J Inf Process Syst, Vol.17, No.1, pp.1~13, February 2021
https://doi.org/10.3745/JIPS.02.0152

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

SkelGAN: A Font Image Skeletonization Method

Debbie Honghee Ko*, Ammar Ul Hassan*, Saima Majeed*, and Jaeyoung Choi*

Abstract

In this research, we study the problem of font image skeletonization using an end-to-end deep adversarial network, in contrast with the state-of-the-art methods that use mathematical algorithms. Several studies have been concerned with skeletonization, but a few have utilized deep learning. Further, no study has considered generative models based on deep neural networks for font character skeletonization, which are more delicate than natural objects. In this work, we take a step closer to producing realistic synthesized skeletons of font characters. We consider using an end-to-end deep adversarial network, SkelGAN, for font-image skeletonization, in contrast with the state-of-the-art methods that use mathematical algorithms. The proposed skeleton generator is proved superior to all well-known mathematical skeletonization methods in terms of character structure, including delicate strokes, serifs, and even special styles. Experimental results also demonstrate the dominance of our method against the state-of-the-art supervised image-to-image translation method in font character skeletonization task.

Keywords

Generative Adversarial Network, Image-to-Image Translation, Skeletonization, Style Transfer

1. Introduction

Skeletonization (also termed as thinning) is a widely used technique for extracting the skeleton of an object by reducing its dimensionality. A skeleton is a compressed and simple, yet a highly effective representation of 2D (even 3D) objects. Skeletonization reduces a binary image to a one-pixel-width representation, and is important in image processing and computer vision, such as in face or fingerprint recognition [1], detection of specular tumors on mammograms or blood vessels [2], natural street-object recognition [3], text identification [4,5], and human action recognition [6].

Several skeletonization algorithms have been proposed. They are primarily categorized into two classes: iterative and non-iterative. In the former, skeletons are generated by examining and deleting the contour pixels in a repetitive manner either sequentially or in parallel, whereas in the latter, skeletons are obtained in a non-repetitive manner, that is, not all pixels are examined. Some of the most commonly used skeleton extraction techniques include morphological thinning [7], geometric methods based on Voronoi diagrams [8], and methods based on distance transformations, in which the goal is to extract the skeleton using a non-pixel-based approach in order to reduce the time complexity [9]. A detailed survey on skeletonization methods is provided in [10].

Manuscript received July 14, 2020; first revision September 4, 2020; second revision October 21, 2020; accepted November 8, 2020.

Corresponding Author: Jaeyoung Choi (choi@ssu.ac.kr)

* School of Computer Science and Engineering, Soongsil University, Seoul, Korea (debbie.pust@gmail.com, ammar.instantsoft@gmail.com, saimamajeed089@gmail.com, choi@ssu.ac.kr)

Font character skeletonization is also a challenging problem where each character has its own delicate structure and style. Furthermore, the Chinese, Japanese, and Korean (CJK) based languages which consist of a large number of characters with complex structures and shapes make this skeletonization task even more challenging. Most of these mathematical methods [11-13] can be used for skeleton generation of text-images, however, these approaches may result in disjointedness. Furthermore, they may attach complicated elements and generate serious artifacts as well as non-realistic images. Even though some of these issues are acceptable, the handling of delicate aspects, such as special stokes/slant/serif details, is an issue and requires post-processing techniques for pruning these artifacts [10,14].

Recently, deep neural networks have been highly successful in various computer vision tasks, such as image classification, image segmentation, object detection, and image synthesis. Generative models based on convolutional neural networks, such as autoencoders [15], variational autoencoders [16], and generative adversarial networks (GANs) [17], are widely used for image synthesis. All these methods primarily represent data in the latent feature space, from which images are synthesized using a decoder. GANs additionally incorporate an adversarial network that facilitates the synthesis of high-quality images using a minimax game formulation. This ability of generative models to represent the data in the latent feature space could be exploited for font character skeletonization without any post-processing techniques in an end-to-end manner.

