Synthesis of Human Inspired Intelligent Fonts using Conditional-DCGAN *

Ranjith Kalingeri, Vandana Kushwaha, Rahul Kala, and G C Nandi

Center of Intelligent Robotics
Indian Institute of Information Technology Allahabad
Prayagraj-211015, U.P.- INDIA
ranji.iitb@gmail.com, kush.vandu@gmail.com
rkala@iiita.ac.in, gcnandi@iiita.ac.in

Abstract. Despite numerous fonts already being designed and easily available online, the desire for new fonts seems to be endless. Previous methods focused on extracting style, shape, and stroke information from a large set of fonts, or transforming and interpolating existing fonts to create new fonts. The drawback of these methods is that generated fonts look-alike fonts of training data. As fonts are created from human handwriting documents, they have uncertainty and randomness incorporated into them, giving them a more authentic feel than standard fonts. Handwriting, like a fingerprint, is unique to each individual. In this paper, we have proposed GAN based models that automate the entire font generation process, removing the labor involved in manually creating a new font. We extracted data from single-author handwritten documents and developed and trained class-conditioned DCGAN models to generate fonts that mimic the author's handwriting style.

Keywords: DCGAN · Intelligent fonts · Random fonts · cGAN · FID · WCFID · BCFID

1 Introduction

The font is a graphical representation of a text. It is characterized by its design, weight, color, point size, and typeface. There are tons of use cases out of which some are listed below:

- Fonts add style and visual appeal to documents, web pages, newspapers, books, billboards, etc.
- Out of a vast number of fonts, specific fonts are selected based on specific medium, background, and context. Even in the same document different fonts are used for different sections, authors, headings, titles, etc.

The fonts evolved into different classes like random fonts, variable fonts, handwriting style fonts, etc are shown in Fig. 1.

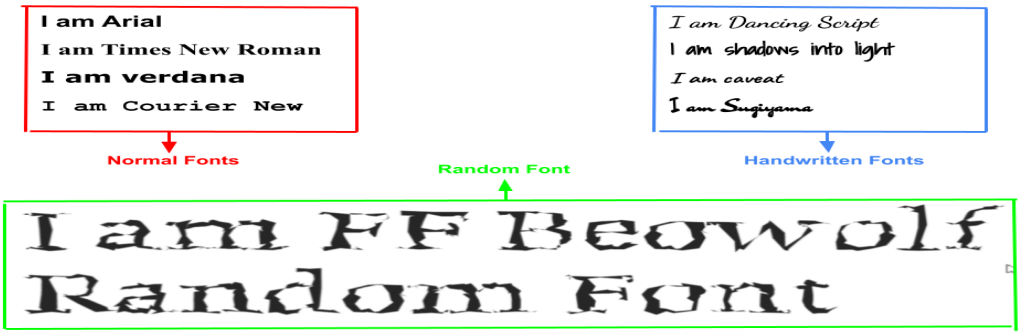


Fig. 1: Normal fonts, Handwritten fonts and Random font

* Supported by I-Hub foundation for Cobotics

1.1 Handwritten Fonts

Handwritten fonts, as shown in Fig.1 resembles human handwriting style. Content created using these fonts look similar to content written with a pen or marker by hand. Handwritten fonts mirror the penmanship, so this kind of font can be integrated and used in various informal communications like:

- WhatsApp chats or any social messaging chat box.
- Personal letters and mails, greeting cards, invitation cards, and personalized notes preparations.

Usually, these fonts are not incorporated with randomness (refer subsection 1.2) or intelligence (refer subsection 1.3). Font's appearance is the same every time they are displayed.

1.2 Random Fonts

Van Rossum has said, "A certain roughness or varying unevenness is quite pleasing to the eye". So, he and Van Blokland brought uncertainty and randomness to typography by co-designing the first random font, "Beowolf". In this font, the character appears differently every time it is displayed or printed, as shown in Fig. 1. This is achieved by randomly shifting the ragged edges each time the font is displayed. Random fonts are the first stepping stones toward intelligent fonts.

