage MC-GAN framework, AGIS-Net [8] transfers both shape and texture styles in one-stage with only a few stylized samples in order to improve the computational efficiency. FontRNN [24] treats Chinese characters as sequences of points (writing trajectories) and proposes to handle the font generation task via a Recurrent Neural Network (RNN) model. Bhunia et al. [4] proposed a convolutional recurrent generative model to solve the word level font transfer problem and maintain the consistency of the final font images. GlyphGAN [10] proposes a style-consistent font generation method based on GANs with the input vector for both considering the character class and style. FontGAN [16] proposes a unified GAN framework for modeling Chinese character stylization and de-stylization together. However, most of the above methods are not designed to automatically generate large-scale fonts with high-correctness and high-consistency in geometry and content (style) for characters and sentences from a few given samples.

3 JOINT GEOMETRY-CONTENT GAN

In this paper, we propose an end-to-end deep neural network to predict a full group of style-and-content-harmonious images with specific classes, i.e., a large variety of English fonts in the wild, when taking a small / limited subset of observations as input. Inspired by [1, 11], we propose the extended conditional GAN (XcGAN) to compose the full architecture, including two-stream generative

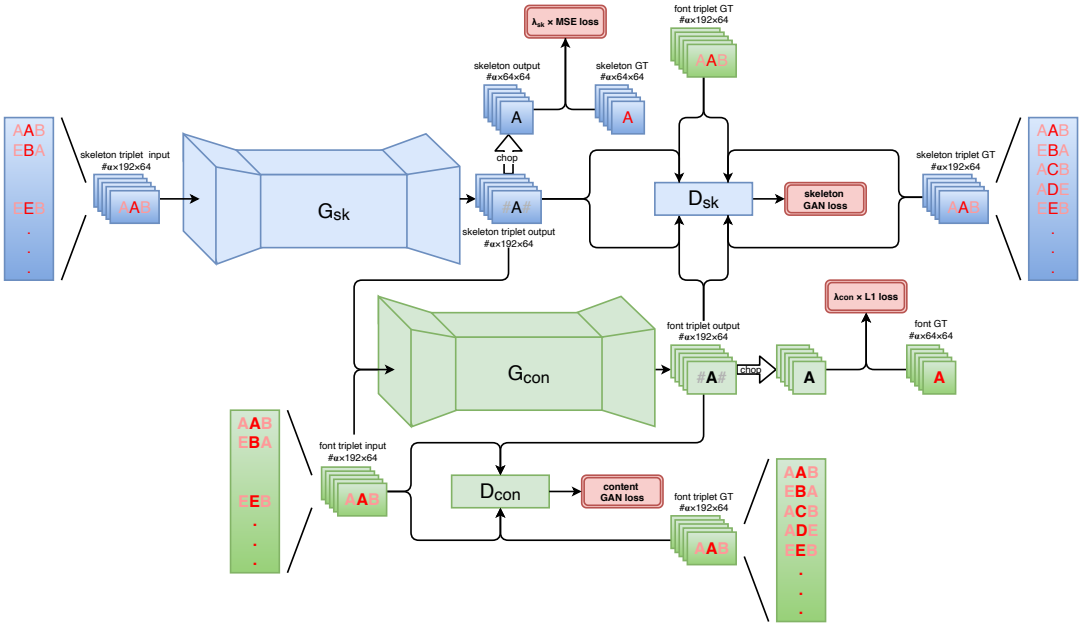

Figure 1: The overall architecture of JointFontGAN model, including two-stream GAN subnetworks, i.e., Sk-XcGAN and Con-XcGAN, composed of two generators G_{sk}, G_{con}, two discriminators D_{sk}, D_{con}, and some additional L_1 and MSE losses. Detailed explanations of each component are demonstrated among subsections in Sec. 3.

subnetworks. The first stream subnetwork, Sk-XcGAN, predicts the general skeletons of the glyphs. The second stream subnetwork, Con-XcGAN, predicts the detailed content of the glyphs. Both of the subnetworks work together as two streams within an one-stage process. In the following subsections, we introduce the components of the JointFontGAN model: XcGAN model, two-stream XcGAN architectures, i.e., Sk-XcGAN and Con-XcGAN, and loss functions. The overall architecture is demonstrated in Fig. 1.

3.1 Extended Conditional Generative Adversarial Network (XcGAN)

Generally, a generative adversarial network (GAN) [9] trains a model to generate data y (e.g., images) with a random noise vector z as the input. It comes with a corresponding discriminator $z \rightarrow y$ following a distribution by adversarially training a generator. While the discriminator tends to differentiate between real and fake images, the generator opposes the discriminator by creating realistic-looking fake images. The conditional GAN (cGAN) has an improved scenario [11, 20], where the generator is fed with an observed image x along with the random noise vector, making the mapping as $\{x, z\} \rightarrow y$. In order to maintain consistency among glyphs, we need to pay more attention to their relationship when placed them together. To achieve this, we apply a simple but effective strategy, extending our input from a single letter to a triplet with three letters (e.g., a letter and its two neighbors) in a cGAN model, i.e., an extended observed image triplet $\tilde{x}$. In this way, the generator can extract and learn features not only from individual letters, but also from inter-letter consistency. Now we have the new mapping with extended input and output, as $\{\tilde{x}, \tilde{z}\} \rightarrow \tilde{y}$, which we consider as the *extended conditional GAN (XcGAN)* and the corresponding loss function is formulated as:

$$\begin{aligned}\mathcal{L}_{XcGAN}(G, D) = \mathbb{E}_{\tilde{x}, \tilde{y} \sim p_{data}}[logD(\tilde{x}, \tilde{y})] \\ + \mathbb{E}_{\tilde{x} \sim p_{data}, \tilde{z} \sim p_{\tilde{z}}}[1 - logD(\tilde{x}, G(\tilde{x}, \tilde{z}))],\end{aligned} \tag{1}$$

where G is to minimize this loss function, while D is to maximize it, respectively.

