Knowledge-Based Systems 229 (2021) 107304

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Unpaired font family synthesis using conditional generative adversarial networks

Ammar Ul Hassan[1], Hammad Ahmed[1], Jaeyoung Choi *

Department of Computer Science and Engineering, Soongsil University, Seoul, South Korea

ARTICLE INFO

Article history:
Received 11 January 2021
Received in revised form 26 May 2021
Accepted 11 July 2021
Available online 21 July 2021

Keywords:
Font generation
Generative adversarial networks
Style transfer
Unsupervised image-to-image translation

ABSTRACT

Automatic font image synthesis has been an extremely active topic in recent years. Various deep learning-based approaches have been proposed to tackle this font synthesis task by considering it as an image-to-image translation problem in a supervised setting. However, all such approaches mainly focus on one-to-one font mapping, i.e., synthesizing a single font style, making it difficult to handle more practical problems such as the font family synthesis, which is a one-to-many mapping problem. Moreover, this font family synthesis is more challenging because it is an unsupervised image-to-image translation problem, i.e., no paired dataset is available during training. To address this font family synthesis problem, we propose a method that utilizes a single generator to conditionally produce various font family styles to form a font family. To the best of our knowledge, our proposed method is the first to synthesize a font family (multiple font styles belonging to a font), instead of synthesizing a single font style. More specifically, our method is trained to learn a font family by conditioning on various styles, e.g., normal, bold, italic, bold-italic, etc. After training, given an unobserved single font style (normal style font as an input), our method can successfully synthesize the remaining styles (e.g., bold, italic, bold-italic, etc.) to complete the font family. Qualitative and quantitative experiments were conducted to demonstrate the effectiveness of our proposed method.

1. Introduction

Font designers refer to typography as the style and appearance of the printed text. To specialize in typography, creativity and font expertise are required which are not generally found in common users. In addition, font design is a time-consuming task. It takes from several weeks up to months to design a single font for Latin alphabet characters. This time complexity increases massively for some languages such as Chinese and Korean which consists of a large number of characters (up to 50,000 characters for Chinese and 11,172 characters for Korean). Moreover, the font designer must design separate font styles when creating a single font family.

Existing state-of-the-art methods in font synthesis mainly consider this to be an image-to-image translation problem in a supervised setting (paired dataset). All recent methods learn a one-to-one mapping function from a source font to a target font conditioned on a single font style, where the font style is fused in the form of a style label [1–4] or is disentangled using a separate style encoder [5–8]. Although these supervised image-to-image translation font synthesis methods are successful in style transfer between the source and target fonts, the generalization capability when synthesize a complete font family (consisting of multiple fonts of that font family) from a single source font at the inference time is beyond their reach. Specifically, a paired training dataset is needed to learn this one-to-one mapping function conditioned on a single target style, whereas font family synthesis is a one-to-many mapping task in an unsupervised setting (unpaired training examples) and it may be impractical or even impossible to acquire a large number of font family paired datasets, making this a more challenging problem.

This study is an attempt to bridge the gap between the font designer and machine design capability by proposing an unpaired font-family conditional generative adversarial network (UFFG), which aims to synthesize a complete font family given a single font as an input. To the best of our knowledge, the proposed UFFG is the first method that automatically synthesizes a font family using generative adversarial networks (GANs) [9]. We denote the terms font family (FF) as a set of font styles (FS) referring to a specific style of FF in which each FS is represented by its own font style label (FSL). For example, Arial is a FF consisting of various FS's such as Arial Black and Arial-thin, each represented

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail address: choi@ssu.ac.kr (J. Choi).

[1] Equal contribution.

https://doi.org/10.1016/j.knosys.2021.107304
0950-7051/

by its own FSL. Based on this, we train the UFFG to synthesize a complete FF conditioned on a single FS image and the target FSL as an input, where each FF may or may not contain all of the corresponding FSs thus making it an unsupervised image-to-image translation problem. Please note that the term "unsupervised" refers exclusively to unpaired image-to-image translation, despite the fact that we require FSLs as input and as the basis for our font style classification loss function discussed in Section 3.3.

Specifically, we train our model to learn the global features (thickness, thinness and slant) of all FSs by learning the FSLs. By contrast, the local FS details (strokes and shapes) are learned using a conditional input image. We show that by learning the global features of each FS using its corresponding FSL, the model learns generalized FS features that can be applied to an unseen font (an unseen font refers to a FF that was never seen during the training process) or an unobserved FS (an unobserved FS is a FS, that was unavailable during training, whereas the other FSs of that FF may have been available) at the inference stage for synthesizing its corresponding FF. We conducted qualitative and quantitative experiments on the proposed dataset, including comparison with the baseline methods to validate our proposed network. We also conducted extensive ablation studies to choose the architecture of the proposed network. Finally, our cross-language evaluation demonstrates that our method can be applied to any writing system (e.g., Korean, Chinese) that contains a large number of font characters but not many font families.

The main contributions and features of the proposed method are as follows:

- This paper provides the first automatic font family synthesis method, which unlike existing methods focuses on complete font family synthesis during the inference stage.
- This paper proposes a new font family dataset for Latin characters that consists of 118,456 font images in eight FSs which we refer to as a FF herein.
- The proposed UFFG can generate a style and character-consistent FF given a single FS image as an input during the inference stage
- In our experiments, we demonstrate the superiority of the proposed method to the baselines not only on a font family dataset but also on non-glyph datasets such as CelebA [10] as demonstrated in the experiments section.

2. Related work

2.1. Image synthesis based on GAN's

GANs [9] are a class of generative models that are widely used for image synthesis tasks. GANs incorporate an adversarial network that facilitates the synthesis of high-quality images using a minimax game formulation. Since the invention of the vanilla GAN, many variants have been proposed to improve the quality and training stability of GANs [11–14].

Owing to recent advancements, many state-of-the-art methods in various computer vision applications utilize GANs, e.g., image-to-image translation [15], semantic-image-to-photo translation [16], text-to-image translation [17], face synthesis [18], image in-painting [19], and super resolution [20].

A supervised image-to-image translation (I2I) framework [15] aims to learn a conditional image synthesis function to map a source domain image to a corresponding target domain image given a paired dataset during training. By contrast, unsupervised image-to-image translation methods [21–23] aim to learn a conditional image synthesis function to map an source domain image to a target domain image without a paired dataset. These supervised or unsupervised approaches have shown great success in uni-domain I2I tasks; however, they only consider a mapping between two domains (one-to-one).

To address this domain scalability problem (also known as multi-domain I2I), several studies have proposed a unified architecture [24,25]. Such methods are based on a single generator to learn multiple domains (one-to-many) unlike previous methods that require m(m-1) generators for learning $m_{1,2,...}$ number of domains. Similarly, some studies have focused on synthesizing images considering diverse styles also known as multi-modality [26–28]. Several recent studies have also applied few-shot image translation techniques [29,30].

