International Journal on Document Analysis and Recognition (IJDAR) (2021) 24:325–337
https://doi.org/10.1007/s10032-021-00374-4

ORIGINAL PAPER

SKFont: skeleton-driven Korean font generator with conditional deep adversarial networks

Debbie Honghee Ko[1] · Ammar Ul Hassan[1] · Jungjae Suk[1] · Jaeyoung Choi[1]

Received: 9 May 2020 / Revised: 10 May 2021 / Accepted: 26 May 2021 / Published online: 10 June 2021
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2021

Abstract

In our research, we study the problem of font synthesis using an end-to-end conditional deep adversarial network with a small sample of Korean characters (Hangul). Hangul comprises of 11,172 characters and is composed by writing in multiple placement patterns. Traditionally, font design has required heavy-loaded human labor, easily taking one year to finish one style set. Even with the help of programmable approaches, it still takes a long time and cannot escape the limitations around the freedom to change parameters. Many trials have been attempted in deep neural network areas to generate characters without any human intervention. Our research focuses on an end-to-end deep learning model, the Skeleton-Driven Font generator (SKFont): when given 114 samples, the system automatically generates the rest of the characters in the same given font style. SKFont involves three steps: First, it generates complete target font characters by observing 114 target characters. Then, it extracts the skeletons (structures) of the synthesized characters obtained from the first step. This process drives the system to sustain the main structure of the characters throughout the whole generation processes. Finally, it transfers the style of the target font onto these learned structures. Our study resolves long overdue shortfalls such as blurriness, breaking, and a lack of delivery of delicate shapes and styles by using the 'skeleton-driven' conditional deep adversarial network. Qualitative and quantitative comparisons with the state-of-the-art methods demonstrate the superiority of the proposed SKFont method.

Keywords Generative models · Generative adversarial networks · Font generation · Style transfer · Image-to-Image translation

1 Introduction

Font is a fine art that has long been a part of the creative world. Its manifestations have varied from drawings of letter after letter on paper, to copperplate engraving, to film. In the twentieth century, the advent of the computer has resulted in the emergence of the digital font, which, as the personal computer emerged, has gradually made its return to its original attributes, "creative", "unique", and "personalized".

The earliest form of digital font was bitmap fonts, followed by outline fonts, such as Type1 and TrueType, and finally, variable fonts, such as OpenType. However, creating digital fonts still requires heavy human labor. A font designer needs, at the least, months to create a new font set with a couple of strokes, serifs, and thickness. Designing one font style based on the English characters, which are no more than 200 characters including lower and uppercase alphabets, numbers, and symbols, can easily take up to a few months. On the other hand, CJK (Chinese, Japanese, Korean) based languages consist of a large number of characters. For example, the Korean alphabet, Hangul, is composed of 11,172 characters; likewise, Chinese Hanja characters in the Chinese dictionary exceed 50,000. Therefore, it can easily take up to a year or more to design just one font style for the CJK-based languages. Furthermore, the complex structures and shapes of the CJK characters make the font designing process even more difficult for the font designers.

With the recent advances in deep learning-based approaches, high-quality image synthesis tasks can now be resolved in an end-to-end fashion. Recently, the image-to-image translation

✉ Jaeyoung Choi
choi@ssu.ac.kr

Debbie Honghee Ko
debbie.pust@gmail.com

Ammar Ul Hassan
ammar.instantsoft@gmail.com

Jungjae Suk
jjsuk256@gmail.com

[1] School of Computer Science and Engineering, Soongsil University, Seoul, Korea

(I2I) framework "pix2pix" [1], was proposed, based on conditional generative adversarial networks (cGANs) [2], where the goal is to translate a reference input image in one domain into a target output image in another domain, given an input–output image pair as training data.

Until now, the studies based on deep learning merely regard the font synthesis task as an I2I translation problem, where the goal is to learn the mapping from a reference font style in Domain D_R, to any target font style in Domain D_T. This font style domain transfer can be named as F2F, where the first F is a fixed reference font style and the second F is any target font style to be learned. However, there have been ongoing challenges of the F2F-based approaches such as blurriness, unexpected serious artifacts, and non-realistic images especially when the target font style is different from the reference font style. Additionally, due to the complex structures of CJK characters, these F2F-based methods often generate fonts with inconsistent styles and structures (strokes, serifs, etc.). Even if some of the results are acceptable, dealing with delicate touches, such as special stokes/slant/serif details, along with style transfer, is always challenging.

In this paper, we propose an end-to-end skeleton-driven Korean font generative model, SKFont, which transfers the style of a small sample of characters (114) to the structures (skeletons) of unseen characters (2350) to ensure high-quality characters with consistent style and structure. Instead of learning the traditional F2F mapping, we decompose the font synthesis task into three stages:

1. F2F (Font-to-Font) Mapping: By observing a partial set of characters (114), the SKFont learns to translate the style in the reference font into those in the target style. By the end of the first stage, the full set of target font characters (2350) are generated.
2. F2S (Font-to-Skeleton) Mapping: This network learns the structures (skeletons) of target characters which are generated from the first step F2F. It guarantees the detailed structure of each glyph sustained throughout the whole generating processes.
3. S2F (Skeleton-to-Font) Mapping and Style Transfer: With the estimated target skeletons from F2S, the network learns to synthesize them with the target font style.

Finally, by observing a partial set of characters, the complete font characters can be obtained in the desired target font style with consistent style and structure. This skeleton-driven approach, by starting with the skeleton, and later fleshing it out based on the styles, ensures that our model preserves its bone structure. We regard the font synthesis task as an I2I translation problem and utilize conditional GANs for all three steps.

2 Related works

Various trials have been conducted on font generation using deep learning. In this study, we first focus on generic font synthesis methods; thereafter, we mainly focus on methods that consider font generation as an I2I translation problem.

2.1 Generic font generation methods

Inspired by the great advances in image recognition achieved using the convolutional neural network (CNN) [3], "Rewrite" [4] was proposed for synthesizing Chinese calligraphy. This method uses a top-down CNN architecture, where a reference-to-target font image mapping is learned using the L1 loss. The results are often blurry; further, it can only learn one font style at a time.

Neural style transfer [5] is a widely used method for image style transfer. It uses a CNN model to synthesize images using both the content and style of the image. Inspired by this method, a neural font style transfer method [6] was proposed. This method transfers the style of one font image to another font image using the style features extracted from the intermediate layers of the CNN. In this approach, the images of various font styles of different languages, such as Arabic, Japanese, and Korean, are used to generate the font images.

GlyphGAN [7] used a deep convolutional GAN (DCGAN) architecture [8] to generate alphabets. GlyphGAN combines character vector, z^c, and style vector, z^s ($z = z^c + z^s$), as an input to the generator instead of the plain z vector used in the original DCGAN. The problem with GlyphGAN is that the style of the synthesized characters cannot be controlled based on the user's preference, as the style vector, z^s, is always random (sampled from a normal distribution).

2.2 Image-to-image translation methods

I2I translation based on cGAN ("pix2pix", cycleGAN [9]) methods have recently achieved great success. Many font generation methods are inspired by these I2I translation frameworks, where a new font is synthesized from the reference font image provided as an input.

Pix2pix method is used for one-to-one mapping problems, for instance, converting from the edge of a shoe to a complete shoe image, a day time scene to a night time image, etc. On the other hand, font generation can be framed as a one-to-many mapping problem, where the same font character can appear in various target font styles. To overcome the one-to-one mapping problem characterizing the pix2pix method, the zi2zi network [10], which exploits category embedding, was proposed for generating Chinese characters. This method can generate many target fonts from a reference font by combining the pix2pix framework with AC-GAN [11] and a domain transfer network (DTN) [12]. More recently, the DCFont

[13], which combines the category embedding of the zi2zi framework with a font feature reconstruction method for more quality synthesis of Chinese calligraphy was proposed. The DCFont method combines a pretrained VGG network for extracting the font style features and latent features from the encoder and category embedding (borrowed from zi2zi).