1.3 Intelligent Fonts

Intelligent fonts alter their appearance depending on its previous and succeeding character. Like random fonts, characters in intelligent fonts appears differently every time they are displayed or printed. The dependence of intelligent fonts on adjacent characters distinguishes them from random fonts, which has random appearance.

Intelligent font's concept is inspired from human handwriting style. In normal human hand writings, we found that the same character is written in different ways based on its adjacent characters. For example, in Fig. 2, the character 'r' (underlined in blue color) between 'e' and 's' is different from 'r' (underlined in green color) written between 'u' and '␣'. However, character 'r' between 'e' and 's' is similar to 'r' (underlined in red color) written between 't' and 'u' but not exactly the same.

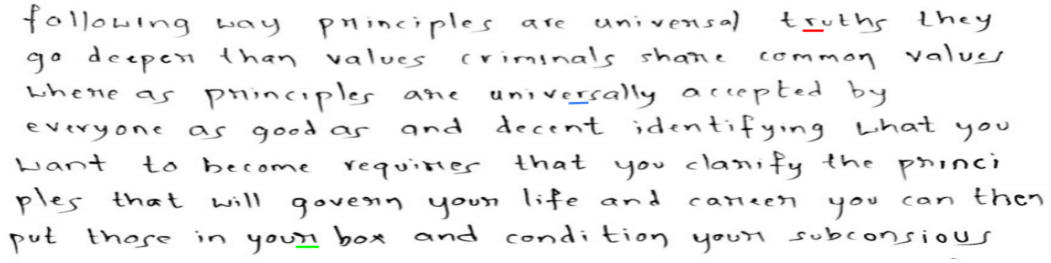


Fig. 2: Represents behavior of intelligent font

1.4 Motivation

This study aims to generate fonts from human handwritten TXT files to mimic the human handwritten style and computerize font generation instead of manually designing fonts due to the following reasons:

- Just as a fingerprint, handwriting is unique to each person. It is possible to create as many fonts as there are many humans with our model.
- In intelligent font, the appearance of each alphabet varies depending on its adjacent alphabets. So, a type designer needs to create large variations of alphabets to develop an intelligent font.
- As uncertainty and randomness is incorporated in intelligent fonts, it gives a more natural feeling than normal font.

1.5 Our contribution

Here are the lists of contribution of our work:

1. We have proposed R-cDCGAN and I-cDCGAN models, two-class conditioned versions of DCGAN for generating random font and intelligent font, respectively.
2. We have collected key data from handwritten documents, named it as IIITA-ALPHABETS dataset, that is used for training our models. This method can also be used to extract key data from handwritten documents of other users.
3. We have proposed an algorithm to form sentences from generated alphabets.
4. The experimental results show that the sentences formed using synthetic fonts generated from proposed models are incorporated with uncertainty and randomness, giving them a more natural depiction than standard fonts.
5. We have also computed FID, WCFID and BCFID for our proposed models R-cDCGAN and I-cDCGAN for its performance evalution.

2 Literature Review

There have been various attempts lately to use generative adversarial networks (GANs) [1] for font generation. One of these methods is zi2zi [2] which combines the pix2pix [3], auxiliary classifier GAN (AC-GAN) [4], and domain transfer network [5] to convert a specific font pattern into a target font pattern. Even though generated fonts have sharp edges and a variety of styles, the target font is limited to alphabets with a significant number of character sets, such as Japanese, Chinese, and Korean, hence, applying this method to alphabets with few letters is challenging.

Several alternative approaches like example-based font generation using GANs are proposed. Chang and Gu [6] used U-net [7] architecture as generator for producing character patterns with the desired style, which is more efficient than zi2zi at balancing the loss functions. Azadi et al. [8] used conditional GAN (cGAN) [9] to generate fonts with limited examples.

In some studies, style and shape of the target alphabet are used as input for their GAN-based font creation network. Guo et al. [10] fed their GAN with skeleton vector of the target character and font style vector to generate fonts with target character. Later, Lin et al. [11] introduced a stroke-based font creation method in which two trained styles are interpolated by modulating a weight.