2.2. Font synthesis based on GAN's

Most font generation studies consider the font synthesis task as an I2I task, where the goal is to transfer from an input font style to a target font style conditioned on the target font style information. The style information is either injected by the style label or is disentangled with the help of a separate style encoder. We discuss below some recent studies that follow these approaches.

2.2.1. Style label guided font synthesis

Zi2zi [1] proposed the first method based on I2I for Chinese font synthesis. Zi2zi was based on pix2pix [15], AC-GAN [31], and a domain transfer network [32]. A category label representing the target font category (style) is concatenated with the character vector from the encoder that guides the decoder to generate the target style. The approach in [2] was built on the same idea for generating Chinese handwriting; however, the authors used an additional font feature reconstruction network to provide a style vector in the latent along with a character vector and category embedding.

In [3], the authors proposed a skeleton-guided font generation method for Chinese font synthesis. They created a stack architecture for learning the skeletons of the target font followed by two I2I methods based on category embedding to generate smooth Chinese characters. Chang et al. [4] utilized cycleGAN to synthesize Chinese characters in personalized handwriting. Instead of using the default ResNet blocks [33] as used in the original cycleGAN, they utilized DenseNet [34] in their architecture. To generate Korean hangul characters, Ko et al. [35] proposed a three-stage generator architecture guided by skeletons. Their proposed network generates 2,350 of the most frequently used hangul characters using the 114 basis characters.

2.2.2. Style disentangle guided font synthesis

Recent font synthesis methods attempt to disentangle the style and content representations by splitting the style and content encoders in this I2I translation setup. The content encoder extracts the character content information; by contrast, the style encoder extracts style information such as the strokes and thickness. Azadi et al. [5] proposed a two-stage network architecture for synthesizing stylish alphabet glyph characters. The first stage network learns the target shape from a few characters, and the second stage network style transfers the target texture on the learned shape from the first stage network. Gao et al. [6] improved the approach in [5] by proposing a top-down generator architecture along with a local texture discriminator and texture refinement loss for stylish alphabet and Chinese character synthesis.

Cha et al. 2020 [7] proposed a component-guided font synthesis method restricted to complete compositional scripts, such as Korean and Thai handwritings. This method cannot be applied to other complex writing systems such as Chinese, which consists of complex glyph structures and diverse local styles. Inspired by low-rank matrix factorization, LF-Font [8] was recently proposed, which learns to disentangle the glyph structure and local style representations to generate complex Chinese handwriting.

2.2.3. Additional image and font synthesis studies

Lopes et al. [36] proposed a sequential generative model based on variational auto encoders [37] for disentangling the style as well as a stacked LSTM [38] and an MDN [39] guided SVG decoder. GlyphGAN [40] modified the DCGAN architecture to generate font styles in a multimodal setting by fusing the style information as a random sample from a Gaussian distribution to the generator. FontRNN [41] focuses on tracing writing trajectories by utilizing a monotonic attention mechanism to synthesize Chinese handwritings. In [42], the authors proposed an image-to-image translation method for generating skeletons of font images. They demonstrated in their paper that the synthesized skeletons from their model are of high quality than the state-of-the-art mathematical methods of skeletonization.

Some studies [43,44] have focused on a controllable image synthesis by controlling the image attributes, e.g., controlling the output image attributes such as hair color and age for faces images. Attribute2Font [45] was recently proposed to allow user-controllable font synthesis. They embedded an attribute attention module to improve the model performance.

All of the above-mentioned font synthesis studies achieve good results in one-to-one font synthesis tasks, i.e., given a content font input, the goal is to learn an I2I function that transfers the style to a single target font conditioned on the style information. The style information in these studies represents a single target font style, which limits their usage for many complex problems such as the font family synthesis. In addition, these methods require a paired training dataset during training, which makes their usage impossible for the font family synthesis problem where paired dataset is unavailable. To address this font family synthesis problem, in this paper an I2I based model is proposed that directly generates the complete font family given a single font style. Our model is trained in an unsupervised setting, i.e., no paired training data are used. More details of our method are described below.

3. Unsupervised font family GAN

The goal of the proposed UFFG method is to learn the global FS features of all styles that form a complete FF. To achieve this, the UFFG aims at mapping an input image from any FS into an output image conditioned on the target FSL. To train the UFFG, we use images from a set of FSs called style classes. The existence of paired images does not exist during training (i.e., a normal font may not have a corresponding font in bold or italic, etc. FS). To learn the target FS, we utilize the style class images that guide the network to learn the local style (strokes, shape, and content) along with the target FSL, which guides the network to learn the global style (e.g., bold and italic). At the inference stage, we provide the model any single FS image of a novel font, and the model should generate the remaining FSs with a global style guided by the FSL and the local style guided by the style class image (input image).

Our model consists of a conditional generator G and a multi-task discriminator D. Here, G simultaneously takes an input image x and a set of randomly generated FSLs c as an input and generates the output image y using

$$y = G(x, \{c_1, c_2, \ldots, c_k\}).$$

The reason for randomly generating c is not only to generalize the model but also to learn the global features of all FSs and make the model flexible for translating any FSs to the rest at the inference stage. In addition, G conditionally maps an input image x to an output image y such that y shares a similar structure and content with the input image but different FSs depending on the conditional FSL.

3.1. Font family image generator

The font-family generator G consists of an encoder E_x and decoder D_x. The encoder input is an image concatenated with the target FSL on the channel axis. This input is passed through several 2D convolutional layers. It maps the input to a latent code z_x. This latent code is then passed to the decoder, which consists of a series of upscaling convolutional layers to generate the target FS image y. We also use skip-connections between the encoder and decoder layers, i.e., every layer in the encoder is connected with the corresponding layer in the decoder excluding the first and last layer of the encoder and decoder, respectively. The key purpose of using skip-connections is to preserve high-level features. Later in the experiments, we demonstrate the effectiveness of using skip-layers. ReLU is used as an activation function for every encoder and decoder layer except the last layer of the decoder, which uses hyperbolic tangent activation.

3.2. Multi-task discriminator

Our discriminator D simultaneously performs adversarial and FSL classification tasks during training. The adversarial task in our discriminator is a binary classification task that determines whether an input image is a real image from the training distribution or a synthesized image from the generator. The FSL classification task in our discriminator is a multiclass classification task that determines the FS of the input image. For the adversarial loss, we use the PatchGAN discriminator which classifies whether the patch in the input image corresponding to the n x n output is real or fake. For the multiclass task, we add an additional fully connected layer at the top of the discriminator to predict the FSL.

3.3. Learning

The learning objective for the proposed UFFG is composed of four loss functions given by the following:

$$\min_D \max_G L_{\text{adv}}(D, G) + \lambda_C L_{\text{cyc}}(G) + \lambda_{\text{CRF}} L_{\text{fsc}}(D, G) + \lambda_F L_{\text{fm}}(G), \quad (1)$$

where $L_{\text{adv}}(G, D)$, $L_{\text{cyc}}(G)$, $L_{\text{fsc}}(D, G)$, $L_{\text{fm}}(G)$ are the adversarial loss for GAN, cycle consistency loss, font style classification loss, and feature matching loss.