In this paper, we propose a skeleton generator, SkelGAN, for synthesizing font-character skeletons. The method is primarily targeted at Korean Hangul characters, but it can also be adopted for alphabets or Chinese characters as shown in Section 4.5.2. We demonstrate that the proposed deep-learning method is more effective than existing mathematical methods in generating font skeletons. The proposed SkleGAN is constructed using deep neural networks in an end-to-end manner. The generated font-character skeleton comprises of a one-pixel-width structure. Moreover, they are more robust, and have better style as well as a more consistent structure than those obtained by current mathematical skeletonization methods. We also demonstrate how our SkelGAN can synthesize more realistic font character skeletons than the state-of-the-art supervised image-to-image (I2I) translation method [18], when applied for font character skeletonization task.

2. Related Work

Skeletonization methods can be primarily divided into two categories: mathematical and neural network approaches.

2.1 Mathematical Approaches

Over the years several algorithms have been proposed for extracting the skeleton of an image object. We present the three most widely used skeletonization algorithms. The fast-parallel algorithm for skeletonization [11] is the most widely used skeletonization algorithm owing to its robustness. It performs continuous passes on the input image. In each pass, it removes pixels on the object borders (object in the input image). This process continues until no more pixels can be removed.

An algorithm based on morphological thinning was proposed in [12]. It is based on the same principle as that of the algorithm in [11], that is, it removes the pixels from the borders in each iteration until no pixel

can be removed. However, it uses different rules for removing pixels to enhance the skeletonization process.

The medial axis thinning algorithm [13] uses an octree data structure to examine a 3×3×3 neighborhood of a pixel. An iterative convolution is performed over the input image, and the pixels at each iteration are removed until the input image cannot be further altered. Specifically, a group of pixels for removal are selected and the resulting groups are sequentially reexamined to preserve image connectivity.

Further, there are various methods [10,19,20] that perform skeletonization from the object segmentation. Usually, these methods are relatively sensitive to the artifacts of the given object shape. Some methods [21-23] exploit gradient intensity maps for generating skeletons of objects like sky, sea, etc. which usually require a prior object to be stifled. However, all existing approaches are sensitive to noise. They often add pixels on the boundary of an object skeleton, and thus post-processing algorithms such as redundant branch pruning [14] are required to remove noise and preserve connectivity. More specifically, when these approaches are used for font-character skeletonization, which is more delicate than natural-object skeletonization, they result in disjointedness; furthermore, they attach complicated elements and produce serious artifacts as well as non-realistic skeletons. Some common problems of these approaches are shown in Fig. 1.

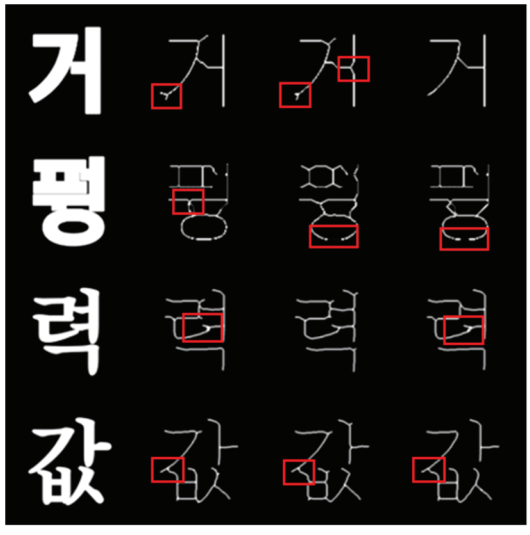

Fig. 1. Problems of existing skeletonization approaches. By column order: ground truth font, skeletons generated by [11], [12], and [13] methods, respectively.

2.2 Neural Network Approaches

As deep learning technology advances, various methods have been proposed to extract skeletons using neural networks. In some studies, the skeletonization problem has been regarded as a pixel-by-pixel classification problem [24] which can be addressed by semantic segmentation.

DeepSkeleton [25] accomplished skeletonization from natural images using fully convolutional neural networks [26]. Skeletonization has also been regarded as an I2I translation problem [27,28], where the vanilla U-Net architecture [18] was modified for extracting skeletons from binary images. PSPU-SkelNet [29] utilized three U-Nets for extracting the skeletons from a given shape point cloud. Some architectures integrate convolutional neural network (CNN) and long short-term memory (LSTM) to achieve high

performance in terms of visual recognition in tasks such as Chinese character recognition [30].