During training, when ground truth of the prediction is available, it is possible to push the model to mimic real images against the discriminator through an additional loss, such as L_1 loss as suggested in [11]. It is noted that the L_1 loss is only applied on the generated (single) letter y without considering the generated triplet $\tilde{y}$. So the objective of the generator is:

$$G^* = \arg\min_G \max_D \mathcal{L}_{XcGAN}(G, D) + \lambda \mathcal{L}_{L_1}(G), \tag{2}$$

where

$$\mathcal{L}_{L_1}(G) = \mathbb{E}_{\tilde{x}, \tilde{y} \sim p_{data}, \tilde{z} \sim p_{\tilde{z}}}[||y - G^{chop}(\tilde{x}, \tilde{z})||_1], \tag{3}$$

and $G^{chop}(\tilde{x}, \tilde{z})$ is the chopped result after triplet generation. This XcGAN setting is applied in both Sk-XcGAN and Con-XcGAN subnetworks to generate the whole set of letter skeletons and glyphs with a consistent structure and content y, by only a few observing examples fed in a stack $\tilde{x}$. The random noise is ignored as the input to the generator, and dropout is the only source of randomness in the network. The general architecture of XcGAN is shown in Fig. 2. To evaluate the importance of XcGAN model in our framework, we demonstrate the comparison between cGAN and it in Sec. 4.

3.2 Few-Shot Learning

To generate a full set of letters (and punctuations) of a specific font from a limited number of example glyphs, the correlations and

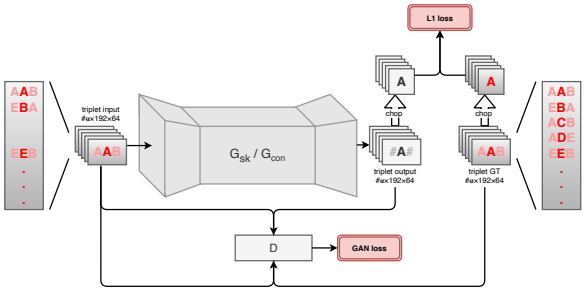

Figure 2: The architecture of XcGAN model.

similarities among source letters and the unseen ones should be captured. The basic XcGAN structure is expected to predict the unseen letters based on the visible source letters, such as the font skeleton and content used in this work.

In order to achieve the goal of the style similarity among all font images, we add one input channel for each individual skeleton / glyph in our proposed XcGAN resulting in a 3D volumetric skeleton / glyph stack in both input and the generated output. Note that, a basic tiling of all skeletons / glyphs into a large single 2D image is not an ideal choice, since it may fail to capture correlations among them particularly for those far from each other along the image length. This occurs due to the smaller size of convolution receptive fields than the image length within a reasonable number of convolutional layers. Moreover, each font image includes a triplet with three letters, which can enhance the sparse and consistent information of the input, especially for these unseen letters.

With the help of the novel input *3D skeleton / glyph triplet stack* design, correlation between different skeletons / glyphs are learned across network channels in order to transfer their style automatically. With the proposed input data structure, 3D convolutional layers can be applied to capture features shared among the whole alphabet. We employ our generator G_{sk} and G_{con} based on the image transformation network [14], including convolutional feature extractor, deconvolutional image reconstructor, and six ResNet blocks. The full architectural specifications of both G_{sk} and G_{con} are discussed in the following and shown in Fig. 3 and Fig. 4.

3.3 Sk-XcGAN Subnetwork

To our knowledge, a font design can be mainly defined based on geometric structure and shape content. Different from most of other content designs, fonts are originally written by human and have a nature of continuous writing routes, which can be comprehended and represented as skeletons. Skeletons only reveal (geometric) structural similarity of style without the content / texture from the glyphs. The Sk-XcGAN subnetwork is designed to learn skeletons of the unseen letters, which are simultaneously generated along with the final glyphs in the Con-XcGAN subnetwork. The ground truth of skeletons is generated by a classical fast parallel approach [30].

In the Sk-XcGAN subnetwork, skeleton generator G_{sk} (as shown in Fig. 3) is modified from our basic XcGAN model. Due to the sparseness of the skeleton information in the whole image, mean squared error (MSE) loss function is also applied as shown in Fig. 1. In the generator module, we assume the size of font skeleton images is 64×64. Based on our 3D skeleton triplet stack design, the original skeleton image is extended by adding random known skeletons to its left and right, so the dimension is $\#batch \times \#alphabet \times 192 \times 64$, which is a tensor representing gray-scale skeleton images after being stacked. Comparing with Con-XcGAN, we add two more components of Conv+BN+Relu layers (next to both input and output) in skeleton generator as shown in Fig. 3. The results indicate that these additional layers can learn more effective and discriminative features from skeletons. During the training stage, we add two random visible letter skeletons to the input and ground truth to compose the skeleton triplets, and the generator generates a stacked skeleton triplet for the output. We apply the GAN loss with the skeleton triplets, while we use the MSE loss only on the generated center letter skeletons with a chopping operation as shown in Fig. 1.