Similarly, Chang et al. [14] proposed a method for generating Chinese characters in a personalized handwritten style using the DenseNet CycleGAN. The only difference between this network and CycleGAN is that CycleGAN, by default, uses ResNets [15] blocks after the encoder, whereas DenseNet CycleGAN uses DenseNet [16] instead. Hanfei Sun [17] modified the architecture of zi2zi to synthesize Chinese typography. They also incorporated a pretrained CNN model that extracts the content features of the input character image into the modified architecture. They claimed that their method, because it had strong, soft, and random pairs, was more flexible than the zi2zi network.

Several works have used the skeleton-driven approach to synthesize font characters; however, they either apply a manual skeleton extraction technique, which is a cumbersome task [18], or use a slightly different approach that entails learning handwriting trajectories from the reference style to the target trajectories of the characters [19] by utilizing FlowNet2.0 stacked networks [20].

The common problem with the methods discussed above is that when the reference font style is different from the target font style in overall structure and style, such as strokes and serifs, the synthesized font image is often blurry, broken, or too close to the reference font style. To address the problems mentioned above, we employed a skeleton-driven approach. Compared with related methods, our proposed skeleton-driven approach focuses on learning the skeleton of a given font character based on the I2I translation framework. More specifically, our skeleton-driven approach is inspired by recent advances in skeletonization techniques, mainly the modified U-Net architecture [21] for extracting skeletons from binary images. Our skeletons consist of the overall structure of the font character; therefore, our method extracts the structural information, not the style, of the reference character when it passes through the encoder. We additionally concatenate the style information in the latent features, before passing it to the decoder in all the stages (F2F, F2S, and S2F).

2.2.1 Additional font generation methods

Lian et al. [25] proposed a handwriting font synthesis model, EasyFont, to generate all personal Chinese characters from a small number of samples by learning handwriting strokes and overall handwriting styles. Recent study on generating Chinese handwriting FontRNN [26] focuses on stroke tracing (writing trajectories) method with monotonic attention mechanism.

Lopes et al. [27] proposed a generative model for scalable vector graphics (SVG) using VAE and SVG decoder. This work enables systematic font manipulation and its style propagation. Sun et al. [28] proposed a variational auto-encoder (VAE) framework to synthesize recognizable Chinese content (sentences) transferring style from a few given characters. The model used extra knowledge (the configuration and radical information) which are shared among all the Chinese characters to get more informative content representation.

Baluja et al. [29] is one of the pioneers on English glyph image generation using deep neural networks with a few samples. Azadi et al. [30] proposed a stacked cGAN model (MC-GAN) to generate same style glypes from a few examples, capturing typographic and textual stylization respectively. On the other hand, AGIS-Net [31] is a one-stage model which transfers content and texture styles using two encoders respectively from a few samples.

3 SKFont description

We propose an end-to-end deep neural network to take small sample of characters designed by the font designer in a specific font style and then synthesize the complete set of character images in the sample style. In order to have structure and style consistency, especially for Korean Hangul characters (this can be applied for Chinese), which are complex in shapes and large in numbers, we have designed our architecture to predict the set of the most commonly used Korean Hangul characters (2350) by observing very small examples (114). For this task, we divide this problem into three steps (three subnetworks): predicting the full set of characters from the small samples, structure (skeleton) modeling, and style transfer.

Our first network, F2F, predicts the full set of the target font characters by observing small sample of characters. The second network, F2S, generates the overall structure of the target characters generated by the first network, and our third network, S2F learns to flesh out the generated skeletons by transferring the style. Each of our subnetwork conform to the conditional GAN (cGAN) architecture modified for the specific font problem (one-to-many mapping). The SKFont is pretrained on 85% of the 81 font styles, where each style contains the 2350 most commonly used Hangul characters. To learn a specific font style, the network is fine-tuned on 114 sample characters, and then a complete font set (2350) is generated in that specific font style. In the following sections, we first describe the cGAN model briefly, and then discuss the architecture and details of our proposed three-stage network architecture and the loss functions.

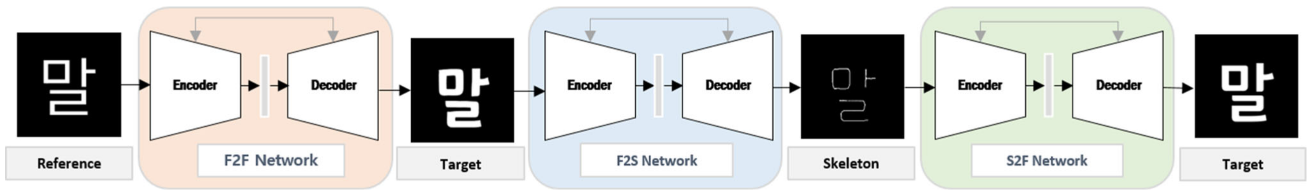


Fig. 1 Overview of our subnetworks F2F, F2S, and S2F networks, respectively, with their respective references and the generated images

3.1 cGAN

The vanilla GAN [32] is composed of two models: a generator and a discriminator. A generative model $G(z;\theta_g)$ is a differentiable function with parameter θ_g, which learns the distribution of sample data, x, from a prior input noise variable $p_x(z)$. G's goal is to mimic the distribution as close as possible to x. On the other hand, the discriminator model, $D(x;\theta_d)$, determines whether the sample it takes is from G (generator's distribution p_g) or the ground truth (GT) data x. The output of D is a single scalar [0–1] as a probability.

D and G appear to play a minimax game with value function, $V(G, D)$. G and D are trained simultaneously. The model adjusts the parameters for D to maximize the probability of both the real samples and the generated samples. On the other hand, it adjusts the parameters for G, to minimize the probability to only the generated samples,

$$\min_G \max_D V(D,G) = E_{x\sim p\mathrm{data}(x)}[\log D(x)] + E_{z\sim pz(z)}[\log(1-D(G(z)))]. \tag{1}$$

The cGAN is simply an extension of the vanilla GAN, wherein any conditional information, c, is added to both the generator and the discriminator for the parameter intended to be controlled.

$$\min_G \max_D V(D,G) = E_{x\sim p\mathrm{data}(x)}[\log D(x|c)] + E_{z\sim pz(z)}[\log(1-D(G(z|c)))]. \tag{2}$$

We follow the cGAN settings in each of our subnetworks (F2F, F2S, and S2F) to generate the whole set of characters with style and structure consistency by observing limited characters. We use a reference image as an input to the generators in all the subnetworks as the condition, and ignore the noise, z, similar to pix2pix model.

3.2 SKFont architecture

Our model consists of three subnetworks: Font-to-Font network (F2F), Font-to-Skeleton network (F2S), and Skeleton-to-Font network (S2F), as depicted in Fig. 1.

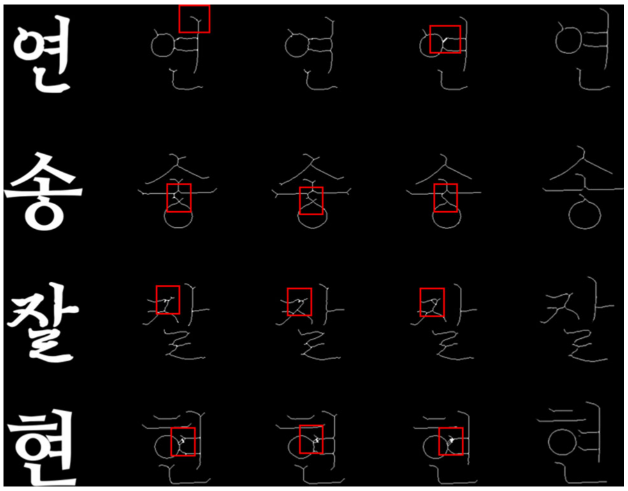


Fig. 2 Problems of existing skeletonization mathematical approaches vs F2S. By column order: GT font, [22–24], and F2S (ours) methods respectively

3.2.1 F2F network

The F2F network conforms to the same principle underlying the traditional methods for generating a full set of font characters; it learns a target font style from a fixed reference font style. The reference font is chosen based on the most common style and structure of the Hangul fonts; i.e., the overall structure and style of almost all Hangul font styles can be derived from this font. The input to this network is a reference font image x which is downsampled via an encoder to extract the high-level features. Here, we combine a style vector with the encoded features. This latent is then passed through a series of upsampling layers to generate the target font image. More network details are described in Sect. 3.3.