The aforementioned studies primarily focused on generating skeletons from natural images. However, none of the skeletonization approaches using deep learning have been concerned with font images, which are more delicate and stylish than natural images. The proposed skeleton generator extracts skeletons from font characters, primarily from Korean Hangul characters, which contain stylish strokes, serifs, and radicals. We consider this font character skeletonization (font-to-skeleton) to be an I2I translation problem, where the goal is to learn the domain transfer function from a font image to its corresponding skeleton. In the next section, we will define the proposed network architecture and loss functions.

3. Network Architecture

Recently, the U-Net architecture has been widely used in various I2I tasks, such as semantic segmentation, background masking, season transfer, and object transfiguration. The vanilla U-Net [18] consists of an encoder–decoder generator with skip connections, and a PatchGAN discriminator.

3.1 Modification of U-Net Architecture

The vanilla U-Net architecture was designed for one-to-one mapping problems, whereas font-to-skeleton is a one-to-many mapping problem, in which a single font character in the reference domain can have several different skeleton styles in the target domain. Accordingly, we modified the vanilla U-Net architecture by concatenating a style vector V^s and character class vector V^c to the encoded z vector in the latent feature space. The style vector guides our network to generate the skeleton in various styles. On the other hand, the character class vector is associated with the character content. These style and character vectors in our network are one-hot encoded corresponding to the style and character class. The size of V^s and V^c are the total number of styles and characters used for learning. Our decoder thus takes an input latent vector which consists of z, V^s, and V^c instead of only encoded z vector (vanilla U-Net decoder).

Our discriminator is trained by solving multiple adversarial classification tasks simultaneously. We modified the original PatchGAN discriminator [18], which outputs a multi-dimensional vector, where each point of the vector corresponds to a N×N patch of the input image. The discriminator determines whether the given patch is real or fake. Moreover, we added two fully connected classification layers at the end of the discriminator. The first is a style classification (SC) layer (Fig. 2) and is trained to predict the style of the generated skeleton, whereas the second is a character classification (CC) layer (Fig. 2) and is trained to predict the character label of the generated skeleton (more details in Section 3.2). The SkelGAN architecture is shown in Fig. 2.

3.2 Learning Objectives

The proposed SkelGAN utilizes four learning objectives: an adversarial loss, a style classification loss, a character classification loss, and a L1 loss. Each one of these losses aim at imposing various properties of the synthesized skeleton image y^i.

Our discriminator estimates the probability that the samples come from a real distribution, i.e., training data X, or belong to the artificially synthesized distribution X^i. This setting of generator and discriminator corresponds to a min max optimization problem. This formulation helps in improving the visual