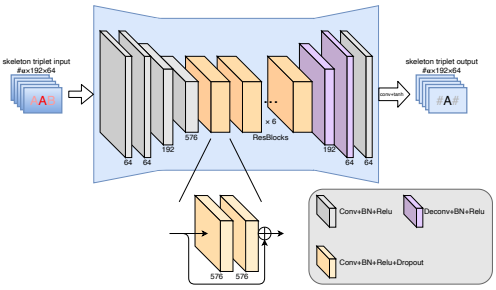

Figure 3: Skeleton generator G_{sk} in our Sk-XcGAN model.

In the discriminator module D_{sk}, we choose a two-level design as suggested by PatchGAN model [11], which provides both local and global discriminations between real and fake skeletons. The local part has three convolutional layers over the generated output skeleton triplet stack in order to discriminate between real and fake local patches resulting in a receptive field. In parallel, two extra convolutional layers are added as a global discriminator, resulting in a receptive field covering the whole skeleton image to distinguish between realistic skeleton images and generated ones. The local and global parts both contribute to the GAN loss. It is noted that D_{sk} learns to distinguish between fake and real skeletons based on not only skeleton styles and distinctions, but also those from font (glyph) content as shown in Fig. 1, which can further enhance the discriminative capability for Sk-XcGAN with some reference / intervention from Con-XcGAN.

3.4 Con-XcGAN Subnetwork

Our another subnetwork, Con-XcGAN, is to transfer the detailed contents of the few observed letters to the gray-scale glyphs through another XcGAN model consisting of a generator G_{con} and a discriminator D_{con}. Feeding the generated 3D skeleton triplets and the input 3D glyph triplet stack as input, the Con-XcGAN subnetwork generates the final outputs, which have been enriched with desirable font contents / textures upon the geometric structures. The main differences of our proposed Con-XcGAN subnetwork compared to the above Sk-XcGAN subnetwork are: (1) In the generator, the input is to concatenate 3D skeleton and glyph triplets together along the feature channels, so the size of the input data fed is $\#batch \times \#alphabet \times 2 \times 192 \times 64$. The generator configuration as shown in Fig. 4 is slightly modified from Sk-XcGAN. (2)

In the discriminator, it also uses a two-level design, which is the same as Sk-XcGAN architecture, including both local and global discrimination between real and fake glyph contents. But it learns to distinguish between fake and real glyphs based on font styles and distinctions as shown in Fig. 1. Other components in the generator and the discriminator architecture are quite similar to Sk-XcGAN.

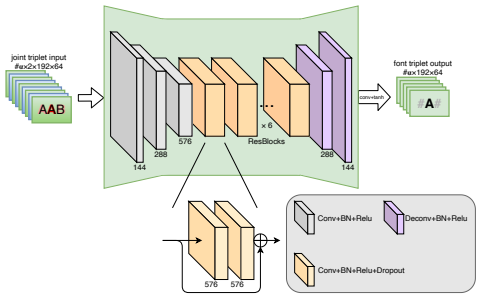

Figure 4: Content generator G_{con} in our Con-XcGAN model.

3.5 Loss Functions

The major learning objective of our end-to-end JointFontGAN model is to generalize both skeleton and content of the observed letters to the unobserved ones. Because there are two generators with different types of generated information (i.e., skeleton and content), we use two XcGAN losses to regulate the adversarial process. From the diagram of the overall architecture of JointFontGAN model in Fig. 1, there are two separate discriminators, D_{sk}, focusing on the correlation of font skeletons, and D_{con}, working on the correlation of font contents. Through our design, the two-stream networks are co-driven and work synergically in an one-stage scenario.

For higher quality results and to stabilize GAN training [32], we use two least squares GAN (LSGAN) loss functions [19] on our local and global discriminators added with an L_1 loss penalizing deviation of generated content images G_{con} from their ground truth y_{con} and an MSE loss penalizing deviation of generated skeleton images G_{sk} from their ground truth y_{sk}. The total loss function is:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_{LSGAN}(G_{sk}, G_{con}, D_{sk}) + \mathcal{L}_{LSGAN}(G_{con}, D_{con}) \\ &\quad + \lambda_{sk}\mathcal{L}_{MSE}(G_{sk}) + \lambda_{con}\mathcal{L}_{L_1}(G_{con}),\end{aligned} \tag{4}$$

where

$$\begin{aligned}&\mathcal{L}_{LSGAN}(G_{sk}, G_{con}, D_{sk}) \\ &= \mathbb{E}_{\tilde{y}_{sk},\tilde{y}_{con}\sim p_{data}}[(D_{sk}(\tilde{y}_{sk},\tilde{y}_{con})-1)^2] \\ &+ \mathbb{E}_{\tilde{x}_{sk},\tilde{y}_{con}\sim p_{data}}[(D_{sk}(G_{sk}(\tilde{x}_{sk}),\tilde{y}_{con})-1)^2] \\ &+ \mathbb{E}_{\tilde{x}_{sk},\tilde{x}_{con},\tilde{y}_{sk}\sim p_{data}}[D_{sk}(\tilde{y}_{sk}, G_{con}(\tilde{x}_{con}, G_{sk}(\tilde{x}_{sk})))^2] \\ &+ \mathbb{E}_{\tilde{x}_{sk},\tilde{x}_{con}\sim p_{data}}[D_{sk}(G_{sk}(\tilde{x}_{sk}), G_{con}(\tilde{x}_{con}, G_{sk}(\tilde{x}_{sk})))^2],\end{aligned} \tag{5}$$