Adversarial loss. We use the conditional GAN loss as our adversarial loss given by the following:

$$\min_D \max_G L_{\text{adv}}(D, G) = E_x[-\log D(x)] + E_{x,c}[\log(1 - D(y))], \quad (2)$$

where the discriminator D tries to minimize this loss function by predicting x as a real image and y as a fake image from the generator G, where $y = G(x, c_k)$. By contrast, the generator tries to fool D by maximizing y as a real image.

Cycle consistency loss. Unlike the other state-of-the-art supervised font synthesis methods where the ground truth (GT) image is used for character content reconstruction, i.e., $l_1 = (GT - y)$ (where l_1 is the mean absolute error (MAE), and y is the image generated by the generator), our problem is in unpaired setting, where we do not have the GT images for each font style. Therefore, to have an identical character content for the input and generated image, we utilize the cycle consistency loss. This is given by

$$L_{\text{cyc}}(G) = E_x, c, c^*[\|(x - G(G(x, c), c^*))\|_1], \quad (3)$$

where G takes the synthesized image $G(x, c)$ and the font style label c^* of input image x and attempts to generate an identical

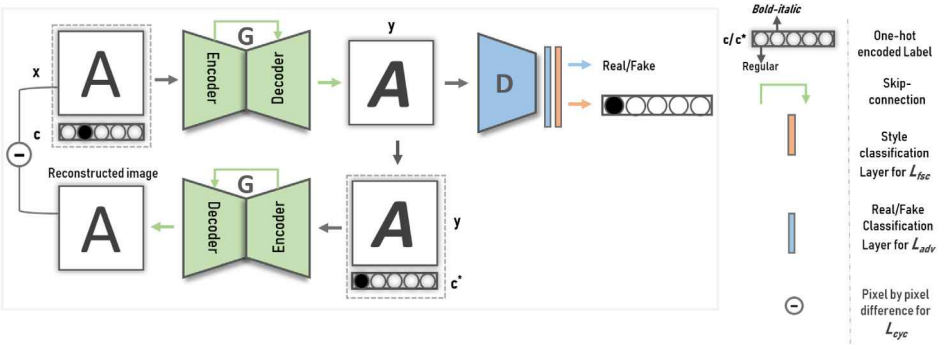

Fig. 1. Visual demonstration of the proposed adversarial, cycle-consistency, and the font style classification losses.

image as the input image x. This loss helps G learn the reconstruction translation model. Specifically, in cases in which the input and target labels c^* and c, respectively, belong to the same *FS*, this loss pushes G to synthesize an output image indistinguishable from the input.

Font style classification loss. Our goal is to synthesize various FSs in a FF conditioned on the FSL as an input to the generator. Specifically, we would like our model to learn the global features of the possible FSs belonging to a FF. To achieve this, we add a fully connected layer on the top of the discriminator that plays the role of a classifier and impose a font style classification loss to differentiate between various FSs belonging to a FF. This font style classification loss is for both the real images and the synthesized images. The discriminator is optimized to minimize the first part of the loss function for real images, whereas the generator is optimized to minimize the second part of this loss for the fake images to learn different FS's (For simplicity we have written the two losses for D and G in a single equation). The loss is given by the following:

$$L_{\text{fsc}}(D, G) = E_{x,c^*}[-\log D(c^*|x)] + E_{x,c}[-\log D(c|y)], \tag{4}$$

where x is the input image, c^* is the original FSL of x and y is the synthesized image conditioned with a target FSL c i.e. $y = G(x, c)$. The adversarial, cycle consistency, and the font style classification losses are demonstrated in Fig. 1.

Feature matching loss. In our experiments, we found that the feature matching loss improves the quality of the generated images. The generated characters are smoother when trained with this loss function. We utilize D as our feature extractor, denoted as D_f instead of using a separate feature extractor such as VGG etc. During training, we use the features from the feature extracting layer of D_f (by removing the real/fake prediction layer and style classification layer from D). We then extract features from the input image x and the generated image y when both conditioned with the same font style labels c and c^*, respectively, and minimize using the l_1 norm. This process is visually demonstrated in Fig. 2. The loss is given by the following:

$$L_{\text{fm}}(G) = E_x, c, c^*[\|(D_f(x) - D_f(y))\|_1]. \tag{5}$$

4. Experiments

4.1. Implementation and network architecture

For loss function hyperparameters, we use $\lambda_C = 10, \lambda_{CRF} = 1, and \lambda_F = 1$. We use the non-saturated GAN loss as our adversarial loss which performs better than the saturated loss [46]. Our main objective in Eq. (1) is trained using the Adam optimizer with a learning rate of 0.0002. We use $256 \times 256 \times 3$ sized images as our inputs. Our generator consists of an encoder–decoder architecture. The encoder takes an input image x and the target FSL as an input, both concatenated on the channel axis. The dimension of the input thus becomes $256 \times 256 \times (3 + N_{FS})$, where N_{FS} is the total number of FSs in a FF. Here, N_{FS} is determined before training depending on the chosen FSs to represent a FF. The encoder then maps this concatenated input to a latent vector z_x of size 16×16 because there are 4 down-sampling layers with a stride of 2. This latent vector is then up-sampled using 4 up-sampling layers. All of the convolutional layers in the encoder are followed by instance normalization [47] and ReLU activation function respectively. As mentioned in Section 4.2, our discriminator is a PatchGAN [15]. We use LeakyReLU [48] as our activation function in the discriminator with no normalization layer. The architecture of our network is shown in Fig. 3. Each block in G represents the number of filters in that layer.

4.2. Dataset

There are a few datasets available for font synthesis problem such as the stylish Latin alphabet dataset proposed by [5] or the stylish Chinese dataset proposed by Gao et al. [6]; however, there are no publicly available datasets for the font family synthesis problem. Therefore, we built our own dataset for this font family synthesis task.

4.2.1. Dataset description

Google Fonts[2] is an open-source font catalog consisting 1023 font families in approximately 29 different languages including English Alphabets, Chinese, and Korean, etc. We downloaded all font families from 3740 fonts in total from Google fonts (all fonts were stored in a single directory with no class subdivision). We then converted these font files into images of English alphabet letters for a total of 52 characters (upper and lower case). We conducted a data preprocessing on the generated images, similar to an image analysis, and discovered that there are fonts that do not support Latin alphabets. Thus, the generated images are blank with no characters. We also found that some of the fonts are too artistic and cursive, and do not fall within our font family synthesis problem. We removed these fonts from the original dataset; hence, our final dataset contains 2,278 font files ($52 \times 2{,}278$ images). We then split our fonts into eight FSs (black, black-italic, bold, bold-italic, regular, italic, medium, light) all having various numbers of fonts, which we refer to as a FF in our paper. Each FS in a FF is represented by its own FSL, i.e., a one-hot vector representing the FS. We then converted all fonts into an image size of 256×256 and then randomly divided them into training and testing datasets with a 90% to 10% ratio. Fig. 4 shows example images of the dataset.