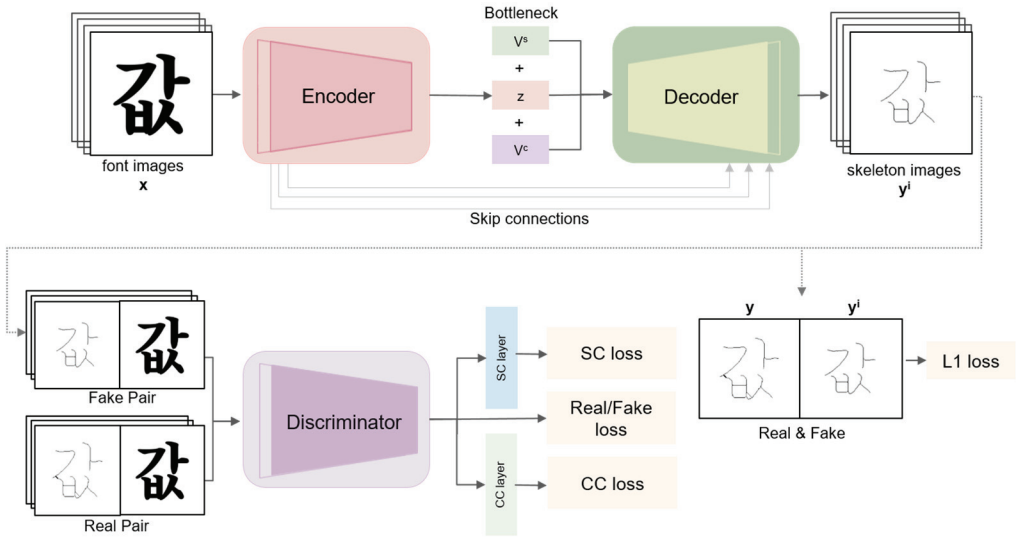

Fig. 2. Our proposed skeleton generator with modified U-Net architecture. The input to this network is a reference font image, x, which is downsampled via an encoder to extract the high-level features *z*. Here, we combine these *z* features with a style vector V^s and a character class vector V^c. This latent is then passed through a series of upsampling layers to generate the target skeleton image y^i.

appearance of the generated images (look realistic). However, it does not take into consideration neither the style nor contents of the generated images. More explicitly, we use the non-saturated GAN loss for both the generator and discriminator. This loss is more stable and converges quickly, compared to the saturated loss [31] used in the original U-Net architecture. This loss is formally defined as follows:

$$Loss_{cGANns}(D,G) = E_y\,[Log\ D(y)] + E_x[Log(1 - D(G(x)))], \tag{1}$$

where *x* given to the generator is the input font image and it generates the corresponding skeleton image y_i (fake skeleton image) close to the ground truth skeleton y image such that the discriminator cannot distinguish between the real and fake skeletons.

To handle the one-to-many mapping problem, where a single font character in the reference domain may correspond to various skeleton styles in the target domain, we employed the style classification loss ($Loss_{sc}$). This loss not only provides diversity on the generated skeletons, but also prevents the mode collapse problem. The proposed style classification loss guides our generator to synthesize skeletons conditioned to a particular font style by means of an input font image *x*. In our discriminator, we add a style classification dense layer with the amount of styles in our training dataset. Hence the discriminator job is not only to tell either the image is real or fake, but it also checks whether the generated skeleton is in the same style as the target domains skeleton style. We utilize the cross-entropy loss, formally defined as

$$Loss_{sc} = -E_{x \sim \{X, X^i\}} \left(\sum_{i=1}^{N} s_i . \mathrm{Log}(\hat{s}_i)\right), \tag{2}$$

where $\hat{s}_i$ is the predicted probability distribution over the styles and s_i is the real style distribution. Synthesized skeletons should be classified as the style s_i used to construct the input style conditioning image *x*. The classifier is optimized with both real and fake samples drawn from *X*, X_i distributions, respectively.

We also experimentally demonstrate that, sometimes, the generator synthesizes skeleton characters that do not semantically map to the actual target character (in terms of content). For example, "힁" and "잏" are very close but semantically totally different Korean Hangul characters (demonstrated in Section 4.5, second last row). To generate the accurate skeleton content, it is beneficial to use the character classification loss ($Loss_{cc}$) in the discriminator. This loss guides the generator to generate skeletons with the actual content. $Loss_{cc}$ is trained using the character classification dense layer added at the end of discriminator. $Loss_{cc}$ is given by:

$$Loss_{cc} = -E_{x \sim \{X\}} \left(\sum_{i=1}^{N} c_i . \mathrm{Log}(\hat{c}_i)\right), \tag{3}$$

where $\hat{c}_i$ is the predicted probability distribution over the character labels and c_i is the real character distribution. Synthesized skeletons should be classified as the character c_i. This classifier is only optimized with real samples drawn from X distribution.