In the above GAN-based font generation methods, a new typeface is created by combining style information with character shape information retrieved from input character images. Since the generated font is influenced by the geometry of the input image, creating a novel font is difficult. We attempted to overcome this problem through our proposed models, intelligent font- conditional deep convolutional GAN (I-cDCGAN) and random font-cDCGAN (R-cDCGAN), to generate fonts inspired by human handwritten style. First, we extracted shape and location information related to all alphabets from the author's handwritten documents and used this extracted information to train class-conditioned DCGAN to produce fonts that mimic the author's handwriting style.

3 Dataset

3.1 About Dataset

We created a dataset consisting of, 12552 lower case alphabetical character images of size 64×64 pixel from handwritten documents of a single author and named it IIITA-ALPHABETS dataset as shown in Fig 3.

The dataset is in the form of a CSV file, as shown in Table 1. It has 5 columns and, 12552 rows, where each row represents an alphabet of the dataset and columns represents crucial information related to alphabet as described below:

(a) The first column and the third column refers to previous and next alphabets of the current alphabet (second column).

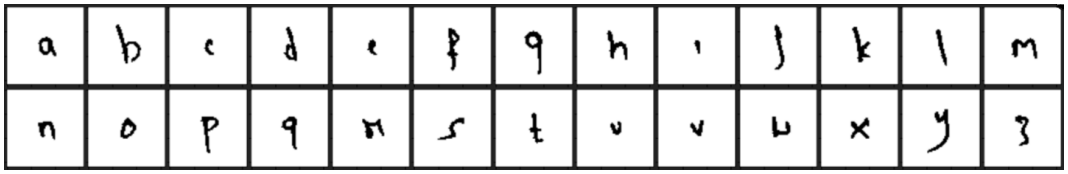


Fig. 3: Sample of a to z alphabets collected from IIITA-ALPHABETS data set

(b) The fourth column contains spacial information of the current alphabet in the handwritten PNG image file. It is in the form of python 2D list like [[ele1, ele2], [ele3, ele4]]. ele1 refers to the line number of the current alphabet in a handwritten PNG image file, ele2 refers to after position of the current alphabet in the ele1 line. ele3 and ele4 represent Y-mean and X-mean of the current alphabet in the handwritten PNG image file in pixel units.
(c) The fifth column represents pixel values of the current alphabet's image of size 64×64.

Table 1: First three rows in CSV file of IIITA-ALPHABETS data set

Previous Alphabet	Current Alphabet	Next Alphabet	Current Alphabet Position	Current Alphabet Shape
␣	f	o	[[0,0], [39.14,67.000]]	[[255, 255, . . . , 255], ...]
f	o	l	[[0,1], [44.83,84.265]]	[[255, 255, . . . , 255], ...]
o	l	l	[[0,2], [40.78,100.57]]	[[255, 255, . . . , 255], ...]

Note that preceding and succeeding character of the current alphabet need not be only alphabets they can be spaces ('␣') too, as the alphabet might be present at the end or start of the word in the handwritten PNG image file.