[2] https://fonts.google.com/.

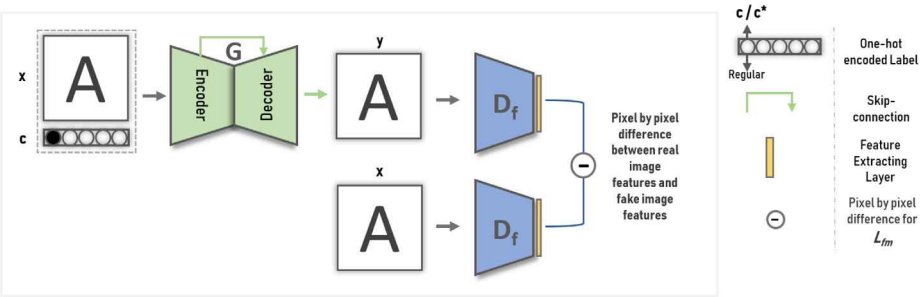

Fig. 2. Visual demonstration of the proposed feature matching loss.

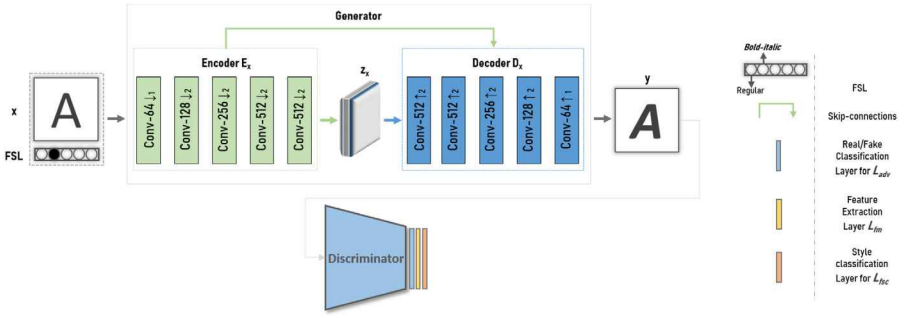

Fig. 3. The generator in our network accepts the input image x and the font style label FSL as inputs, and generates a translation output y that contains the local style information from x and global style information from the FSL. The discriminator is used to classify images as real or fake, to classify font styles, and to extract feature maps for the feature matching loss. Note that the other FF generated images, layer details of the discriminator, activation functions, and normalization layers are not included in the figure for simplicity.

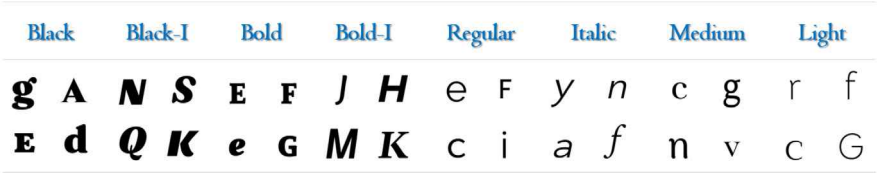

Fig. 4. Examples from the proposed font family dataset.

4.3. Baseline

The font family synthesis problem lays in two categories i.e., unsupervised I2I translation and multi-domain I2I. In an unsupervised I2I setting, the model learns a translation function between two domains, where the training dataset is not paired. CycleGAN [21], is the first state-of-the-art two domain translation model in the unsupervised setting. We picked CycleGAN as our first baseline and trained it for our font family synthesis problem. As CycleGAN is incapable of managing this multiple FS problem, we trained N number of CycleGANs for learning N FS's from a FF. More specifically, we treat regular FS as the first domain and all the other FS's as the second domain (black, black-italic, bold, bold-italic, regular, italic, medium, and light). This results in $N = 8$ CycleGAN models per each FS in our training dataset.

In addition, the font family synthesis problem is also a multi-domain I2I translation problem, where a single generator is used to learn N number of domains in an unpaired setting. Existing font synthesis methods rely on paired datasets but follow the same encoder–decoder architecture trained with an adversarial loss. StarGAN [24] can also manage this unsupervised setting of multi-domain I2I and has achieved some remarkable results. As a result, we also selected StarGAN as the second baseline for this study.

We trained CycleGAN and StarGAN using the authors source code and default parameters, but extended them to solve the font synthesis problem using our proposed font family dataset.

4.4. Evaluation

To measure the effectiveness of our proposed model, we conducted qualitative and quantitative evaluations. We trained the baselines with the default settings including the hyperparameters, architecture, and source code provided by the authors.

Qualitative evaluation. For a qualitative evaluation, we visualized the results of the proposed model on the font family dataset shown in Fig. 5. We can observe from the figure that the proposed model generates high-quality results compared to the baselines. CycleGAN produces better results on various FSs of a FF than StarGAN, however the approach for learning multiple FSs of a FF is computationally expensive and impractical as the number of FSs increases. The synthesized font images from our model are more photorealistic. One possible reason for this is the architecture and loss functions we used for our model. We conducted several ablation studies to justify the architecture and loss functions of our model in Section 4.5. Additional qualitative results are shown in Appendix A.1.

Quantitative evaluation. Existing font synthesis methods are mostly evaluated on the structural similarity index (SSIM), multi-scale structural similarity index (MSSIM), or pixel by pixel difference metrics such as the mean absolute error (MAE) or mean square error (MSE) because they have the ground truth font images in their respective tasks. However, we cannot evaluate

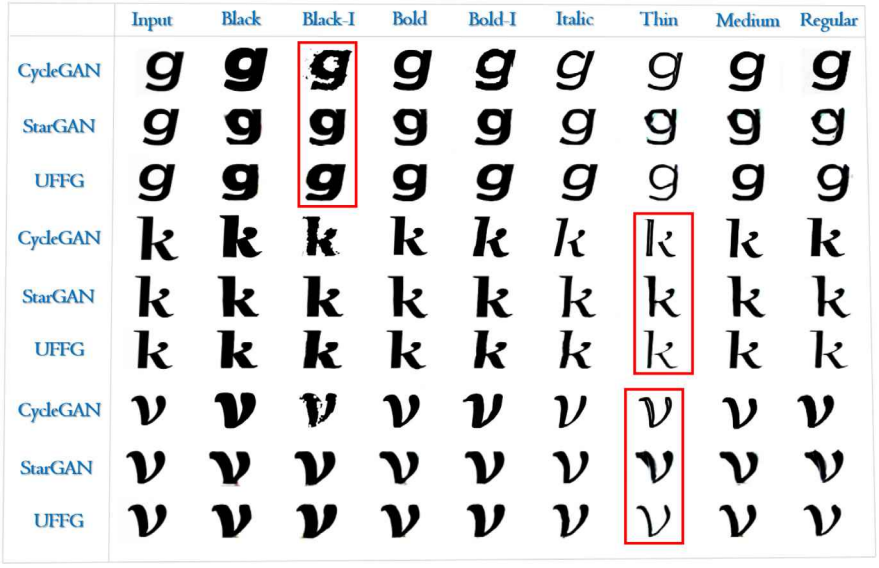

Fig. 5. Visual comparison of our model with CycleGAN and StarGAN. We highlight some of the characters where our model produces realistic and better quality results than the baselines.