$$\begin{aligned}&\mathcal{L}_{LSGAN}(G_{con}, D_{con}) \\ &= \mathbb{E}_{\tilde{x}_{con},\tilde{y}_{con}\sim p_{data}}[(D_{con}(\tilde{x}_{con},\tilde{y}_{con})-1)^2] \\ &+ \mathbb{E}_{\tilde{x}_{con},\tilde{y}_{sk}\sim p_{data}}[D_{con}(G_{con}(\tilde{x}_{con},\tilde{y}_{sk}))^2],\end{aligned} \tag{6}$$

$$\mathcal{L}_{MSE}(G_{sk}) = \mathbb{E}_{\tilde{x}_{sk},y_{sk}\sim p_{data}}[(y_{sk} - G_{sk}^{chop}(\tilde{x}_{sk}))^2], \tag{7}$$

$$\mathcal{L}_{L_1}(G_{con}) = \mathbb{E}_{\tilde{x}_{con},\tilde{y}_{sk},y_{con}\sim p_{data}}[||y_{con} - G_{con}^{chop}(\tilde{x}_{con},\tilde{y}_{sk})||_1], \tag{8}$$

$$\mathcal{L}_{LSGAN}(G_{sk}, D_{sk}) = \mathcal{L}_{LSGAN}^{local}(G_{sk}, D_{sk}) + \mathcal{L}_{LSGAN}^{global}(G_{sk}, D_{sk}), \tag{9}$$

$$\begin{aligned}\mathcal{L}_{LSGAN}(G_{con}, D_{con}) &= \mathcal{L}_{LSGAN}^{local}(G_{con}, D_{con}) \\ &\quad + \mathcal{L}_{LSGAN}^{global}(G_{con}, D_{con}).\end{aligned} \tag{10}$$

Putting all these loss terms together, the gradients of the above loss functions would be passed through both Sk-XcGAN and Con-XcGAN, and then the end-to-end one-stage training is realized.

4 EXPERIMENTS AND RESULTS

The performance of our JointFontGAN model is extensively evaluated on two datasets mentioned in the following. In this section, we demonstrate the advantage of various components of our proposed model through different comparison and ablation studies on alphabet font generation at first. Next, we show the application of our model on the sentence generation to illustrate the significant improvement on the font consistency. Finally, some other analytic experiments are implemented. All methods in comparison are numerically evaluated by three quantitative metrics, i.e., structural similarity (SSIM), mean squared error (MSE), and L_1 error. The results are qualitatively and quantitatively evaluated. It is noted that for the comparison experiments, best results in the tables are shown in bold font. All the few-shot reference / observation sets for each font style are randomly selected in our experiments.

We train our JointFontGAN network for 600 epochs using Adam optimizer with learning rate as 1×10^{-3}. The batch size is 150, $\lambda_{sk} = \lambda_{con} = 100$. The network is implemented in PyTorch framework. Data and source code of this work will be made available to public.

4.1 Font Dataset Preparation

There are two large-scale font datasets used to evaluate our proposed JointFontGAN method. The first one is Capitals64, collected from [1]. This dataset includes 10K gray-scale Latin fonts with 26 capital letters. The dataset was processed by finding the bounding box of each glyph to create the font image with resolution of 64×64.

The second one is SandunLK64, collected from Sandun.LK [17]. This dataset includes 10K gray-scale Latin fonts with both 26 lowercases and 26 uppercases as well as 8 punctuations. This dataset was designed for editors, graphic designers, web designers, architectural engineering designers, etc. It was processed by finding the bounding box of each glyph to create the font image with resolution of 64×64. Some examples and results of these two datasets are shown in Fig. 5, Fig. 8, and Supplemental Material.

4.2 Font Generation

We first compare our JointFontGAN performance on two large-scale font datasets with two state-of-the-art deep learning based methods of font generation (font style transfer), i.e., zi2zi [25] and Glyph Network in MC-GAN [1]. Both of these two methods are baseline and close approaches for our comparison and they are derived from cGAN model (i.e., the image-to-image translation [11]). The network structure of zi2zi is based on [11] with the addition of category embedding and two other losses, i.e., category loss and constant loss, from AC-GAN [21] and DTN [23], respectively. Glyph Network in MC-GAN is to generalize all 26 capital letters of a font from a few example glyphs by capturing correlations and

similarities among source letters and the unseen ones. It is also based on [11] with the additional L_1 loss in order to force the model to generate images which are close to their targets and further fool the discriminator. Our JointFontGAN model can jointly learn and generate both font skeletons and glyphs at the high quality simultaneously, by using our co-driven Sk-XcGAN and Con-XcGAN subnetworks. For the fair comparison, we choose the same few-shot learning data as the default setting of the original Glyph Network in MC-GAN as well as zi2zi. Then, we have done extensive evaluations on these methods qualitatively and quantitatively. In Fig. 5, we can see that our method can generate more complete shape contents (fewer artifacts) and more consistent font styles compared with zi2zi and Glyph Network on Capitals64 dataset. The selected results demonstrate that our method can generate much better fonts with different styles and contents with a higher consistency. The results have a well-broad coverage of a variety of fonts, such as thin and bold, italic and upright, solid-lined and dotted-line, with and without some textures or patterns, regular and special / artistic, etc. Meanwhile, the visual results of the generated skeletons by our method are also given, which are very similar to the ground truth skeletons. Furthermore, the quantitative performance comparison of these methods is shown in Tab. 1. The numerical evaluation can give a higher-level indication of performance on the whole datasets and it is clear to see that our method has improved the font quality by examining the quantitative metrics. There are some additional visualization comparison results given in Supplemental Material.

Table 1: Quantitative results for comparison and ablation study on two large-scale font datasets.