3.2 Dataset Collection Process

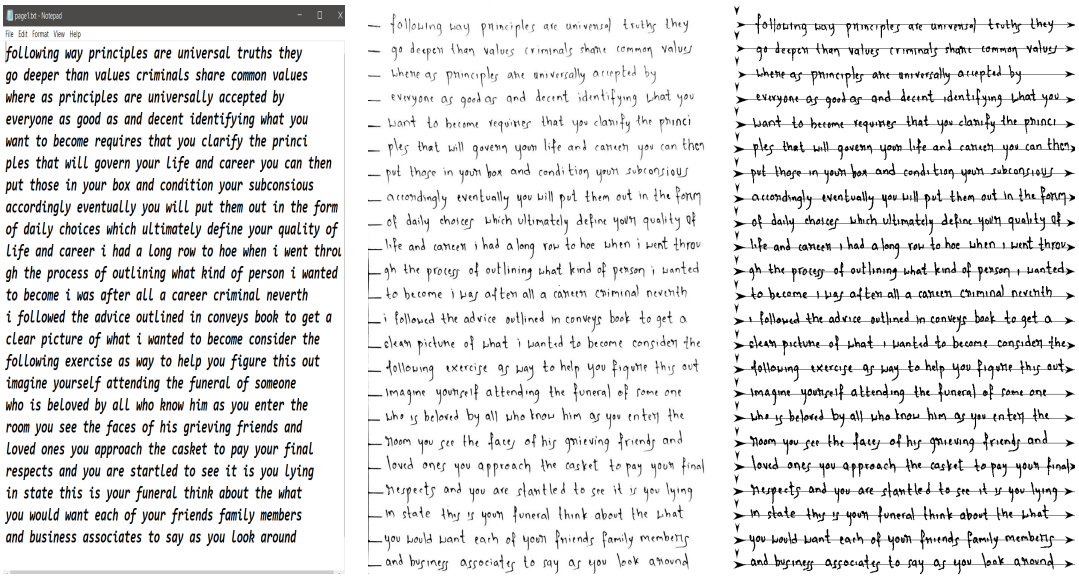
1. Twelve TXT files (Fig. 4a) of text data are collected. From this text data, all non-alphabetic characters are removed except spaces and all upper case alphabets are converted into lowercase.
2. Then, the author is asked to manually write content in each TXT file on an A4 sheet. So, an equivalent handwritten PNG image (Fig. 4b) corresponding to each TXT file is created.
3. We traversed left to right on every handwritten line in the handwritten PNG images (Fig. 4c). During traversal, if we found any alphabet that is not encountered yet, we collected and stored the spatial and shape information related to that alphabet in the 4th and 5th column of CSV file (Table 1)) respectively.
4. The information about previous and next alphabets of currently found alphabet is collected from TXT file and is stored in 1st and 3rd columns of CSV file (Table 1)) respectively, while the current alphabet is stored in the 2nd column.

Like this, with TXT files and their respective user handwritten PNG images, shape and spatial information of each alphabet is collected and stored in CSV file.

4 Preliminary Knowledge

4.1 Generative Adversarial Network (GAN)

GAN is proposed by Ian Goodfellow et al. in 2014 [1], consists of two neural networks: the generator G and the discriminator D. The generator accepts a z-dimensional vector of random numbers as input and outputs data with the same dimensions as the training data. The discriminator, on the other hand, distinguishes between samples from genuine data and data created by the generator.



(a) (b) (c)

Fig. 4: (a) TXT file (b)Handwritten PNG image of TXT file (c) Direction of traversal on Handwritten PNG images, arrows indicate top to bottom and left to right traversal.

G and D play the minimax game in training with the value function $\mathcal{L}_{GAN}$ is represented by (1):

$$\min_G \max_D \mathcal{L}_{GAN}(G,D) = \mathbb{E}_{x\sim P_{data}(x)}\big[\log D(x)\big] + \mathbb{E}_{z\sim P_z(z)}\big[\log\big(1-D\big(G(z)\big)\big)\big] \tag{1}$$

Where $P_{data}(x)$ and $P_z(z)$ denote the training data and z distributions, respectively. The $D(x)$ refers to discriminator output, probability of x belongs to the real data distribution, while mapping from z to data space is denoted by $G(z)$.

4.2 Conditional GAN (cGAN)

In traditional GANs, predicting what sort of pattern would be created from a given input z via G is tricky. Mirza et al. [9] demonstrated a conditional GAN that controls the class of the output image by adding class information (y) encoded as a one-hot vector to the generator's input and a channel encoding the class to the discriminator's input. The loss function of cGAN is represented by (2):

$$\min_G \max_D \mathcal{L}_{cGAN}(G,D) = \mathbb{E}_{x\sim P_{data}(x)}\big[\log D(x\mid y)\big] + \mathbb{E}_{x\sim P(z)}\big[\log\big(1-D\big(G(z\mid y)\big)\big)\big] \tag{2}$$

4.3 Deep Convolutional GAN (DCGAN)

DCGAN proposed by Metz et al. [12], is a variant of GAN architecture based on convolutional neural networks (CNNs). The DCGAN architectural guidelines are given below:

- Strided convolutions in D and fractional-strided convolutions in G are used instead of any pooling layers.
- Both G and D uses batchnorm.
- For deeper architectures, remove fully connected hidden layers.
- Tanh activation is used for the last layer, while other layers uses ReLU activation in G.
- For all layers, use LeakyReLU activation in D.