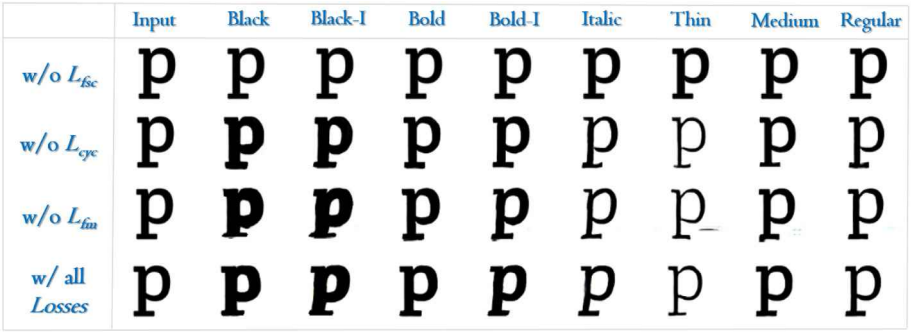

Fig. 6. FF results by changing loss functions. The first column demonstrates the loss functions where w/o means without.

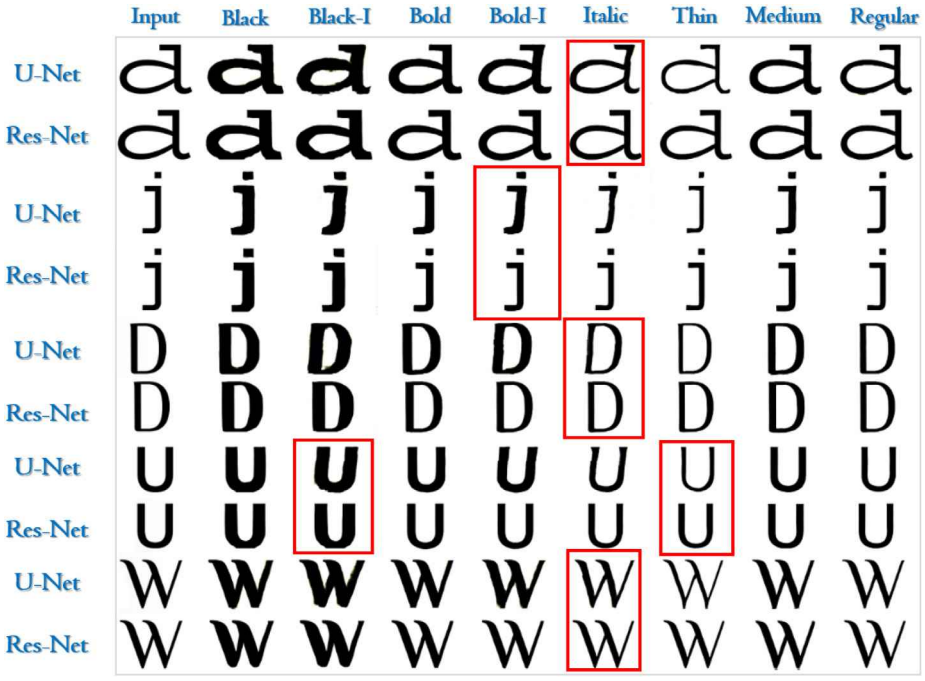

Fig. 7. Comparison of U-Net based generator and Res-Net based generator. We highlight some of the results where the Res-Net based generator is unable to capture the global FSs on the other hand U-Net based generator captures the global style like italic FS.

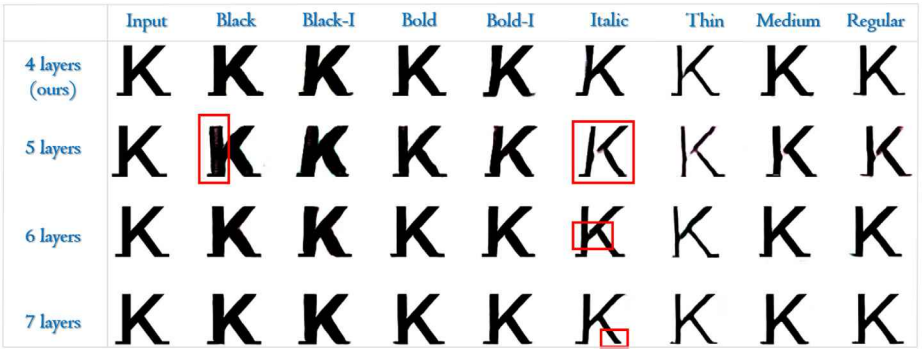

Fig. 8. Comparison of various number of layers in the Generator.

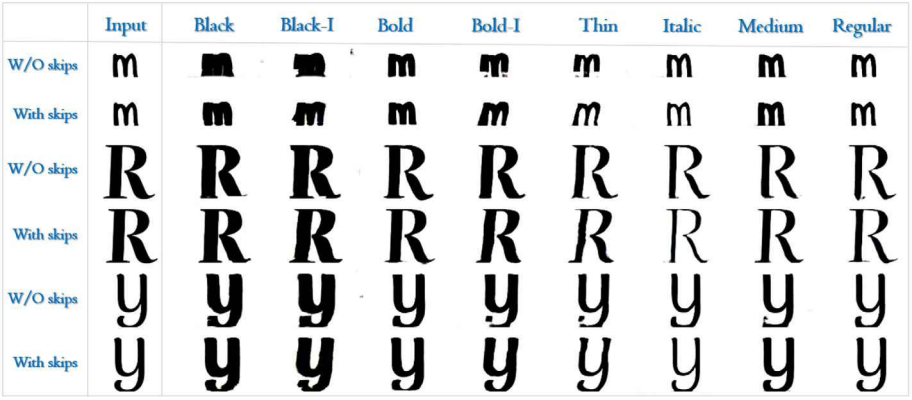

Fig. 9. U-Net based generator consisting of skip layers results in higher quality synthesized FSs.

Fig. 10. Korean hangul characters synthesized by the proposed model. Note that these language characters were never observed by the model during training stage.

our method using these metrics because our setting is unpaired (no ground truths). Therefore, for the quantitative evaluation, we computed the font character classification error and the Frechet Inception Distance (FID) score [49].

Font character classification measure. We measured the accuracy of the synthesized characters by training a classifier for the character classification task, i.e., whether a generated character has the same character content as the ground truth.

Table 1
Quantitative evaluation of proposed method and the baselines. ↑ means that larger numbers are preferable, while ↓ means smaller numbers are preferable.