3.2.2 F2S network

Our main goal for employing a traditional F2F network is to estimate the full set of target domain font characters from the small sample characters. We then utilize the F2S generator that learns to translate the estimated font character from F2F to its corresponding skeleton. The training target skeletons are produced using a Python-based model that used Lee's method proposed in this work [24]. The reason for training a GAN-based F2S network over mathemati-

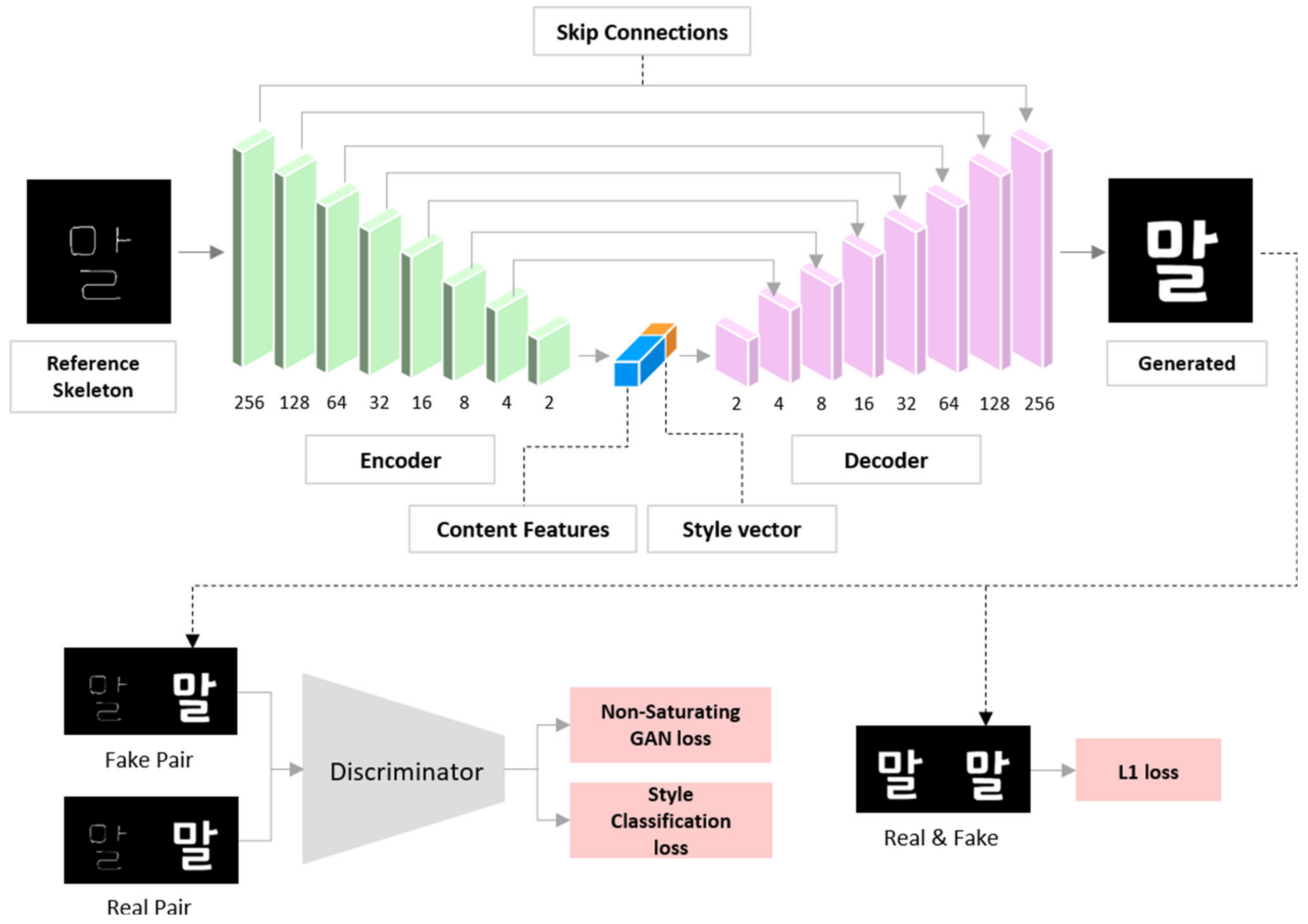


Fig. 3 Architecture of skeleton-to-font (S2F) network. Our generator takes a skeleton character as an input, and passes it through the encoder to obtain the content features of a given skeleton. The content features are then combined with a styled vector to recover the style-related details of the target font glyph via the decoder

cal algorithms was that we found some problems, such as disjointedness, attached complicated elements, and serious artifacts in the skeletons generated by [24] (and other mathematical approaches [22,23]), as shown in Fig. 2.

3.2.3 S2F network

Finally, the S2F network learns to transfer the style to the synthesized skeletons from the F2S network to the target fonts. In all of the networks, we use the same architecture, but learn different mapping functions; in the F2F, the reference domain is a specific font style and the generator learns the target domain font style by observing 114 sample characters. While in the F2S, the reference image is from the font domain and the generator learns to transform it into the corresponding skeleton domain. And in the S2F, the generator learns to style transfer from the skeletons generated by F2S to the target font style. In the next section, we present the network details of our model.

3.3 Network details

For all of the subnetworks, we use the same network architecture, therefore we only present the details of the S2F network, as shown in Fig. 3 (applicable to all the networks with respective reference and target images). In all networks, the generator employed a U-Net architecture [33], instead of the vanilla autoencoder. The only difference is that U-Net uses skip connections from every encoder layer to the corresponding decoder layer except the last layer and the first layer of the decoder. The reference image is passed through the series of downsampling layers (encoder) to obtain high-level features of the image, which we call the "content features". Then these content features are concatenated with a vector named "style vector" which is used to distinguish multiple styles. The style vector used in our networks is a one-hot encoded vector indicating various styles. The style vector in our network plays the key role for learning various font styles. During training time, our network learns various font styles based on this one-hot encoded style vector. For the unseen font style, we do a fine-tuning step. The goal of this fine-tuning step is to learn the new style vector (one-hot encoding) of the new font style. The dimension of the style vector is evaluated based on the total number of train and test fonts. For example, the content features from our encoder have a dimension of (batchsize, 1, 1, 512). Then, we concatenate these features with the style vector (batchsize, 1, 1, length(train+test font styles)). This concatenated vector is passed to the decoder that upsamples (reverses the encoder operations) to generate the target image. The skip connections between the encoder and decoder layers ensure that the rich information is not lost during downsampling.

PatchGAN method [1] is employed in our discriminator model. It only penalizes structure at the scale of patches. A normal discriminator is determined by the input image; this discriminator outputs a scalar for a real or fake image, whereas the patch discriminator runs convolutionally across the image, and outputs a patch (30×30) where each patch in the image is either real or fake. In this process, the responses are averaged for the final decision. The purpose of this is to obtain more detailed results. We additionally add fully connected layers at the end of the discriminator to predict the styles of the generated images. This layer helps in controlling embeddings to generate a specific style image (skeleton or font).

3.4 Loss function

The loss function of our SKFont consists of an adversarial loss, style classification loss, and L1 loss. Because of the recent success of adversarial networks in generative problems, we utilize an adversarial loss. More specifically, we use the Non-Saturated GAN loss for both the generator and discriminator. We found with our experiments that this loss is more stable and converges quickly, compared to the saturated loss [34].