However, the loss function of DCGAN is the same as GAN.

5 Proposed Methodology

We implemented class-conditional DCGAN (cDCGAN), where class label (y) is initially fed into the embedding layer to obtain dense vector representation. This dense vector is concatenated with the noise (z) and input image (x) of the generator (G) and discriminator (D) of DCGAN, respectively.

5.1 Random Font-cDCGAN (R-cDCGAN)

R-cDCGAN is proposed to generate synthetic handwritten alphabets (Random font) of a given class. For training of R-cDCGAN, IIITA-ALPHABETS dataset consisting of 26 classes is used. Alphabet 'a' is labelled as class 0, 'b' as class 1, and so on. The class label is passed into the embedding layer to generate 100-dimensional one-hot encoded vector, which is further concatenated with 100-dimensional uniformly distributed noise. Then, the resultant vector is fed as input into the generator of R-cDCGAN.

Similarly, in the discriminator, the class label is projected into 64×64 dimensional vector space when passed through the embedding layer. This resultant encoded vector and 64×64 pixel image input are concatenated before being fed as input into the discriminator of R-cDCGAN.

5.2 Intelligent Font-cDCGAN (I-cDCGAN)

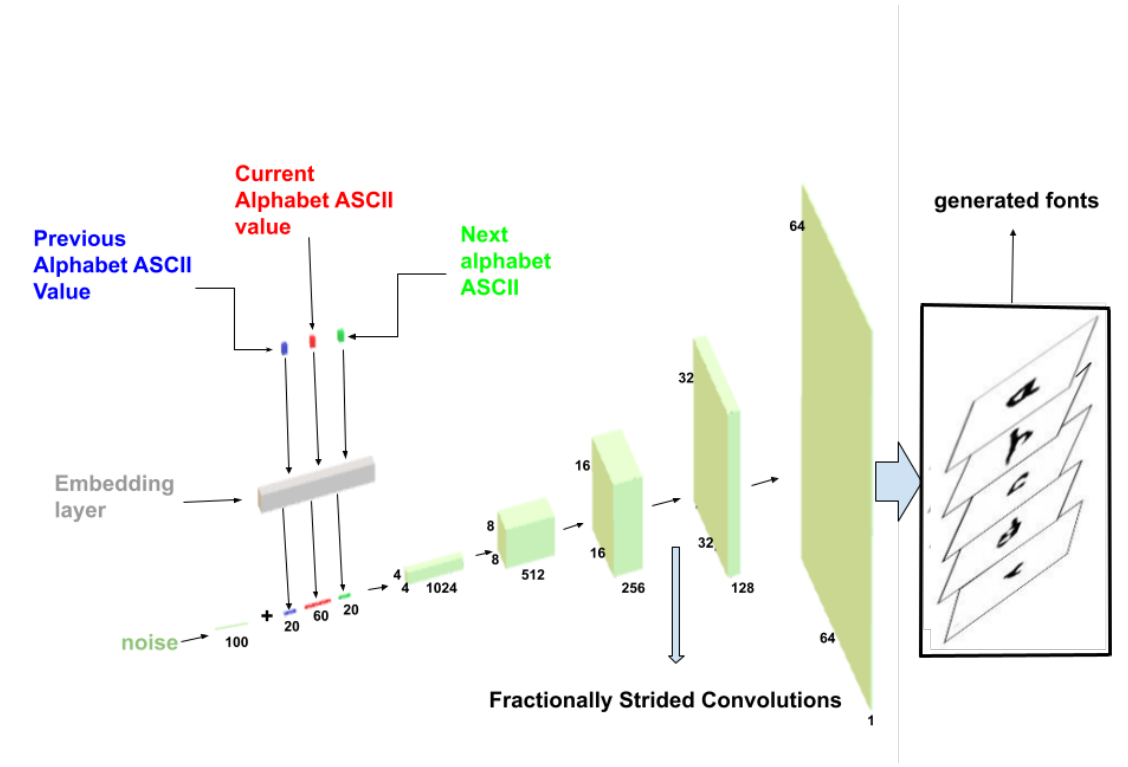
I-cDCGAN (Fig. 5) is proposed to generate synthetic handwritten alphabets of a given class whose appearance depends on the previous and next alphabets (Intelligent font). As shown in Fig. 5a, previous, current, and next class labels are passed into the embedding layer to produce 20, 60, and 20 dimensional class encoded vectors, respectively. This resultant class encoded vectors and 100-dimensional uniformly distributed noise are concatenated before being fed as input into the generator (Fig. 5a) of I-cDCGAN.