Method	Accuracy [%] ↑	mclean-FID ↓
CycleGAN	0.9273	57.39
StarGAN	0.9143	61.14
UFFG (Ours)	0.9387	53.48

For this experiment, we trained a classifier with a dataset having a 80% to 20% training and testing split. The classifier consisted of five convolutional layers and a fully-connected layer for character classification. Each convolution layer was followed by the ReLU activation function. In addition, dropout and batch-normalization were applied after the fully-connected layer. The size of the fully connected layer was measured based on the total number of the characters in the dataset, i.e., 52 characters for English alphabets (upper and lower cases) in this setting. We then trained the classifier on the training set and evaluated the classifier with the testing set, which resulted in 99% training and 98% testing accuracy. Next, we trained the proposed method and the baselines with the same training dataset and generated the results on the testing dataset. Finally, we used the synthesized images generated by the methods to test the accuracy of the trained classifier. The results are shown in Table 1.

Frechet Inception Distance (FID). Frechet Inception Distance (FID) is a widely used metric for evaluating generative models. We used FID to evaluate our model's and the baselines synthesized images for the quantitative evaluation. We computed the difference between two distributions, such as the test set and the samples generated by our generator, using the recently proposed clean-FID score [49].

More specifically, we used the proposed network to synthesize FF's from the FS's in the test set. We then computed the clean-FID between each of the FS in the validation set and the corresponding FS generated by our model. This produces $|N|$ clean-FID scores, where $|N|$ denotes the total number of FS's (8 in our case). The mean of these $|N|$ clean-FID scores is then used to calculate our final clean-FID metric, which is referred to as the mean clean-FID (mclean-FID) in our paper. Similarly, for the baselines, we calculate the mclean-FID and visualize the results in Table 1. The proposed method also outperforms the baselines on mclean-FID metric.

4.5. Ablation study

We conducted various experiments to analyze the impact of our loss functions, generator architecture, proposed network generalization capability, and performance of the proposed method for applications other than font family synthesis.

Impact of loss functions. To determine the impact of the individual components of our objective in Eq. (1), we conducted ablation studies. We conducted an experiment in which we compared our full model (model trained with Eq. (1)) against its variations, i.e., by removing each loss at a time. We found the visual effects of each loss function such as the classification loss, the cycle consistency loss, and the feature mapping loss by training N variations, where $N = 3$ as per the losses mentioned above. Fig. 6 demonstrates the visual effects of these variations, and the proposed model, all trained on the same dataset. The feature matching loss when added smoothens the synthesized characters. Adding all terms together results in the best performance by minimizing the artifacts.

Fig. 11. Results of our proposed method trained with Korean characters.

Fig. 12. Chinese characters synthesized by the proposed model trained with only Korean hangul characters. Note that these Chinese characters and font styles were never observed by the model during training stage.

Analysis of the generator architecture. To determine the effectiveness of our proposed generator architecture, we conducted various ablation studies, as mentioned below.

Choosing between the Res-Net and U-Net architectures. Both ResNet and U-Net-based architectures are widely used in I2I applications. All of these architectures are inspired by jhonson et al. [50] proposed architecture for style transfer which uses Res-blocks and a U-Net-based architecture [51] which uses skip connections. To choose an architecture for our font family synthesis problem among these two types, we conducted an experiment in which we trained our model with a generator using Res-blocks [33] versus a model trained using a U-Net generator [15]. Both of these variations were trained on the same dataset and same hyperparameters.

As shown in Fig. 7, the U-Net-based architecture performs better than the Res-Net-based architecture. We highlight some

Fig. 13. Comparison of StarGAN and the proposed UFFG. We highlight the cases where our proposed model performs better than the StarGAN model.

of the characters where the Res-Net-based generator is unable to learn the global features of a FS.

Choosing number of layers for the generator. The U-Net-based generator [15] used 7 layers in the encoder decoder respectively. To find the best possible number of layers for our font family synthesis problem, we conducted the present ablation study. We trained various variations of our model. For each variation, we reduced the number of layers, i.e., 7, 6, 5, and 4. Fig. 8 demonstrates the results of this experiment. From these experimental results, we chose 4 layers for our generator (encoder–decoder respectively).

Impact of skip layers. We conducted this experiment to determine the effectiveness of the skip layers. We trained our model with the proposed U-Net generator (4-layer generator) and the encoder–decoder generator. The encoder–decoder generator is created simply by removing the skip layers from the proposed U-Net generator. As shown in Fig. 9, the generated fonts from the encoder–decoder are poor in quality. The U-Net architecture clearly achieves a better quality than a simple encoder–decoder, thus demonstrating its effectiveness.

Generalization ability. We conducted the following two experiments to verify the generalization ability of the proposed network.

Cross language evaluation. Can the network generate a font family for the unobserved characters, i.e., the characters that were not seen during training by the model? Can the network generate high-quality results on unseen language characters? To answer these two questions, we conducted a single experiment. We trained our model on the Latin alphabet dataset; however, during testing, we fed the trained model with Korean Hangul characters, which had not been seen by the model.

As depicted in Fig. 10, we can see that the font family synthesized by our model is also impressive. We also prepared a dataset for Korean font families and trained the network on the Hangul dataset. The results of our model trained with the Korean Hangul dataset are depicted in Fig. 11. Additionally, we fed Chinese characters to this pre-trained model (model trained with Korean characters) during inference time. Fig. 12 illustrates the results. Given that the model has never encountered these Chinese characters and FS's during pre-training, the model generates plausible results.

Cross application evaluation. We conduct this experiment to demonstrate that, in addition to the font family synthesis task, our model also produces high-quality images on different datasets. To demonstrate this, we trained our proposed model and StarGAN on the celebA [10] dataset.

As shown in Fig. 13, our model performs better on the facial dataset than the baseline, which was originally proposed for this multi-domain face synthesis task. The synthesized faces are smoother and more realistic than the baseline. This experiment demonstrates the effectiveness of our proposed architecture and the combination of loss functions.

4.6. Failure cases and future work

From our experiments, we noted that the proposed model does not synthesize high-quality images when given cursive font as an input during testing, probably because the model did not see a sufficient number of cursive font styles during training.

Our proposed model is currently able to learn a specified font family, i.e., the font family styles are predefined, and the model learns them based on the font style label, which is a one-hot label in our current setting. Learning a new font style as a part of the font family is impossible with the current model and may require a further fine-tuning step (transfer learning). To solve this issue, we plan to modify our current architecture by adding few-shot learning strategies in a future study.