Dataset	Model	SSIM	MSE	L_1
Capitals64	zi2zi (i.e., cGAN)	0.5642	0.0945	0.1543
	Glyph Network (i.e., w/o XcGAN, Sk-cGAN)	0.6641	0.0753	0.1065
	w/o XcGAN	0.6853	0.0680	0.0947
	JointFontGAN	**0.7217**	**0.0565**	**0.0848**
SandunLK64	Glyph Network (i.e., w/o XcGAN, Sk-cGAN)	0.7925	0.0346	0.0445
	w/o XcGAN	0.8243	0.0272	0.0378
	JointFontGAN	**0.8446**	**0.0233**	**0.0337**

4.3 Sentence Generation

To evaluate the extendability and practicability, we apply our JointFontGAN model to generate some sentences with only a limited number of input letters / words, e.g., the first eight unique characters in our examples. Fig. 6 shows the sentences with capital letters from SandunLK64 dataset. Fig. 7 shows some more challenging cases with both uppercase and lowercase letters, as well as some punctuations from SandunLK64 dataset. It is noted that our model can generate high-quality and high-consistency words and punctuations; it is even better than the ground truth in some cases, such as quotation mark, etc. To our knowledge, this is the first time that a deep neural network can automatically generate both the sentences with corresponding punctuations in a consistent / smooth but different style. Through our model, it is very promising and possible for users to generate large paragraphs with their own handwriting or designed styles by providing very few samples.

4.4 Ablation Study

In this subsection, the goal of our ablation study is to show the importance of the proposed technique components (in Sec. 3) in our JointFontGAN model. We evaluate the effects of XcGAN and Sk-cGAN (i.e., skeleton-based cGAN, but not XcGAN) in our model. The first experiment shows the performance of our full model without both XcGAN and Sk-cGAN components (which is similar to Glyph Network model in MC-GAN [1]) and the second experiment shows the performance of our full model without XcGAN component. Fig. 8 and Tab. 1 show the qualitative and quantitative effects of XcGAN and Sk-cGAN in our model on SandunLK64 dataset. We discover that both of these two proposed key components are very important for our JointFontGAN model and our full model achieves the best results. Furthermore, L_1 error curves (on testing) for ablation study on Capitals64 dataset during the training stage for different epochs are shown in Supplemental Material, which can also explain the effects of the proposed main components.

4.5 Analysis of the Number of Observed Letters

Finally, we analyze the influences of different numbers of the input observed letters in the few-shot learning. It is obvious that the few-shot size could affect the synthesizing performance. Therefore, we conduct experiments to find an optimal (reasonable) setting. In Tab. 2, we compare the performance of our method with three different settings (i.e., one, four, and eight random input samples) on Capitals64 dataset. From both the qualitative (in Supplemental Material) and quantitative (Tab. 2) results, we can see that the more training samples are available, the better quality of results generated by our method is, with clearer shape structures and smaller artifact regions. Although there are just given very few input samples, synthesis results look already visually satisfactory.

Table 2: Quantitative results for comparison with different numbers of few-shot observed letters.

Setting	SSIM	MSE	L_1
#Observed letters = 1	0.4426	0.1262	0.1613
#Observed letters = 4	0.6504	0.0724	0.1030
#Observed letters = 8	**0.7217**	**0.0565**	**0.0848**

5 CONCLUSION

In this work, we propose a new JointFontGAN framework on font generation with a co-driven multi-stream GAN model, which can better capture geometric and content accuracy and consistency of glyphs. Through extensive experiments on two large-scale font datasets, our method has achieved the state-of-the-art performance on font and sentence generation tasks with few-shot references. It has the great capability to enhance texts, sentences, even paragraphs in many multimedia applications. Furthermore, we can apply a two-stage model followed by a (similar) Ornamentation Network in [1] with three RGB channels to our JointFontGAN, to generate colored / ornamented glyphs. Some preliminary results are shown in Supplemental Material. However, the primary goal of our current work is to improve quality of generated (gray-scale) glyphs by adapting to the geometric variability and content scalability. To our knowledge, it is a very tougher task to accurately synthesize skeleton images (a specific glyph image whose stroke width with one pixel) compared to glyph images, we will develop new methods to further improve the quality of generating skeletons and thin glyphs in our future work.