The vanilla cGAN generates the data distribution from the noise distribution ($G : z \rightarrow y$); however, our SKFont learns a mapping from the reference (skeleton or font) domain to the target domain ($G : \{x\} \rightarrow y$), where x is the input image of the reference domain, and y is GT of the target domain. Unlike the vanilla discriminator that downsamples the input image to a one-dimensional scalar, our discriminator classifies whether the $N * N$ patch is real or fake. Our GAN's non-saturated $\text{Loss}_{\text{cGANns}}(G, D)$ objective function is formulated as follows:

$$\text{Loss}_{\text{cGANns}}(D) = E_y[\log D(y)] + E_x[\log(1 - D(G(x)))], \tag{3}$$

$$\text{Loss}_{\text{cGANns}}(G) = -E_x[D(G(x))]. \tag{4}$$

Target domain font images can be of various styles, where a one-character skeleton can correspond to many target character font styles. For example, the skeleton of one Hangul character can have a variety of different styles in the target domain. To control the target font image style, we employ a style classification loss. This loss corresponds to the style vector which is concatenated with the structure vector, before being passed on to the decoder (unlike in the pix2pix or cycleGAN methods). This loss function in our network serves as a style transfer unit. From a given reference image of a Hangul character (skeleton or font), the style vector and style classification loss forces our generator to generate font images in various target domain font styles. The discriminator's job is not only to check if the input image pair is real or fake, but also check whether the generated image is of the same style as the target font style. Our style classification loss implied from the GAN loss is as follows:

$$\begin{aligned}\text{Loss}_{\text{cGANsc}}(G, D) = {} & E_y[\log D_{\text{sc}}(y)] \\ & + E_x[\log(1 - D_{\text{sc}}(G(x)))].\end{aligned} \tag{5}$$

In order to generate font images that have the same overall structure as the target domain font image, we use the $L1$ loss. The $L1$ loss attempts to minimize the pixel-by-pixel difference between the generated image, $Y^{\wedge}$, and the corresponding target image, Y. Further, the $L1$ loss minimizes the blurring, unlike the $L2$ loss. This loss is formulated as follows:

$$\text{Loss}_{\text{L1}} = E_{x,y}[||(y - G(x))||_1]. \tag{6}$$

By combining these three loss functions, we formulate our final loss function of the proposed methods as

$$\begin{aligned}G^* = {} & \arg\min_G \max_D \text{Loss}_{\text{cGANns}}(G, D) \\ & + \text{Loss}_{\text{cGANns}}(G, D) + \lambda \text{Loss}_{\text{L1}}.\end{aligned} \tag{7}$$

4 Experiments and results

In this section, we first describe the dataset and implementation details of our proposed method. Then, we evaluate the performance of the proposed method from various perspectives to verify the proposed model's advantages over other approaches.

4.1 Dataset

Hangul is composed of 11,172 syllables and has multiple patterns in its construction. There are 19 consonants and 21 vowels in the Korean alphabet (Table 3 in Appendix A.1). To construct Korean characters, we need to place those alphabet patterns in phonetic order using six placements (Fig. 9 in Appendix A.1).

We validate our model with 81 Korean font styles (we gathered our Korean public fonts from the following source[1]). Each font consists of 2350 of the most commonly used Korean characters. We trained our model with 85% of these collected font styles. For testing the generalization capability of our model, we used the remaining 15% unseen font styles (the font which were not observed by the model during training time).

For our F2S and S2F networks, where the reference or target is the skeleton of a given font character, we used a Python-based module that generates the skeletons of font characters. We generated skeletons from the state-of-the-art mathematical algorithms of skeletonization [22–24]. Then through visual analysis, we selected Lee's skeletonization method [24] for pretraining our networks, because this algorithm generates reasonable font skeletons, compared to other algorithms. We then prepared a paired dataset where, for F2S network we had font characters in the reference domain and corresponding font skeletons in the target domain. For the S2F network, we prepared the inverse of the F2S networks paired dataset, i.e., font skeletons in the reference domain and corresponding font characters in the target domain.

For the experiments, we tested various fonts based on the overall style and structure of the characters in the reference and target domains, to evaluate the model's diversity in generating various fonts in different styles. We selected 114 characters that cover the combinations of 114 first consonants (Chosung), 42 medial consonants (Joongsung), and 27 final consonants (Jongsung) as the input set. With these basis characters of new unseen style, our model generates the rest of the characters in the same style, i.e., with 114 samples, our model can see sufficient examples to understand the overall structures of the glyphs, even those with complicated structures and different styles from the reference font.

4.2 Network details and parameter settings

For our experiments, the input and output character images are both $256 \times 256 \times 3$ (RGB). We tested various data sizes: 64×64, 128×128, and 256×256, and discovered that 256×256 was the best. A machine recognizes object images as a collection of pixels and it needs to look at every single pixel, resulting in slow processing time: for our training dataset images of size $256 \times 256 \times 3$ that involve approximately 40 million pixel's calculations, it took three days to train our model.

For all of the networks, the encoder contains seven downsampling layers. Every layer in the encoder consists of a convolution operation followed by Batch Normalization [35] and Leaky ReLU activation function [36,37] except the first layer where Batch Normalization is not used. We used 2×2 stride in all layers, except the last, where we have a stride of 1; the batch size was set to 1, and the learning rate was 0.0002, decayed by half after 10 iterations. We trained our model using the Adam optimizer. The decoder consists of seven upsampling layers. Each layer has a deconvolution operation followed by Batch Normalization and ReLU activation function. As an exception to the above operation flow, Batch Normalization is not applied to the last layer of the decoder, and the tanh activation function is used instead of ReLU.

After pretraining, when it comes to learn an unseen new font style, we fine-tune our model with the 114 sample characters that represent the overall structure of all the other Korean Hangul characters. This fine-tuning process helps the network to learn the new font style with its specific style embedding. This way the pretrained model converges fast compared to the one trained from scratch as it has already seen various Hangul characters of various shapes and styles resulting in our model being robust for various target font styles. After learning the new font-style embedding during the fine-tuning phase, our network generates the rest of unobserved 2236 characters in the newly learned style during testing (this can easily be extended to generate 11,172 characters).

4.3 Performance evaluation

In this subsection, we compare our model qualitatively with some recently proposed methods to verify the effectiveness of our model. Additionally, we also compare the results quantitatively.

4.3.1 Qualitative evaluation

We used pix2pix [1] and zi2zi [10] as our baselines, and compared them to our proposed SKFont generator. For a fair comparison, we pretrained all three models using the same dataset, and fine-tuned them with the same unseen styles. Both baselines are trained using the implementations provided by the authors.

As shown in Fig. 4, our method produces non-blurry and photorealistic typefaces showing great dominance over others in visual appearance. Although the other methods can style transfer the overall font style, the synthesized font images are mostly of low quality with serious artifacts. When the synthesized images are zoomed, the results yielded by the baselines exhibit poor smoothness, broken strokes, and blurriness. When the target font style (BinggraeTaomB) is intensely different from the reference font style, broken results and poor quality are observed in the baselines. On the other hand, SKFont preserves the consistency of style and

[1] https://software.naver.com/software/fontList.nhn?categoryId=I0000000.