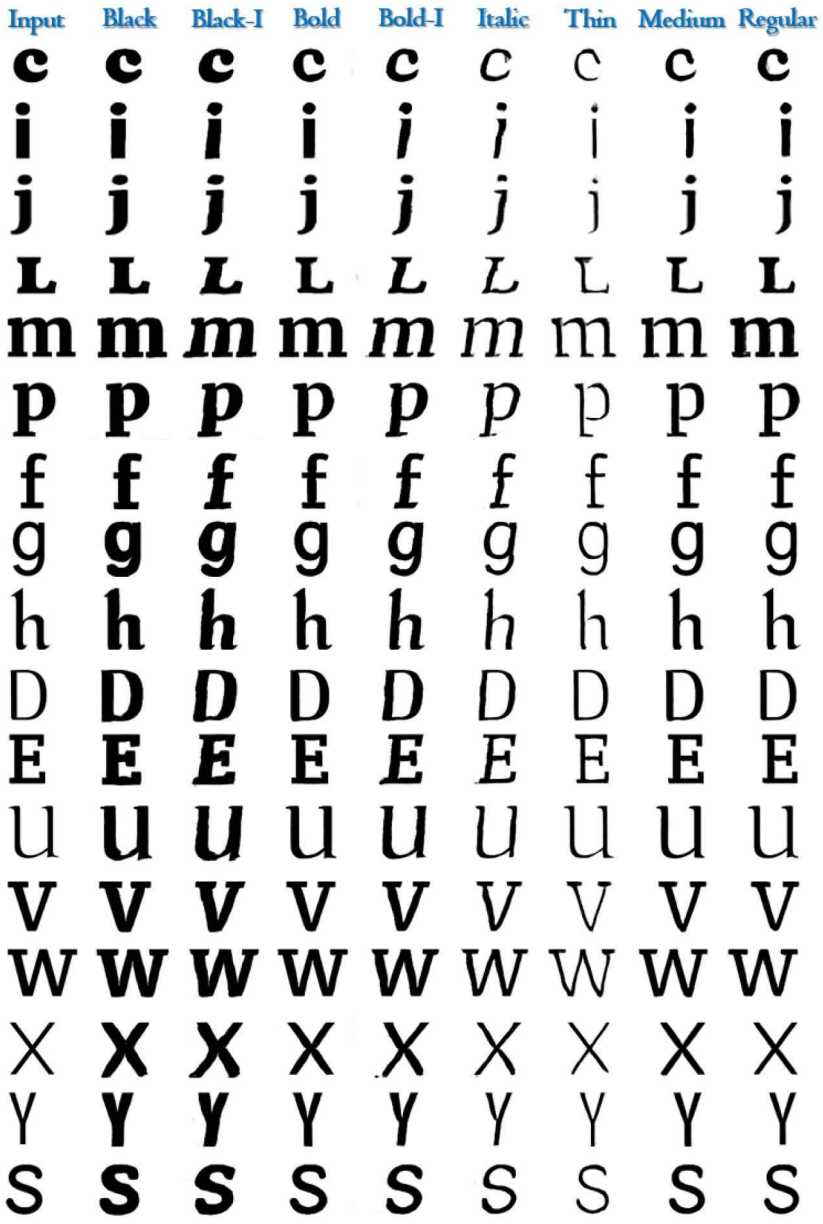

Fig. 14. Additional qualitative results on alphabets of our proposed method.

Fig. 15. Additional qualitative results on Korean hangul characters of our proposed method. Zoom in for better view.

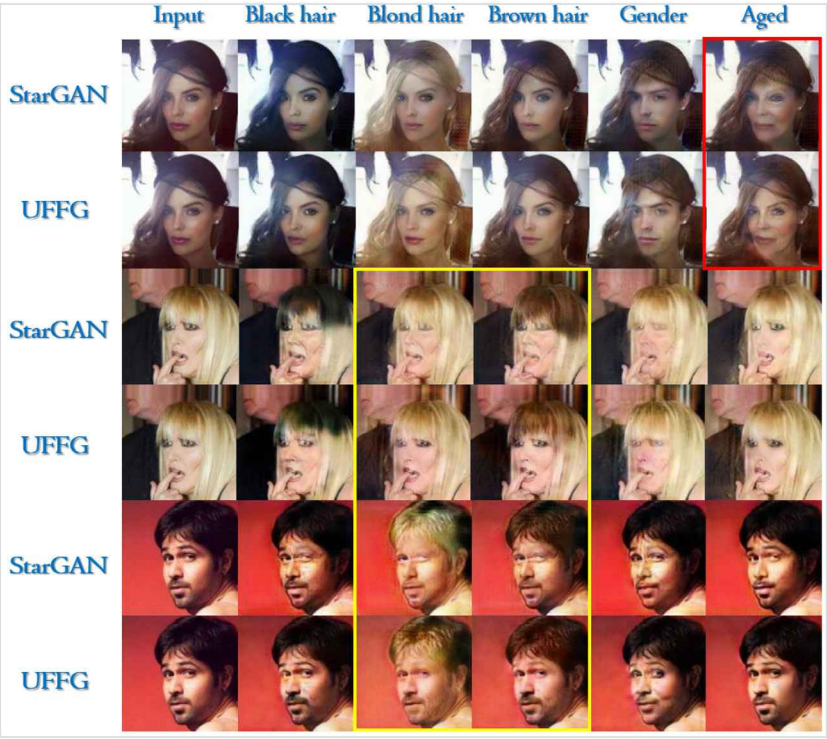

Fig. 16. Additional comparison of our proposed method with StarGAN on CelebA dataset. We highlight the cases where our proposed model performs better than the StarGAN model.

5. Conclusion

In this paper, we introduced a new problem of a font family synthesis and proposed a network to solve this problem based on conditional GANs. We showed that, unlike other state-of-the-art font synthesis models, that rely on paired datasets during training (supervised image-to-image translation) to generate a single font at a time, our model can generate a full font family in an unpaired setting (unsupervised image-to-image translation). We conducted qualitative and quantitative experiments to demonstrate the effectiveness of the proposed method. Extensive ablation studies were also conducted to show the effectiveness of the architecture of the proposed network and the generalization capability of the method.

CRediT authorship contribution statement

Ammar Ul Hassan: Conceptualization, Methodology, Validation, Writing – original draft, Writing – review & editing. **Hammad Ahmed:** Software, Data curation, Visualization. **Jaeyoung Choi:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

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

Appendix

A.1. Additional qualitative results

See Figs. 14–16.

References

[1] Y. Tian, zi2zi: Master Chinese calligraphy with conditional adversarial networks, 2017, URL https://github.com/kaonashi-tyc/zi2zi.
[2] Y. Jiang, Z. Lian, Y. Jianguo, J. Xiao, DCFont: an end-to-end deep Chinese font generation system, in: SIGGRAPH Asia, 2017, p. 22, TB.
[3] Y. Jiang, Z. Lian, Y. Tang, J. Xiao, SCFont: structure guided Chinese font generation via deep stacked networks, 2019.
[4] B. Chang, Q. Zhang, S. Pan, L. Meng, Generating handwritten Chinese characters using cyclegan, 2018, CoRR, arXiv:abs/1801.08624.
[5] Samaneh Azadi, et al., Multi-content GAN for few-shot font style transfer, in: 2018 IEEE/CVF conference on Computer Vision and Pattern Recognition, 2018, pp. 7564–7573.
[6] Yue Gao, et al., Artistic glyph image synthesis via one-stage few-shot learning, ACM Trans. Graph. 38 (2019) 1–12.
[7] J. Cha, S. Chun, G. Lee, B. Lee, S. Kim, H. Lee, Few-shot compositional font generation with dual memory, in: ECCV, 2020.
[8] Song Park, Sanghyuk Chun, Junbum Cha, Bado Lee, Hyunjung Shim, Few-shot font generation with localized style representations and factorization, in: AAAI Conference on Artificial Intelligence, 2021.
[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems (NIPS), 2014, pp. 2672–2680.
[10] Z. Liu, P. Luo, X. Wang, X. Tang, Deep learning face attributes in the wild, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015.
[11] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, 2015, arXiv preprint arXiv:1511.06434.
[12] M. Mirza, S. Osindero, Conditional generative adversarial nets, 2014, arXiv preprint arXiv:1411.1784.
[13] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: Proceedings of the 34th International Conference on Machine Learning (ICML), 2017, pp. 214–223.