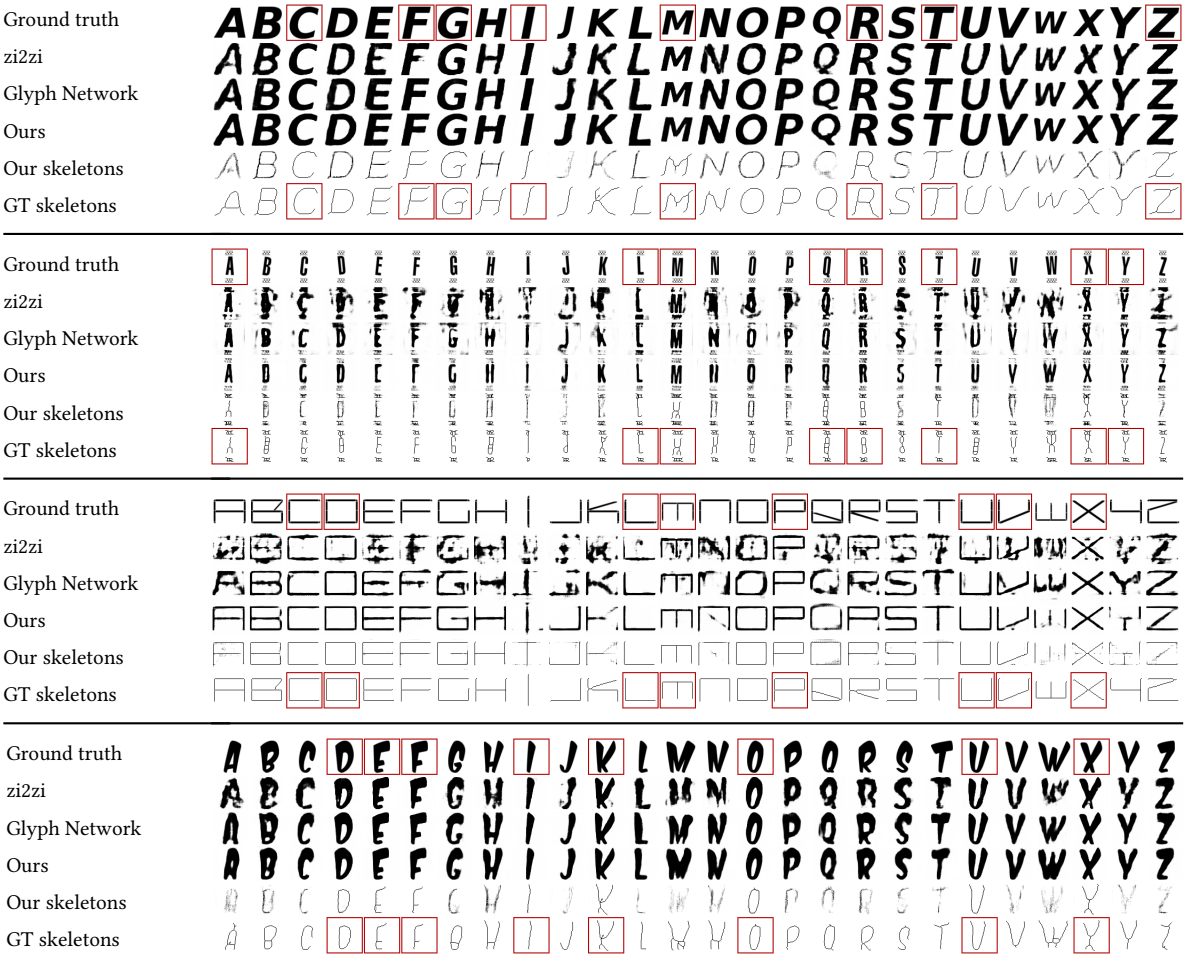

Figure 5: Visualization comparison for our JointFontGAN model on Capitals64 dataset. The ground truth glyphs and the few-shot reference sets (marked in red boxes) are shown in the 1st row. The 2nd row shows the results of zi2zi [25]. The 3rd row shows the results of Glyph Network in MC-GAN [1]. The 4th row shows the results of our model. The 5th row shows our skeleton results. The 6th row shows the ground truth skeletons and the few-shot reference sets (marked in red boxes).

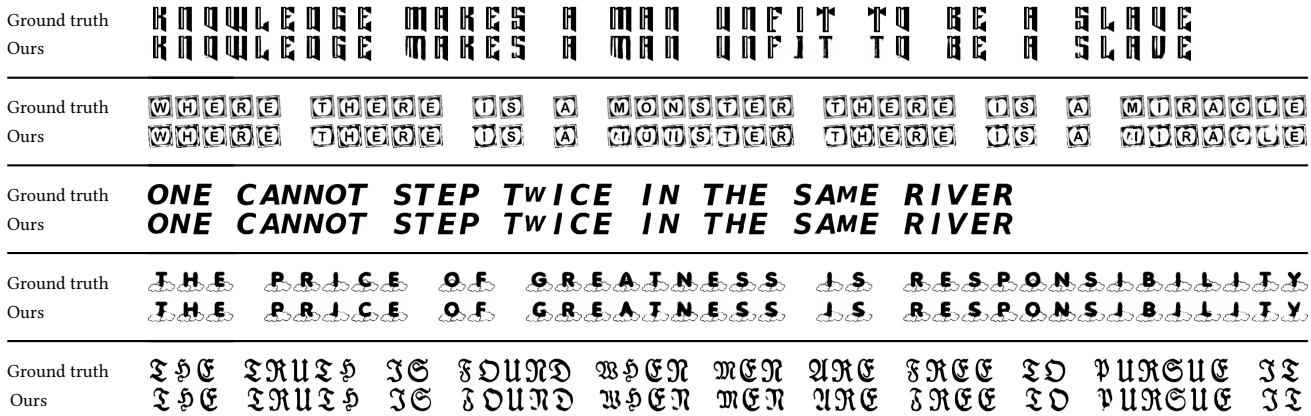

Figure 6: Generated sentences from Capitals64 dataset.