Similarly, in I-cDCGAN discriminator (Fig. 5b), current, previous, and next class labels are passed into the embedding layer to produce $(3 \times 64 \times 64)$(channel $\times$ height $\times$ width), $(1 \times 64 \times 64)$ and $(1 \times 64 \times 64)$ class encoded dense vectors respectively. The resultant class encoded vector and 64×64 input pixel image are concatenated before being fed as input into the discriminator of I-cDCGAN.

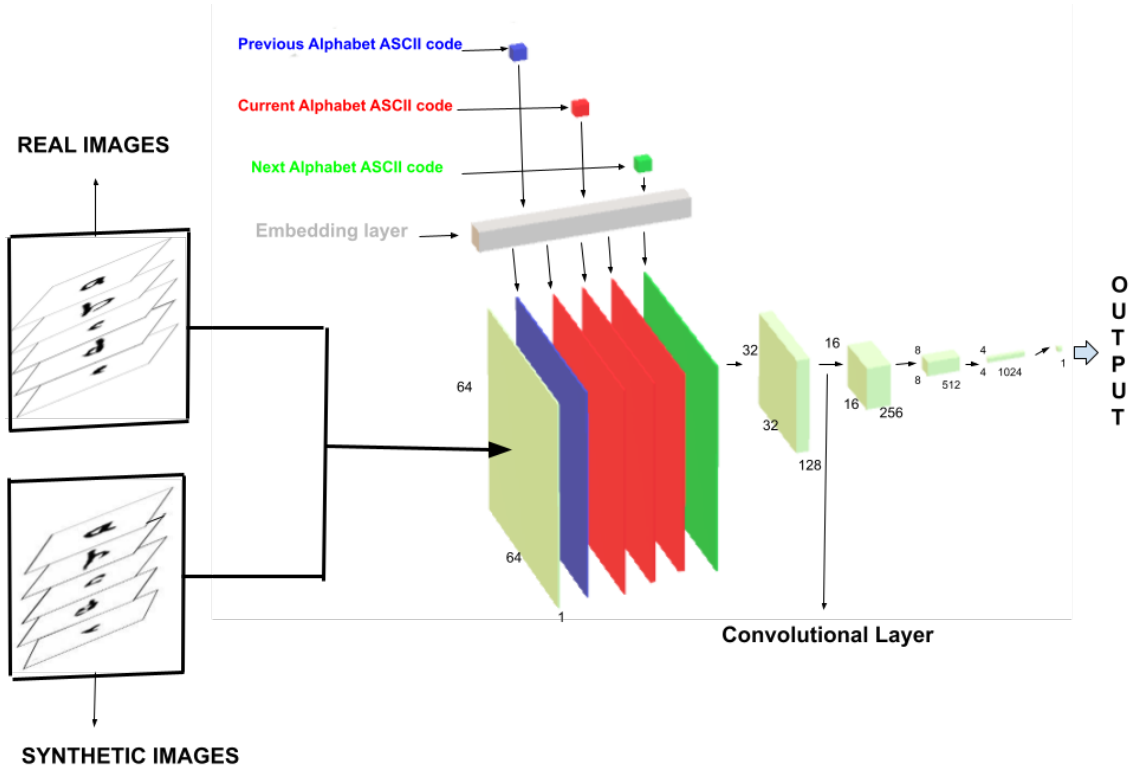
5.3 Sentence Formation from Generated Alphabets