[14] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. Courville, Improved training of wasserstein gans, 2017, arXiv preprint arXiv:1704.00028.
[15] P. Isola, J.-Y. Zhu, T. Zhou, A.A. Efros, Image-to-image translation with conditional adversarial networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
[16] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, Bryan Catanzaro, High-resolution image synthesis and semantic manipulation with conditional gans, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.
[17] H. Zhang, et al., StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks, in: 2017 IEEE International Conference on Computer Vision (ICCV), Venice, 2017, pp. 5908–5916, http://dx.doi.org/10.1109/ICCV.2017.629.
[18] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, T. Aila, Analyzing and improving the image quality of StyleGAN, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020, pp. 8107–8116, http://dx.doi.org/10.1109/CVPR42600.2020.00813.
[19] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, T.S. Huang, Generative image inpainting with contextual attention, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018, pp. 5505–5514, http://dx.doi.org/10.1109/CVPR.2018.00577.
[20] C. Ledig, et al., Photo-realistic single image super-resolution using a generative adversarial network, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 105–114, http://dx.doi.org/10.1109/CVPR.2017.19.
[21] J.-Y. Zhu, T. Park, P. Isola, A.A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017.
[22] Z. Yi, H. Zhang, P. Tan, M. Gong, DualGAN: Unsupervised dual learning for image-to-image translation, in: 2017 IEEE International Conference on Computer Vision (ICCV), Venice, 2017, pp. 2868–2876, http://dx.doi.org/10.1109/ICCV.2017.310.
[23] M.-Y. Liu, T. Breuel, J. Kautz, Unsupervised image-to-image translation networks, 2017, arXiv preprint arXiv:1703.00848.
[24] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, Jaegul Choo, Stargan: Unified generative adversarial networks for multi-domain image-to-image translation, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.
[25] L. Hui, X. Li, J. Chen, H. He, J. Yang, Unsupervised multi-domain image translation with domain-specific encoders/decoders, in: ICPR, 2018.
[26] M. Li, W. Zuo, D. Zhang, Deep identity-aware transfer of facial attributes, 2016, arXiv preprint arXiv:1610.05586.
[27] Z. Shu, E. Yumer, S. Hadap, K. Sunkavalli, E. Shechtman, D. Samaras, Neural face editing with intrinsic image disentangling, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
[28] Y. Choi, Y. Uh, J. Yoo, J.-W. Ha, StarGAN v2: Diverse image synthesis for multiple domains, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020, pp. 8185–8194, http://dx.doi.org/10.1109/CVPR42600.2020.00821.
[29] M.Y. Liu, X. Huang, A. Mallya, T. Karras, T. Aila, J. Lehtinen, J. Kautz, Fewshot unsupervised image-to-image translation, in: IEEE International Conference on Computer Vision, 2019.
[30] K. Saito, K. Saenko, M.-Y. Liu, COCO-FUNIT: Few-shot unsupervised image translation with a content conditioned style encoder, in: ECCV, 2020.
[31] A. Odena, C. Olah, J. Shlens, Conditional image synthesis with auxiliary classifier GANs, in: Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 2642–2651.
[32] Y. Taigman, A. Polyak, L. Wolf, Unsupervised cross-domain image generation, arXiv preprint arXiv:1611.02200.
[33] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: CVPR, 2016.
[34] G. Huang, Z. Liu, K.Q. Weinberger, L. van der Maaten, Densely connected convolutional networks, in: CVPR, 2017.
[35] H. Ko, A. Ul Hassan, J. Suk, J. Choi, SKFont: Skeleton-driven Korean font generator with conditional deep adversarial networks, J. Int. J. Doc. Anal. Recognit. (2021) URL https://doi.org/10.1007/s10032-021-00374-4.
[36] Raphael Gontijo Lopes, David Ha, Douglas Eck, Jonathon Shlens, A learned representation for scalable vector graphics, 2019, arXiv preprint arXiv:1904.02632.
[37] D.P. Kingma, M. Welling, Auto-encoding variational bayes, in: Proceedings of International Conference on Learning Representations, 2014.
[38] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[39] A. Graves, Generating sequences with recurrent neural networks, 2013, arXiv preprint arXiv:1308.0850.
[40] H. Hayashi, K. Abe, S. Uchida, GlyphGAN: Style-consistent font generation based on generative adversarial networks, Knowl.-Based Syst. (2019).
[41] Shusen Tang, et al., FontRNN: Generating large-scale Chinese fonts via recurrent neural network, Comput. Graph. Forum 38 (2019).
[42] H. Ko, A. Ul Hassan, S. Majeed, J. Choi, SkelGAN: A font image skeletonization method, J. Inf. Process. Syst. (2021).
[43] Po-Wei Wu, Yu-Jing Lin, Che-Han Chang, Edward Y. Chang, Shih-Wei Liao, Relgan: Multi-domain image-to-image translation via relative attributes, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 5914–5922.
[44] Ming Liu, Yukang Ding, Min Xia, Xiao Liu, Errui Ding, Wangmeng Zuo, Shilei Wen, STGAN: A unified selective transfer network for arbitrary image attribute editing, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 3673–3682.
[45] Yizhi Wang, Yue Gao, Zhouhui Lian, Attribute2Font: Creating fonts you want from attributes, ACM Trans. Graph. 39 (2020).
[46] K. Kurach, M. Lucic, X. Zhai, M. Michalski, Landscape: losses, architectures, regularization, and normalization, in: International Conference on Learning Representations, 2019.
[47] D. Ulyanov, A. Vedaldi, V. Lempitsky, Instance normalization: The missing ingredient for fast stylization, 2016, arXiv preprint.
[48] A.L. Maas, A.Y. Hannun, A.Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: ICML, 2013.
[49] Gaurav Parmar, Richard Zhang, Jun-Yan Zhu, On buggy resizing libraries and surprising subtleties in FID calculation, 2021, https://arxiv.org/pdf/2104.11222.pdf.
[50] J. Johnson, A. Alahi, L. Fei-Fei, Perceptual losses for real-time style transfer and super-resolution, in: ECCV, 2016.
[51] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: MICCAI, 2015.