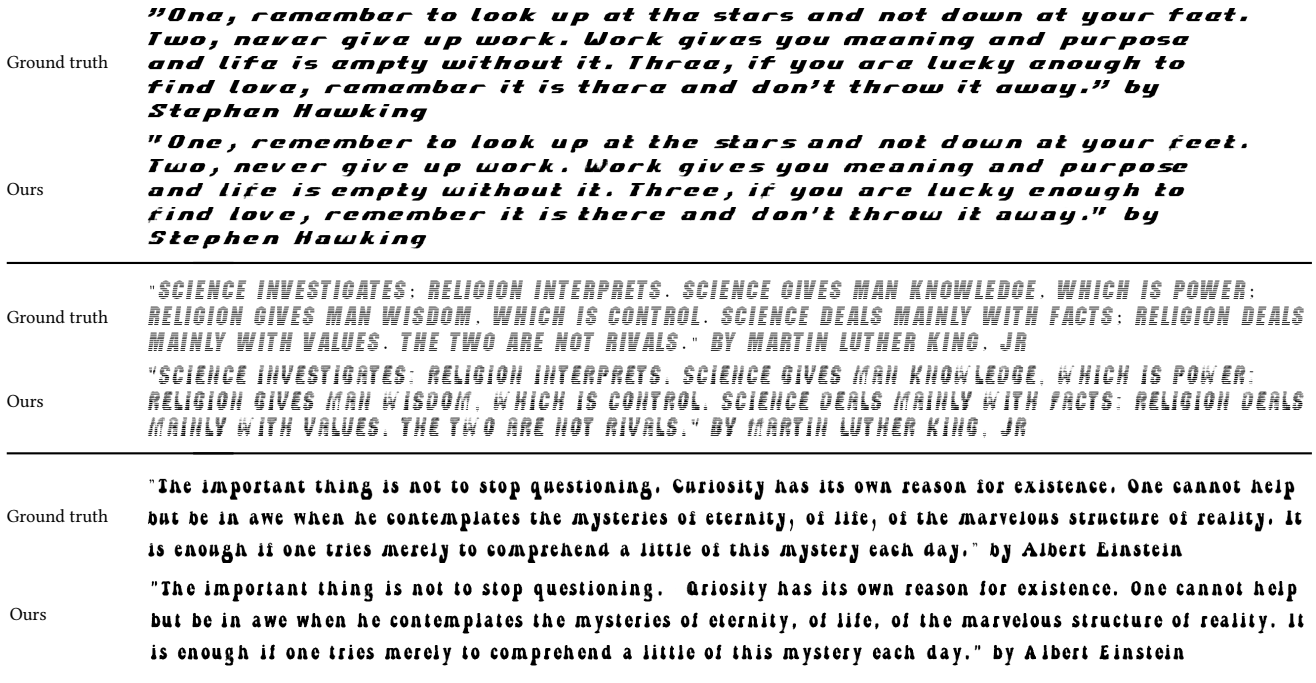

Figure 7: Generated sentences from SandunLK64 dataset.

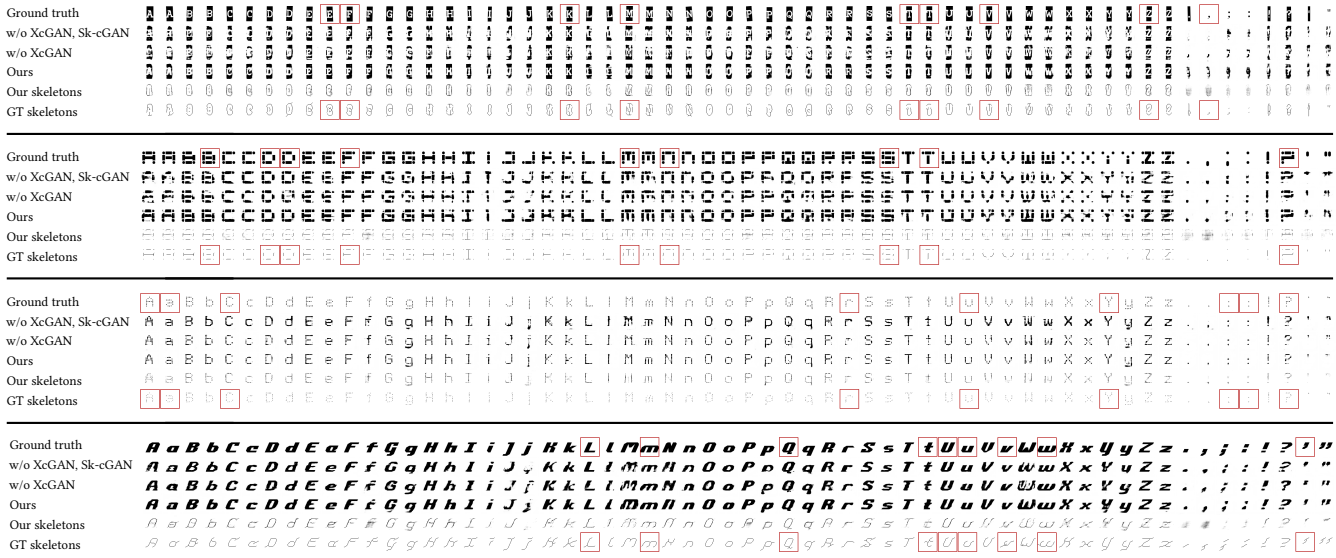

Figure 8: Visualization ablation study for our JointFontGAN model on SandunLK64 dataset. The ground truth glyphs and the few-shot reference sets (marked in red boxes) are shown in the 1st row. The 2nd row shows the results of our model without XcGAN and Sk-cGAN components. The 3rd row shows the results of our model without XcGAN component. The 4th row shows the results of our full model. The 5th row shows our skeleton results. The 6th row shows the ground truth skeletons and the few-shot reference sets (marked in red boxes).

ACKNOWLEDGMENTS

We would like to thank the reviewers for their valuable comments. This work was partially supported by NSF under Grant Numbers IIS-1816511, CNS-1647200, OAC-1657364, OAC-1845962, OAC-1910469, Wayne State University Subaward 4207299A of CNS-1821962, NIH 1R56AG060822-01A1, NIH 1R44HL145826-01A1, ZJNSF LZ16F020002, and NSFC 61972353.