SkelGAN is trained with a paired dataset, i.e., every font image in the reference domain has its ground truth skeleton in the target domain. Taking advantage of this supervised setting, the L1 loss ($Loss_{L1}$ which is less blurry than $Loss_{L2}$) is also used to generate skeleton images that have the overall structure as those in the target domain:

$$Loss_{L1} = E_{X,x^i} [\|y - G(x)\|]. \tag{4}$$

The final objective is:

$$G^* = argmin_G max_D Loss_{cGANns}(G, D) + Loss_{sc}(G, D) + Loss_{cc}(G, D) + \lambda Loss_{L1}. \tag{5}$$

4. Experimental Results

4.1 Training and Testing Datasets

There is no publicly available dataset of font characters and their corresponding ground truth skeletons; therefore, we constructed a custom dataset to train and evaluate the proposed model. First, we selected 30 different font files based on diverse styles and then extracted the 2,350 most commonly used Korean Hangul characters from each file (2,350×30 training images). Subsequently, we generated the corresponding skeletons for these font characters using three mathematical approaches, as discussed in Section 2.1. Then, by visually analyzing the results, we selected the method in [13] for training, as this method generated reasonable font skeletons compared with the other approaches. Both the input font image and the target skeleton image in the model are RGB with a size of 256×256×3 (three channels for RGB).

For testing, we used various unseen Korean font styles that are selected based on the overall appearance and structure of the characters in the reference and target domains. The aim is to evaluate the ability of the model to generate diverse skeletons in different styles.

4.2 Network Details and Parameter Settings

For our experiments, the input and output character images are both 256×256×3 (RGB). For our generator, the encoder contains seven down-sampling layers. Every layer in the encoder consists of a convolution operation followed by instance normalization and leaky ReLU activation function, except

the first layer where instance 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 up-sampling layers. Each layer has a deconvolution operation followed by instance normalization and ReLU activation function. As an exception to the above operation flow, instance normalization is not applied to the last layer of the decoder, and the Tanh activation function is used instead of ReLU.

4.3 Qualitative Evaluation

For a qualitative evaluation of the generated skeletons, we visualized the skeleton images using the proposed model and the baseline mathematical approach [13]. We used the python scikit-image library for generating baseline method skeletons. As shown in Fig. 3, the proposed skeleton generator is superior and overcomes certain typical issues, such as noise on the border, inconsistent shapes, and unnecessary pixels.

(a) (b) (c) (a) (b) (c)

Fig. 3. Qualitative comparison between Lee's mathematical baseline approach and the proposed skeleton generator: (a) original font image, (b) skeletons by [13], and (c) skeletons by SkelGAN.

4.4 Quantitative Evaluation

As perfect ground-truth skeletons are not available, commonly used quantitative metrics in many image generation tasks cannot be computed. Therefore, we trained the SkleGAN to learn the inverse mapping function, that is, skeleton-to-font. First, we trained SkelGAN to produce fonts using skeletons by the baseline method.

Subsequently, we used skeletons obtained from the SkelGAN to synthesize the corresponding font images (with the same network settings). As shown in Fig. 4, the fonts generated by the proposed method are more realistic and non-blurry; furthermore, they have a consistent style and exhibit superior quality over the fonts synthesized by the baseline method.

Thereby, perfect ground-truth font images are obtained, and thus we can compute the mean SSIM (structural similarity index measure) and mean L1 distance of the fonts generated using the skeletons by the proposed network and the fonts by the baseline skeletons. As shown in Table 1, the proposed skeleton generator outperforms Lee's baseline method [13] in terms of both the L1 distance and SSIM for all three unseen styles.

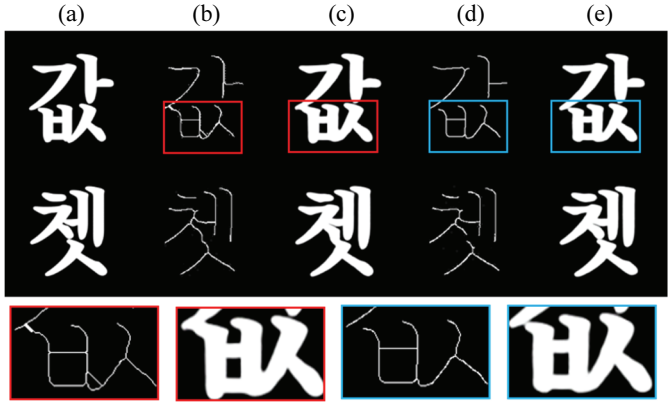

Fig. 4. Examples of skeleton to font generation: (a) ground truth fonts, (b) skeletons by [13], (c) generated fonts using skeleton image of (b), (d) skeletons by SkelGAN, and (e) generated fonts using skeleton image of (d).