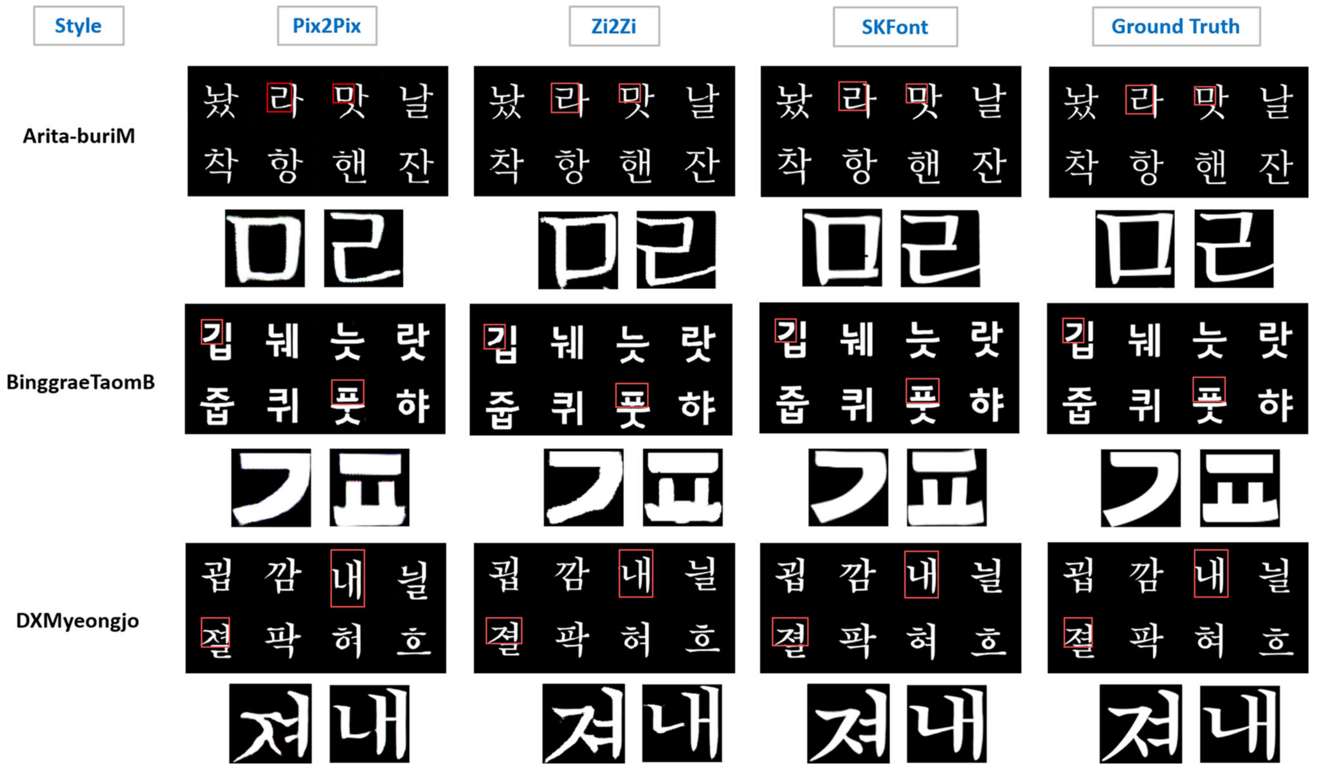


Fig. 4 Comparison of synthesized glyphs in three different styles using the proposed model and other methods

structure while ensuring the smoothness of the font images. In Appendix A.2, we have attached more synthesized images of our model compared with the *GT* and the other two methods.

In addition to comparing the effectiveness of our method with baselines, we also compared the characters generated by our model and the ground truth basis characters. From the 114 basis characters designed by the designer, we generated a complete set of commonly used font characters using the SKFont, and combined them together to generate paragraphs to visually estimate the quality of these synthesized characters. As shown in Fig. 5, we generated a sentence based on three font styles using our method. The highlighted characters (characters in red box) are the *GT* basis characters that were used during the fine-tuning process to learn a specific font style. Other characters are generated by our model in the corresponding font style. From Fig. 5, we can clearly observe that the characters generated by the SKFont are almost consistent in style (strokes and thickness) and are very nearly indistinguishable from those designed by the font designer.

In Fig. 14 (Appendix A.4), we visualize more unobserved font styles generated by our proposed model and the figure shows that the font images synthesized are photorealistic.

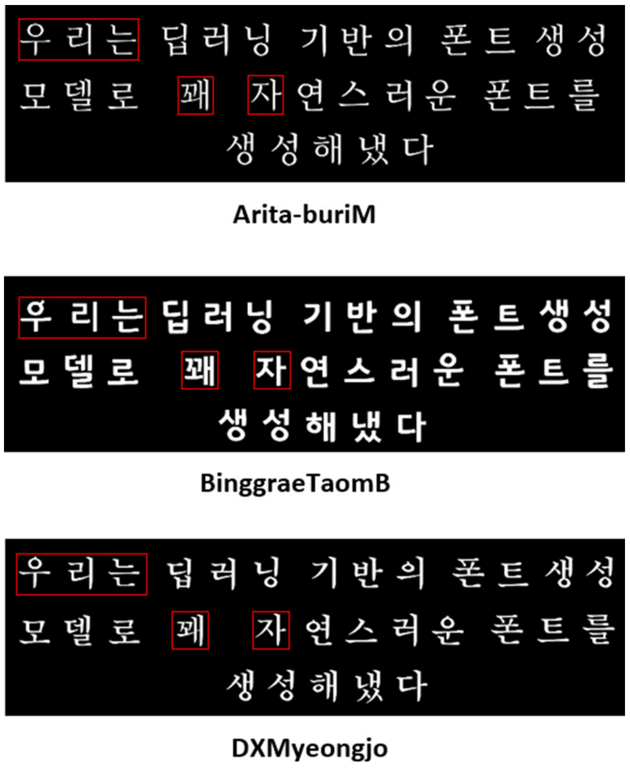

우리는 딥러닝 기반의 폰트 생성 모델로 꽤 자연스러운 폰트를 생성해냈다
Arita-buriM
우리는 딥러닝 기반의 폰트 생성 모델로 꽤 자연스러운 폰트를 생성해냈다
BinggraeTaomB
우리는 딥러닝 기반의 폰트 생성 모델로 꽤 자연스러운 폰트를 생성해냈다
DXMyeongjo

Fig. 5 Text rendered in three different styles generated by our method SKFont. The GT basis characters by human designers are highlighted in red; the rest are SKFont generated characters in the corresponding font style (Color figure online)

Table 1 Quantitative evaluation of proposed method and two other methods

Font styles	Pix2pix			Zi2zi			SKFont		
	HCCS	L1 loss	L2 loss	HCCS	L1 loss	L2 loss	HCCS	L1 loss	L2 loss
Arita-buriM	0.9603	0.2635	0.2529	0.9779	0.2661	0.2521	0.9955	0.2304	0.2275
BinggraeTaomB	0.9158	0.3347	0.2632	0.9236	0.2612	0.2475	0.9234	0.2496	0.2439
DXMyeongjo	0.9451	0.2682	0.2564	0.9583	0.2720	0.2553	0.9789	0.2549	0.2515
Interpark Gothic	0.9326	0.2589	0.2497	0.9517	0.2657	0.2567	0.9824	0.2311	0.2267
Jeju Myeongjo	0.9681	0.2412	0.2265	0.9791	0.2478	0.2411	0.9716	0.2489	0.2359
KBIZmjo L	0.9120	0.3048	0.2716	0.9216	0.2879	0.2655	0.9425	0.2368	0.2214
KoPubWorld	0.9238	0.2789	0.2899	0.9105	0.2941	0.2874	0.9368	0.2741	0.2694
SunBatang	0.9015	0.3814	0.3569	0.9087	0.3776	0.3655	0.9115	0.3213	0.3102

4.3.2 Quantitative evaluation

We also performed a quantitative evaluation of the proposed model against the baselines. For the quantitative evaluation, we computed the L1 and L2 losses between the synthesized image and the *GT* image. Additionally, we computed the Hangul Character Classification Score (HCCS). For the HCCS, we pretrained a CNN model to correctly classify the Hangul characters. This CNN model achieves a 99.4% testing accuracy on real Hangul characters. At inference time, we calculated the HCCS by predicting the character's label using our model and the baselines. As shown in Table 1, we can see that our model outperforms the other two methods on various font styles. These font styles were randomly chosen from the test set for evaluating quantitative metrics. Our model achieves the lowest L1 and L2 loss and the highest HCCS accuracy, which clearly shows our proposed method's superiority.