REFERENCES

[1] Samaneh Azadi, Matthew Fisher, Vladimir Kim, Zhaowen Wang, Eli Shechtman, and Trevor Darrell. 2018. Multi-content GAN for few-shot font style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7564–7573.

[2] Elena Balashova, Amit Bermano, Vladimir Kim, Stephen DiVerdi, Aaron Hertzmann, and Thomas Funkhouser. 2019. Learning A Stroke-Based Representation for Fonts. In *Computer Graphics Forum*, Vol. 38. 429–442.

[3] Shumeet Baluja. 2016. Learning typographic style. *arXiv preprint arXiv:1603.04000* (2016).

[4] Ankan Bhunia, Ayan Bhunia, Prithaj Banerjee, Aishik Konwer, Abir Bhowmick, Partha Roy, and Umapada Pal. 2018. Word level Font-to-Font image translation using convolutional recurrent generative adversarial networks. In *Proceedings of the International Conference on Pattern Recognition*. 3645–3650.

[5] Neill Campbell and Jan Kautz. 2014. Learning a manifold of fonts. *ACM Transactions on Graphics* 33, 4 (2014), 1–11.

[6] Bo Chang, Qiong Zhang, Shenyi Pan, and Lili Meng. 2018. Generating handwritten Chinese characters using cyclegan. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*. 199–207.

[7] Jie Chang, Yujun Gu, and Ya Zhang. 2017. Chinese typeface transformation with hierarchical adversarial network. *arXiv preprint arXiv:1711.06448* (2017).

[8] Yue Gao, Yuan Guo, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. 2019. Artistic glyph image synthesis via one-stage few-shot learning. *ACM Transactions on Graphics* 38, 6 (2019), 1–12.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of the Conference on Neural Information Processing Systems*. 2672–2680.

[10] Hideaki Hayashi, Kohtaro Abe, and Seiichi Uchida. 2019. GlyphGAN: Style-Consistent Font Generation Based on Generative Adversarial Networks. *Knowledge-Based Systems* 186 (2019).

[11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1125–1134.

[12] Yue Jiang, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. 2017. DCFont: an end-to-end deep Chinese font generation system. In *SIGGRAPH Asia Technical Briefs*. 1–4.

[13] Yue Jiang, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. 2019. SCFont: Structure-Guided Chinese Font Generation via Deep Stacked Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4015–4022.

[14] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of the European Conference on Computer Vision*. 694–711.

[15] Zhouhui Lian, Bo Zhao, Xudong Chen, and Jianguo Xiao. 2018. EasyFont: a style learning-based system to easily build your large-scale handwriting fonts. *ACM Transactions on Graphics* 38, 1 (2018), 1–18.

[16] Xiyan Liu, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. 2019. FontGAN: A Unified Generative Framework for Chinese Character Stylization and Destylization. *arXiv preprint arXiv:1910.12604* (2019).

[17] Sandun LK. 2018. SandunLK 10K fonts pack. https://sandunlk.home.blog/.

[18] Pengyuan Lyu, Xiang Bai, Cong Yao, Zhen Zhu, Tengteng Huang, and Wenyu Liu. 2017. Auto-encoder guided gan for Chinese calligraphy synthesis. In *Proceedings of the International Conference on Document Analysis and Recognition*, Vol. 1. 1095–1100.

[19] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. 2017. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 2794–2802.

[20] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[21] Augustus Odena, Christopher Olah, and Jonathon Shlens. 2017. Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the International Conference on Machine Learning*. 2642–2651.

[22] Donghui Sun, Qing Zhang, and Jun Yang. 2018. Pyramid Embedded Generative Adversarial Network for Automated Font Generation. In *Proceedings of the International Conference on Pattern Recognition*. 976–981.

[23] Yaniv Taigman, Adam Polyak, and Lior Wolf. 2016. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200* (2016).

[24] Shusen Tang, Zeqing Xia, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. 2019. FontRNN: Generating Large-scale Chinese Fonts via Recurrent Neural Network. 38, 7 (2019), 567–577.

[25] Yuchen Tian. 2017. zi2zi: Master Chinese calligraphy with conditional adversarial networks. https://kaonashi-tyc.github.io/2017/04/06/zi2zi.html.

[26] Paul Upchurch, Noah Snavely, and Kavita Bala. 2016. From A to Z: supervised transfer of style and content using deep neural network generators. *arXiv preprint arXiv:1603.02003* (2016).

[27] Songhua Xu, Tao Jin, Hao Jiang, and Francis Lau. 2009. Automatic generation of personal Chinese handwriting by capturing the characteristics of personal handwriting. In *Proceedings of the Innovative Applications of Artificial Intelligence Conference*.

[28] Songhua Xu, Francis Lau, William Cheung, and Yunhe Pan. 2005. Automatic generation of artistic Chinese calligraphy. *IEEE Intelligent Systems* 20, 3 (2005), 32–39.

[29] Shuai Yang, Jiaying Liu, Zhouhui Lian, and Zongming Guo. 2017. Awesome typography: Statistics-based text effects transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7464–7473.

[30] TY Zhang and Ching Y. Suen. 1984. A fast parallel algorithm for thinning digital patterns. *Commun. ACM* 27, 3 (1984), 236–239.

[31] Baoyao Zhou, Weihong Wang, and Zhanghui Chen. 2011. Easy generation of personal Chinese handwritten fonts. In *Proceedings of the IEEE International Conference on Multimedia and Expo*. 1–6.

[32] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 2223–2232.

[33] Alfred Zong and Yuke Zhu. 2014. StrokeBank: Automating personalized Chinese handwriting generation. In *Proceedings of the Innovative Applications of Artificial Intelligence Conference*.