Table 1. Quantitative comparison between Lee's method [13] and SkelGAN

Font style	Lee's method [13]		Proposed SkelGAN	
	L1 loss	SSIM	L1 loss	SSIM
KoPub WorldBatang Bold	0.3191	0.9504	0.2723	0.9721
Typo seemyoungjo	0.2912	0.9087	0.2714	0.9446
DXbomgyeolExB	0.3222	0.9221	0.2155	0.9374

4.5 Comparison against the State-of-the-art

According to our knowledge, there is no method based on generative models which synthesize skeletons of font character images. Pix2pix [18] is a state-of-the-art method for I2I translation and it has shown impressive results in many I2I tasks like season transfer, edge-to-shoes, day-to-night image, etc. Pix2pix works on paired datasets, i.e., every reference image must have its corresponding image in the target domain sharing similar structural features. Hence, pix2pix is the most relevant method which can be compared to our SkelGAN. For a fair comparison, we pretrained pix2pix method using the same dataset, and finetuned them with the same unseen styles. Pix2pix is trained using the implementations provided by the authors.

As shown in Fig. 5, our method produces finer skeletons than the baseline methods. Although pix2pix can generate reasonable results whose general shape and styles represent the input font images. However, it also contains some artifacts, such as edge noise and broken strokes.

4.6 Generalization Ability

To test the generalization capability of the proposed SkelGAN, we performed the following few experiments.

4.6.1 Synthesizing unseen glyph images

This experiment is conducted to show that other than the glyph images seen in training, our model also has the ability to synthesize skeletons for unseen characters, which are never seen by our model during

both pretraining and finetuning. For this experiment, we choose 114 sample hangul characters that represent the overall structure of all the other Korean Hangul characters. We then finetune our network with these 114 characters. This finetuning process helps the network to learn the new skeleton style with its specific style embedding. By doing this the pretrained model converges fast compared to the one trained from scratch. After learning the new skeleton embedding during finetuning phase, SkelGAN generates the rest of 2,236 skeletons in the newly learned style. Fig. 6 demonstrates a few unseen samples of the generated skeletons from SkelGAN.

(a) (b) (c) (d)

Fig. 5. Comparison against baseline mathematical and deep learning methods: (a) ground truth fonts, (b) skeletons by [13], (c) skeletons by [18], and (d) skeletons by SkelGAN.

(a) (b) (c)

Fig. 6. Unseen characters rendered by SkleGAN: (a) ground truth unseen fonts, (b) skeletons by [13], and (c) skeleton by SkelGAN.

4.6.2 Cross language evaluation

We perform this cross-language evaluation to check the ability of SkleGAN on languages it has never seen during pre-training and finetuning. For this experiment, we train our model with Korean hangul characters with various styles. Then we feed unseen Chinese and English characters as the reference input. As depicted in Fig. 7, our model can synthesize English alphabets and Chinese unseen language characters, respectively, in acceptable quality. These results demonstrate the powerful generalization ability of our model even for the cross-domain languages. Fig. 7 also demonstrates how the deep learning baseline pix2pix method fails to generalize the cross-language synthesis task.

(a) (b) (c) (a) (b) (c)

Fig. 7. Examples of synthetic alphabets and Chinese characters, respectively: (a) ground truth fonts, (b) skeletons by [18], and (c) skeletons by SkelGAN.

5. Conclusion

In this paper, we proposed a highly effective skeletonization network, SkelGAN, that is based on a modified U-Net architecture, which can handle font character skeletonization problem. State-of-the-art skeletonization methods based on mathematical algorithms produce blurry, broken, and non-realistic skeletons of font images. SkelGAN is based on an end-to-end generative adversarial network architecture.

We further propose a modified encoder–decoder and PatchGAN based architecture to handle our font

skeletonization problem. We demonstrate through qualitative and quantitative experiments that the proposed SkelGAN method outperforms existing baseline mathematical and deep learning methods. The generalization capability of the proposed SkelGAN shows that it can be used for skeletonization on unseen font characters and cross-languages as depicted in our experiments. In our future work, we will focus on more challenging artistic font styles as currently SkelGAN doesn't perform well on these challenging cursive font styles.