4.4 Generalization capability of SKFont

To test the generalization capacity of our proposed model, we conducted an experiment. We performed a cross-language evaluation test, Korean to Chinese and English. These Chinese and English characters are new to the model, i.e., this model has never seen English and Chinese fonts during pretraining and fine-tuning phases.

For this experiment, we fine-tuned our pretrained model on specific Korean Hangul styles. Style 1 was chosen because of its specific strokes (Korean Ming font style), whereas Style 2 based on its thickness (Korean Gothic font style). For the contents, a simple font style is chosen (without special strokes or thickness) as shown in Fig. 6.

Then, we fed some unseen English and Chinese characters to this fine-tuned model as content images. As depicted in Fig. 6, the model is able to synthesize the images of these unseen language characters with an impressive quality. The row of the Reference Style 1 and Style 2 show that the model is able to synthesize the unseen Chinese and English characters by learning the Korean reference font Style 1 and Style 2, respectively. This experiment demonstrates the generalization ability of our proposed model even for the cross-languages. This test also demonstrates that with the skeleton-driven approach, the model is able to transfer the style on the content images.

4.5 Ablation study

4.5.1 Effectiveness of skeletonization introduced in SKFont

To investigate the effectiveness of skeletonization introduced in our proposed SKFont method we performed an ablation study. For this ablation study, we compared our proposed method (three-stage F2F–F2S–S2F model) with a single F2F network that directly synthesizes target font style. We trained both methods on same number of font styles and then evaluated on unseen fonts (font styles not used during pretraining). All the hyperparameters and network structures were same for both F2F and SKFont method.

We observed via visual comparisons that the proposed SKFont method based on skeletonization smooths the synthesized images and unlike the F2F method, the breaking and blurry problems of synthesized characters are not found. This phenomenon is demonstrated in Fig. 7.

Additionally, we conducted quantitative analysis to compare our proposed skeletonization-based method to the F2F method. For this experiment, we calculated the average SSIM and L1 distances between the ground truth and the synthetic images generated by both methods across the entire testing set discussed in Sect. 4.1. Table 2 contains the comparison results. We can see that in both metrics, the proposed method with skeletonization outperforms the single stage F2F method.

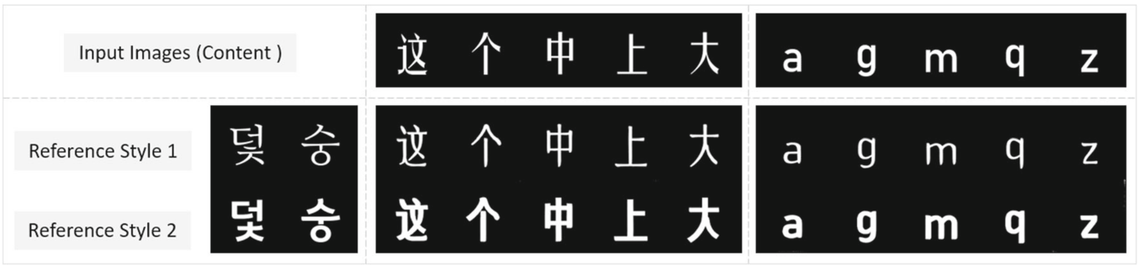


Fig. 6 Chinese and English font synthesis using the SKFont model with two Korean styles (Reference Style 1 and 2). First row demonstrates the input images for content

4.5.2 Number of basis characters in fine-tuning phase

We also performed an ablation study to explore the relationship of the quality of synthesized font characters by changing the number of basis characters in the fine-tuning step for learning unobserved font styles.

For this experiment, we trained SKFont with three different basis character sets; i.e., default 114 basis characters, 256 basis characters, and 512 basis characters (the latter two were chosen based on the Handwritten Korean Character Recognition project [38]).

For evaluation, we performed qualitative and quantitative results as shown in Fig. 8. For the quantitative evaluation, we computed the structural similarity index measure (SSIM) between the ground truths and the synthesized characters for each model (model fine-tuned with 114, 256, 512 basis characters respectively).

The results in Fig. 8 depict few things: first, our selected special 114 basis characters are influential in getting good quality results that closely match the large number of the basis characters quality. Secondly, as expected with the large number of basis characters (512), the quality of the synthesized characters positively increases; however, the goal is to use less number of basis characters and produce high quality results.

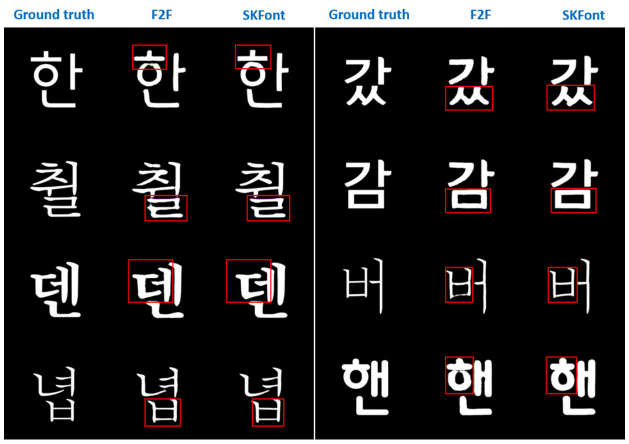
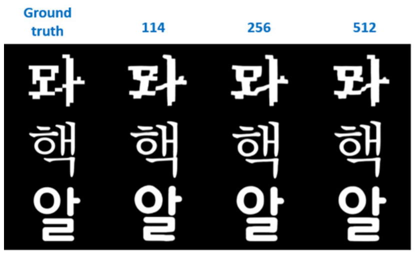
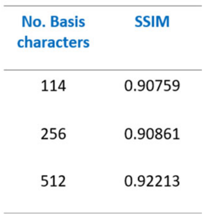


Fig. 7 Single stage F2F versus proposed three-stage SKFont

Table 2 Examining the effectiveness of skeletonization through a quantitative comparison of F2F and the proposed SKFont method

Method	SSIM	L1 loss
F2F	0.9025	0.2623
SKFont	0.9287	0.2441





No. Basis characters	SSIM
114	0.90759
256	0.90861
512	0.92213

Fig. 8 Effect of fine-tuning basis characters

5 Discussion and future work

Here, we describe the theoretical intuition behind the results of our model. Related methods learn from a fixed reference font style to synthesize any target font style. Employing the reference font image as an input to the encoder results in the synthesized image close to the reference font image, since the downsampled latent features are the combination of both the content and style attributes of the reference font image. Ideally, we would like the encoder to downsample the structural attributes of the reference font image as the latent features. Based on this latent, the decoder upsamples the target domain font style, using separate font style information. In such cases where both the target and the reference images are very different in style, the generator gets confuse, and starts generating blurry, broken, or confused images (that are closer to the reference domain font style) as outputs.

Our first F2F network also uses a fixed reference font style to synthesize any target font style; however, our goal to deploy this model is to generate the full set of target characters in the fine-tuning phase, where we have a small number of target characters (114 in our case). F2F network is deployed to learn the target domains full set of characters (2350 in our case). This network helps in estimating the rough style and structure of all the target domain characters. Once we have these characters, the second stage network extracts the skeletons from these estimated target domain characters. Finally, the third network style transfers the target domain font style onto these skeletons. This three-step learning, unlike the single-stage learning of the baselines, impacts the final results significantly.

For our future work, we shall examine font synthesis as a multi-domain I2I translation problem. Currently, our three-stage method composed of the F2F, F2S, and S2F networks has limited scalability and robustness in tackling the three domains, as a result of which we have to train different models for each of the paired datasets. However, we wish to use a single generator to learn the multiple domains. This not only improves the scalability and robustness of our model, but can also greatly improve the training performance. We would like to continue working on synthesizing high-quality fonts with very cursive or artistic styles.

6 Conclusion

In this study, we designed a skeleton-driven Korean font generator using an end-to-end conditional deep adversarial network, with small sample of Korean characters (Hangul). We observed that this model provided robust photo-realistic results, and also markedly improved the traditional problems in font synthesis models, such as blurriness, severe artifacts, and non-photo realistic results. Compared with the baselines, the experimental results of our "skeleton-driven" approach exhibited outstanding qualities based on visual perceptions, and as established by quantitative evaluations. The generalization capability of the SKFont demonstrates that it can be used for transferring the style of one language to another, as depicted in the experiments.