Acknowledgement

This work was supported by Institute of Information & communications Technology Planning and Evaluation (IITP) grant funded by the Korea government (MSIP) (No. 2016-0-00166, Technology Development Project for Information, Communication, and Broadcast).

References

[1] F. Zhao and X. Tang, "Preprocessing and postprocessing for skeleton-based fingerprint minutiae extraction," *Pattern Recognition*, vol. 40, no. 4, pp. 1270-1281, 2007.

[2] M. P. Sampat and A. C. Bovik, "Detection of spiculated lesions in mammograms," in *Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE Cat. No. 03CH37439)*, Cancun, Mexico, 2003, pp. 810-813.

[3] W. Shen, K. Zhao, J. Yuan, Y. Wang, Z. Zhang, and X. Bai, "Skeletonization in natural images and its application to object recognition," in *Skeletonization: Theory, Methods and Applications*. St. Louis, MO: Academic Press, 2017, pp. 259-285.

[4] N. Li, "An implementation of OCR system based on skeleton matching," Computing Laboratory, University of Kent, Canterbury, UK, 1993.

[5] X. Bai, L. Ye, J. Zhu, L. Zhu, and T. Komura, "Skeleton filter: A self-symmetric filter for skeletonization in noisy text images," *IEEE Transactions on Image Processing*, vol. 29, pp. 1815-1826, 2019.

[6] R. Vemulapalli, F. Arrate, and R. Chellappa, "Human action recognition by representing 3D skeletons as points in a lie group," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, 2014, pp. 588-595.

[7] C. Shu and Y. Mo, "Morphological thinning based on image's edges," in *Proceedings of the International Conference on Communication Technology (IEEE Cat. No. 98EX243)*, Beijing, China, 1998.

[8] J. W. Brandt and V. R. Algazi, "Continuous skeleton computation by Voronoi diagram," *CVGIP: Image Understanding*, vol. 55, no. 3, pp. 329-338, 1992.

[9] G. Borgefors, "Distance transformations in digital images," *Computer Vision, Graphics, and Image Processing*, vol. 34, no. 3, pp. 344-371, 1986.

[10] P. K. Saha, G. Borgefors, and G. S. di Baja, "A survey on skeletonization algorithms and their applications," *Pattern Recognition Letters*, vol. 76, pp. 3-12, 2016.

[11] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Communications of the ACM*, vol. 27, no. 3, pp. 236-239, 1984.

[12] Z. Guo and R. W. Hall, "Parallel thinning with two-subiteration algorithms," *Communications of the ACM*, vol. 32, no. 3, pp. 359-373, 1989.

[13] T. C. Lee, R. L. Kashyap, and C. N. Chu, "Building skeleton models via 3-D medial surface axis thinning algorithms," *CVGIP: Graphical Models and Image Processing*, vol. 56, no. 6, pp. 462-478, 1994.

[14] S. Wei, B. Xiang, Y. X. Wei, and L. L. Jan, "Skeleton pruning as trade-off between skeleton simplicity and reconstruction error," *Science China Information Sciences*, vol. 53, pp. 1-14, 2013.

[15] P. Vincent, H. Larochelle, Y. Bengio, and P. A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008, pp. 1096-1103.

[16] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2014 [Online]. Available: https://arxiv.org/abs/1312.6114.

[17] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Advances in Neural Information Processing System*, vol. 27, pp. 2672-2680, 2014.

[18] P. Isola, J. Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, 2017, pp. 5967-5976.

[19] W. P. Choi, K. M. Lam, and W. C. Siu, "Extraction of the Euclidean skeleton based on a connectivity criterion," *Pattern Recognition*, vol. 36, no. 3, pp. 721-729, 2003.

[20] W. Shen, X. Bai, R. Hu, H. Wang, and L. J. Latecki, "Skeleton growing and pruning with bending potential ratio," *Pattern Recognition*, vol. 44, no. 2, pp. 196-209, 2011.

[21] Z. Yu and C. Bajaj, "A segmentation-free approach for skeletonization of gray-scale images via anisotropic vector diffusion," *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Washington, DC, 2004, pp. 415-420.

[22] Q. Zhang and I. Couloigner, "Accurate centerline detection and line width estimation of thick lines using the radon transform," *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 310-316, 2007.

[23] T. Lindeberg, "Edge detection and ridge detection with automatic scale selection," *International Journal of Computer Vision*, vol. 30, no. 2, pp. 117-156, 1998.

[24] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proceedings of the IEEE International Conference on Computer Vision*, Santiago, Chile, 2015, pp. 1395-1403.

[25] S. Chen, X. Tan, B. Wang, and X. Hu, "DeepSkeleton: learning multi-task scale-associated deep side outputs for object skeleton extraction in natural images," *IEEE Transactions on Image Processing*, vol. 26, no. 11, pp. 5298-5311, 2017.

[26] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, 2015, pp. 3431-3440.

[27] O. Panichev and A. Voloshyna, "U-Net based convolutional neural network for skeleton extraction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, Long Beach, CA, 2019, pp. 1186-1189.

[28] S. Nathan and P. Kansal, "Skeletonnet: shape pixel to skeleton pixel," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, Long Beach, CA, 2019, pp. 1181-1185.

[29] R. Atienza, "Pyramid U-network for skeleton extraction from shape points," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, Long Beach, CA, 2019, pp. 1177-1180.

[30] W. Tang, Y. Su, X Li, D. Zha, W. Jiang, N. Gao, and Ji Xiang, "CNN-based chinese character recognition with skeleton feature," in *Neural Information Processing*. Cham, Switzerland: Springer, 2018, pp. 461-472.

[31] K. Kurach, M. Lucic, X. Zhai, M. Michalski, and S. Gelly, "The GAN landscape: losses, architectures, regularization, and normalization," 2018 [Online]. Available: https://openreview.net/forum?id=rkGG6s0qKQ.

Debbie Honghee Ko **https://orcid.org/0000-0003-2780-5178**

She received B.S. degree in Department of Physics from Korea University in 1983, Seoul, South Korea. She then received her M.S. degree in Department of Computer Science from Florida State University, USA, in 1989. She worked at the Fl Department of Education for 10 years as a supervisor. She is currently taking her Ph.D. degree in Department of Computer Science from Soongsil University, Seoul, South Korea, under the supervision of professor, Jaeyoung Choi. Her current research area is deep learning on font synthesis using generative models.

Ammar Ul Hassan **https://orcid.org/0000-0001-6744-507X**

He received B.S. degree in Department of Software Engineering from International Islamic University Islamabad, Pakistan, in 2013. He then received his M.S. degree in Computer Science from Soongsil University, Seoul, South Korea in 2018. He is currently taking his Ph.D. degree in Department of computer science from Soongsil University Seoul, South Korea. He is working as a research associate in System Software laboratory under the supervision of professor, Jaeyoung Choi. His current research interests are deep learning, computer vision, generative models, making font environment for new fonts in Linux operating system.

Saima Majeed **https://orcid.org/0000-0003-0334-694X**

She received her B.S.C.S. degree in Department of Computer Science from Capital University of Science and Technology, Islamabad, Pakistan 2017. She is currently taking her Master's degree in Department of Computer Science from Soongsil University Seoul, South Korea. She is working as a research assistant in System Software laboratory under the supervision of professor, Jaeyoung Choi. Her current research area is about operating system font configuration, deep learning, image processing.

Jaeyoung Choi **https://orcid.org/0000-0002-7321-9682**

He received the B.S. degree in Department of Control and Instrumentational Engineering from Seoul National University, Seoul, Korea, in 1984, the M.S. degree in Department of Electrical Engineering, University of Southern California in 1986, and the Ph.D. degree in School of Electrical Engineering from Cornell University, in 1991. He has previously worked at Oak Ridge National Laboratory (1992–1994) and University of Tennessee, Knoxville (1994–1995) as a postdoctoral research associate and a research assistant professor, respectively, where he had been involved with the ScaLAPACK project. He is currently a professor of School of Computer Science and Engineering at Soongsil University, Seoul, Korea. His current research interests include high performance computing and typography.