Funding This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2016-0-00166).

A Appendix

A.1 About Hangul

There are 11,172 Korean (Hangul) syllables. They can be constructed in six ways (Fig. 9). The syllable blocks are arranged in phonetic order, the initial (Chosung), medial (Joongsung), and final (Jongsung).

There are 19 consonants (14 singles + 5 doubles) for Chosung, 21 vowels (10 basics + 11 combined) for Joongsung, and 27 consonants (14 basics + 11 combined + 2 double) for Jongsung; Here, "single" indicates one consonant, "double" indicates a doubled consonant, and "combined" indicates two different consonants (Table 3).

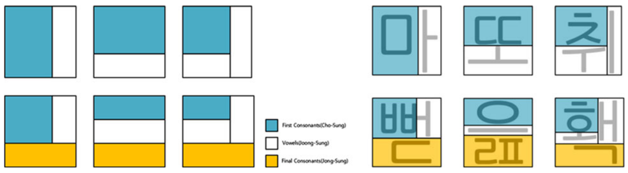


Fig. 9 Hangul placements and their examples

Table 3 19 Consonants and 21 vowels for Hangul

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
First Consonants (Cho-Sung)	ㄱ	ㄲ	ㄴ	ㄷ	ㄸ	ㄹ	ㅁ	ㅂ	ㅃ	ㅅ	ㅆ	ㅇ	ㅈ	ㅉ	ㅊ	ㅋ	ㅌ	ㅍ	ㅎ									
Vowels (Jung-Sung)	ㅏ	ㅐ	ㅑ	ㅒ	ㅓ	ㅔ	ㅕ	ㅖ	ㅗ	ㅠ	ㅘ	ㅛ	ㅙ	ㅚ	ㅜ	ㅝ	ㅞ	ㅟ	ㅡ	ㅢ	ㅣ							
Final Consonants (Jong-Sung)	☒	ㄱ	ㄲ	ㄳ	ㄴ	ㄵ	ㄶ	ㄷ	ㄹ	ㄺ	ㄻ	ㄼ	ㄽ	ㄾ	ㄿ	ㅀ	ㅁ	ㅂ	ㅄ	ㅅ	ㅆ	ㅇ	ㅈ	ㅊ	ㅋ	ㅌ	ㅍ	ㅎ

A.2 More comparison results with other models

See Figs. 10, 11 and 12.

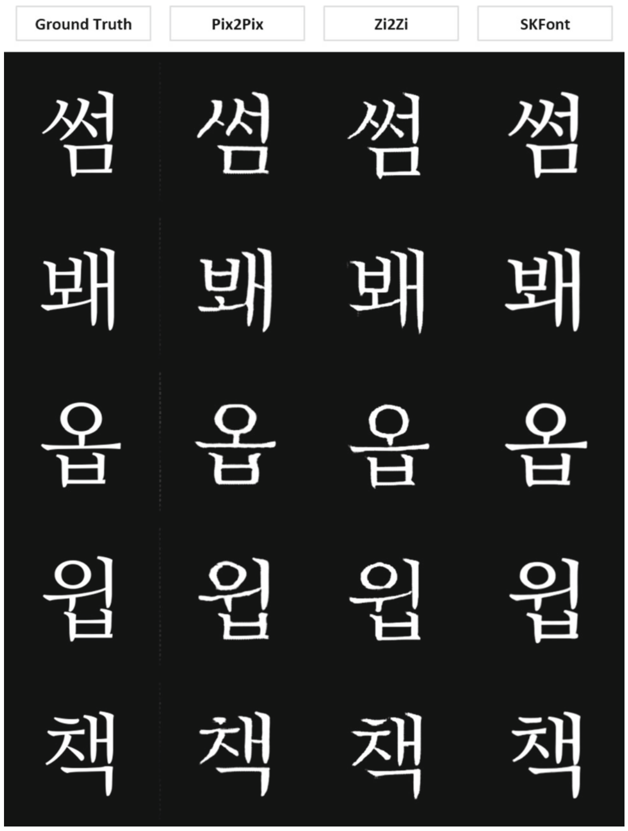


Fig. 10 Style 1: Arita-buriM

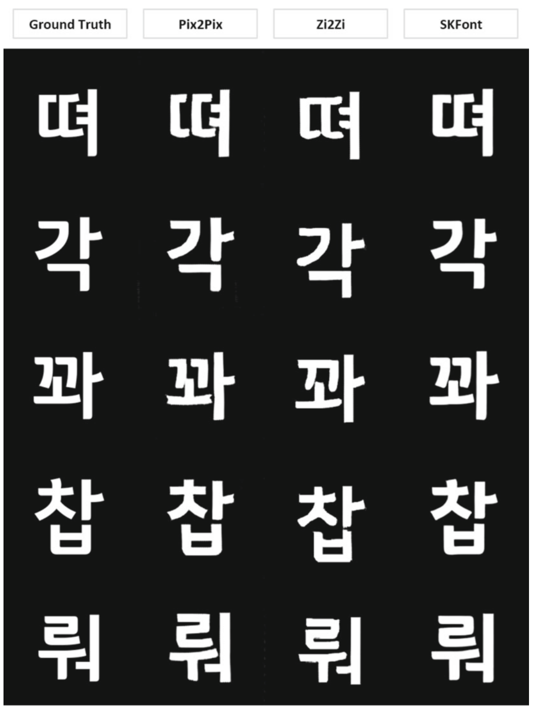


Fig. 11 Style 2: BinggraeTaomB

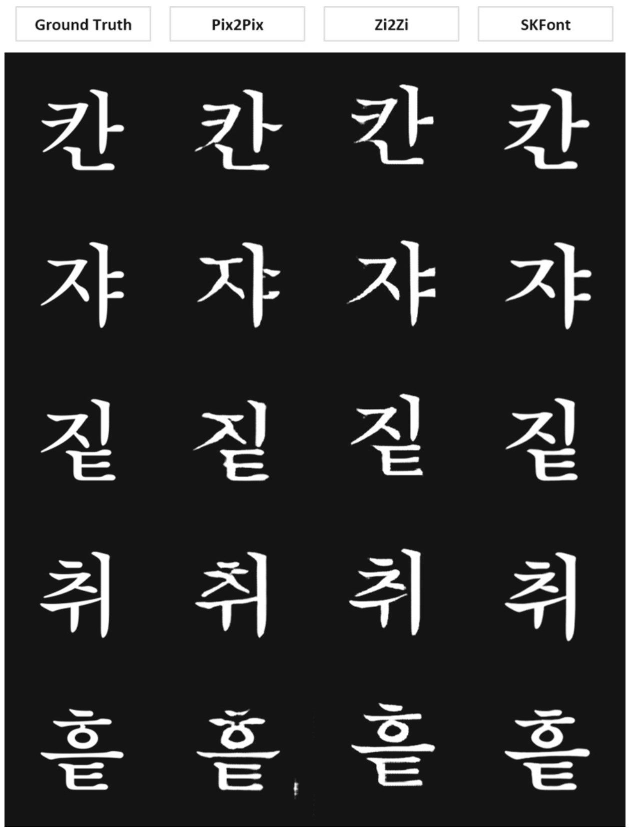


Fig. 12 Style 3: DXMyeongjo

A.3 Generating stylized font styles

We also fine-tuned the proposed model to synthesize cursive and pixel based stylized font styles. As shown in figure below our model can synthesize these font styles in a decent quality although these kind of font styles were not used in the pre-training phase (Fig. 13).

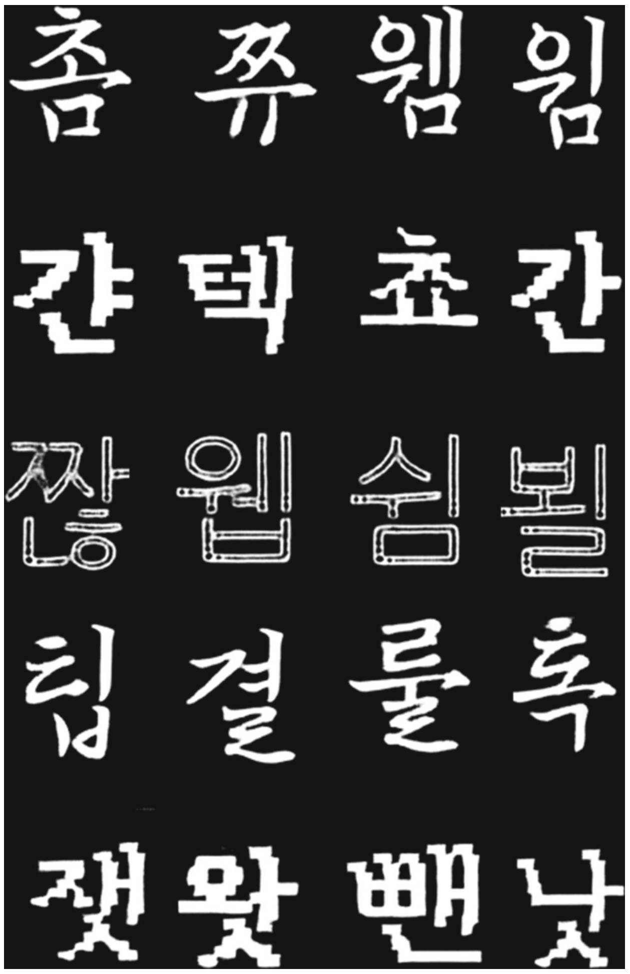


Fig. 13 Cursive and Pixel font style generation

A.4 More qualitative and quantitative results of the proposed SKFont

We synthesized various font styles from the proposed SKFont method and evaluated the generated images from visual and performance metrics perspective. We generated various font styles from which 7 are visually displayed in Fig. 14.

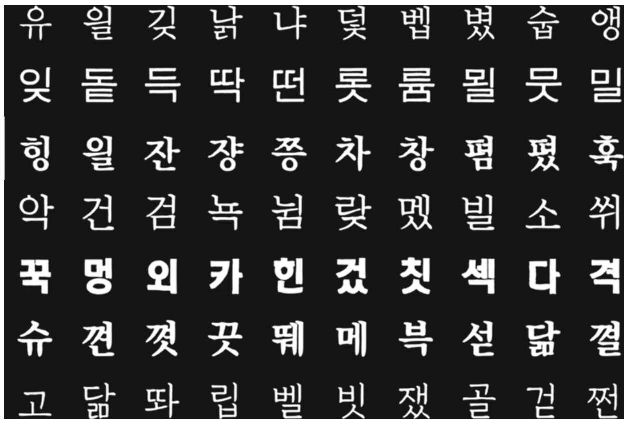


Fig. 14 Font characters generated in various font styles by SKFont

References

1. Isola, P., Zhu, J., Zhou, T., Efros. A.: Image-to-image translation with conditional adversarial networks. In: CVPR (2017)
2. Mirza, M., Osindero, S.: Conditional Generative Adversarial Nets (2014). arXiv preprint arXiv:1411.1784
3. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of IEEE, pp. 2278–2324 (1998)
4. Tian, Y.: Rewrite: neural style transfer for Chinese fonts (2016). https://github.com/kaonashityc/Rewrite
5. Gatys, L.: Image style transfer using convolutional neural networks. In: CVPR (2016)
6. Atarsaikhan, G., Iwana, B.K., Narusawa, A., Yanai, K., Uchida, S.: Neural font style transfer. In: Proceedings of the 14th International Conference 25 on Document Analysis and Recognition (ICDAR), vol. 5, pp. 51–56 (2017)
7. Hayashi, H., Abe, K., Uchida, S.: GlyphGAN: style-consistent font generation based on generative adversarial networks. In: Knowledge-Based Systems, vol. 186 (2019)
8. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks (2015). arXiv preprint arXiv:1511.06434
9. Zhu, J.-Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2017)
10. Tian, Y.: zi2zi: Master Chinese calligraphy with conditional adversarial networks (2017). https://github.com/kaonashi-tyc/zi2zi
11. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier GANs. In: Proceedings of the 34th International Conference on Machine Learning, pp. 2642–2651 (2017)
12. Taigman, Y., Polyak, A., Wolf, L.: Unsupervised cross-domain image generation. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings
13. Jiang, Y., Lian, Z., Jianguo, Y., Xiao, J.: DCFont: an end-to-end deep Chinese font generation system, SIGGRAPH Asia, p. 22, TB (2017)
14. Chang, B., Zhang, Q., Pan, S., Meng, L.: Generating handwritten Chinese characters using cyclegan (2018). CoRR. arXiv:1801.08624
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
16. Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely connected convolutional networks. In: CVPR (2017)
17. Sun, H.: Luo, Y., Ziang , L.: Unsupervised Typography Transfer (2018)
18. Guo, Y., Lian, Z., Tang, Y., Xiao, J.: Creating new Chinese fonts based on manifold learning and adversarial networks. In: Diamanti, O., Vaxman, A. (eds.) Proceedings of the Eurographics—Short Papers. The Eurographics Association (2018)
19. Jiang, Y., Lian, Z., Tang, Y., Xiao, J.: SCFont: Structureguided Chinese Font Generation via Deep Stacked Networks (2019)
20. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In: CVPR, vol. 2, no. 6 (2017)
21. Panichev, O., Voloshyna, A.: U-net based convolutional neural network for skeleton extraction. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) June (2019)
22. Zhang, T.Y., Suen, C.Y.: A fast parallel algorithm for thinning digital patterns. Commun. ACM 236–239 (1984)
23. Guo, Z., Hall, R.W.: Parallel thinning with two-subiteration algorithms. Commun. ACM **32**, 359–373 (1989)
24. Lee, T.C., Kashyap, R.L., Chu, C.N.: Building skeleton models via 3-D medial surface/axis thinning algorithms. Comput. Vis. Graph. Image Process. **56**, 462–478 (1994)
25. Lian, Z., et al.: EasyFont: a style learning-based system to easily build your large-scale handwriting fonts. ACM Trans. Graph. **38**, 61–618 (2019)
26. Tang, S., et al.: FontRNN: generating large-scale Chinese fonts via recurrent neural network. Comput. Graph. Forum **38**, 567–577 (2019)
27. Lopes, R.G., et al.: A learned representation for scalable vector graphics. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 7929–7938 (2019)
28. Sun, D. et al.: Learning to write stylized Chinese characters by reading a handful of examples. In: IJCAI (2018)
29. Baluja, S.: Learning typographic style (2016). arXiv:1603.04000
30. Azadi, S. et al.: Multi-content GAN for few-shot font style transfer. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7564–7573 (2018)
31. Gao, Y., et al.: Artistic glyph image synthesis via one-stage few-shot learning. ACM Trans. Graph. (TOG) **38**, 1–12 (2019)
32. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, Bi., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NIPS (2014)
33. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: MICCAI, pp. 234–241 (2015)
34. Kurach, K., Lucic, M., Zhai, X., Michalski, M.: The GAN landscape: losses, architectures, regularization, and normalization. In: International Conference on Learning Representations (2019)
35. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on Machine Learning, in PMLR, vol. 37, pp. 448–456 (2015)
36. Maas, A., Hannun, A., Ng, A.: Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of ICML, vol. 30 (2013)
37. Xu, B., Wang, N., Chen, T., Li, M.: Empirical evaluation of rectified activations in convolutional network (2015). arXiv preprint arXiv:1505.00853
38. Eck, P.: Handwritten Korean character recognition with tensorflow and android (2017). https://github.com/IBM/tensorflow-hangul-recognition

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.