1. The generated alphabets from proposed models are projected on a new white image at specific target pixel locations to form a sentence. The target pixel location of the first alphabet is $[64, 64]$ by default. In Fig. 6 the alphabet 'h' is projected at pixel location $[64, 64]$.
2. Above target pixel location is obtained by adding displacement values (In Fig. 6, 'o' is projected 4 rows below and 21 columns right to its previously projected alphabet 'h'. In this case, $[4, 21]$ is considered as displacement values) to the target pixel location of the previously projected alphabet.
3. From the data set, all instances where the current alphabet is followed by the previous alphabet are taken into consideration. The mean of all displacement values collected from those instances give an estimated displacement at which the current alphabet should be projected from the previous alphabet.
4. If there is a space between the previous alphabet and the current alphabet, then we added an extra 30 pixels to the column displacement value. In Fig. 6 the mean displacement of 'a' from 'w' is $[-1, 37]$ and by adding an extra 30 pixels to the column displacement value, the resultant mean displacement values become $[-1, 67]$.

Steps 2 to 4 are repeated until all generated alphabets are projected onto a new image to form a sentence.



(a) Generator



(b) Discriminator

Fig. 5: Structure of (a) Generator and (b) Discriminator in I-cDCGAN. R-cDCGAN structure is similar to I-cDCGAN except only current class label information is fed into the generator and the discriminator.

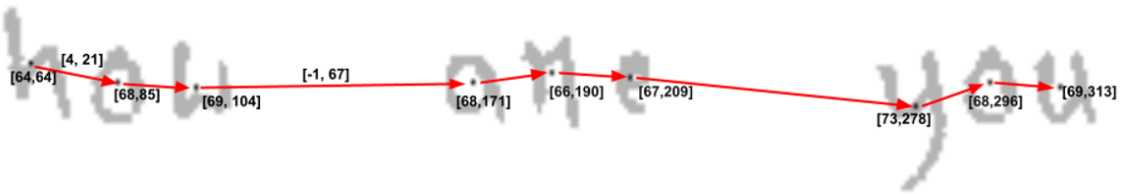


Fig. 6: Sentence formation with generated fonts using our proposed models.

6 Results and Analysis

6.1 Model training and parameters of R-cDCGAN and I-cDCGAN

We have tried some combination of parametric values for training our proposed models and out of those the best ones are mentioned here. For weight updation, Adam optimizer [13] is used with the parameter, $\beta_1 = 0$ and $\beta_2 = 0.9$ and learning rate $= 2e - 4$. We have initialized all the weights using Gaussian distribution with mean $= 0$ and standard deviation $= 0.02$. For the LeakyReLU, the slope is 0.2 in all models. Per generator iteration, the number of discriminator iterations is set to 5. The number of learning iterations per epoch $= 12552$, batch size $= 1$. The proposed models are trained for 35 epochs each.

6.2 Results

Fig.7 shows four variations of alphabets 'a', 'e', 'o', and 's' generated using DCGAN, R-cDCGAN, and I-cDCGAN. For comparison purposes, samples from real data are also added. Alphabets generated by DCGAN are poor in quality and there is less variation in generated data as only a few classes of alphabets are generated. R-cDCGAN and I-cDCGAN generated good quality images and results showed variations not only across the classes but also within each class.

Fig. 8a shows three sentences generated using commonly used normal fonts. The same sentences are generated using alphabets synthesized by R-cDCGAN (Fig. 8c) and I-cDCGAN (Fig. 8d). In Fig. 8b sentences are generated using alphabets taken from real data, this gives a visual estimation of the author's handwriting style. Sentences produced by alphabets generated using I-cDCGAN gives a more realistic look than R-cDCGAN as it considered neighboring alphabets along with the current alphabet in the training process.

7 Performance Evaluation

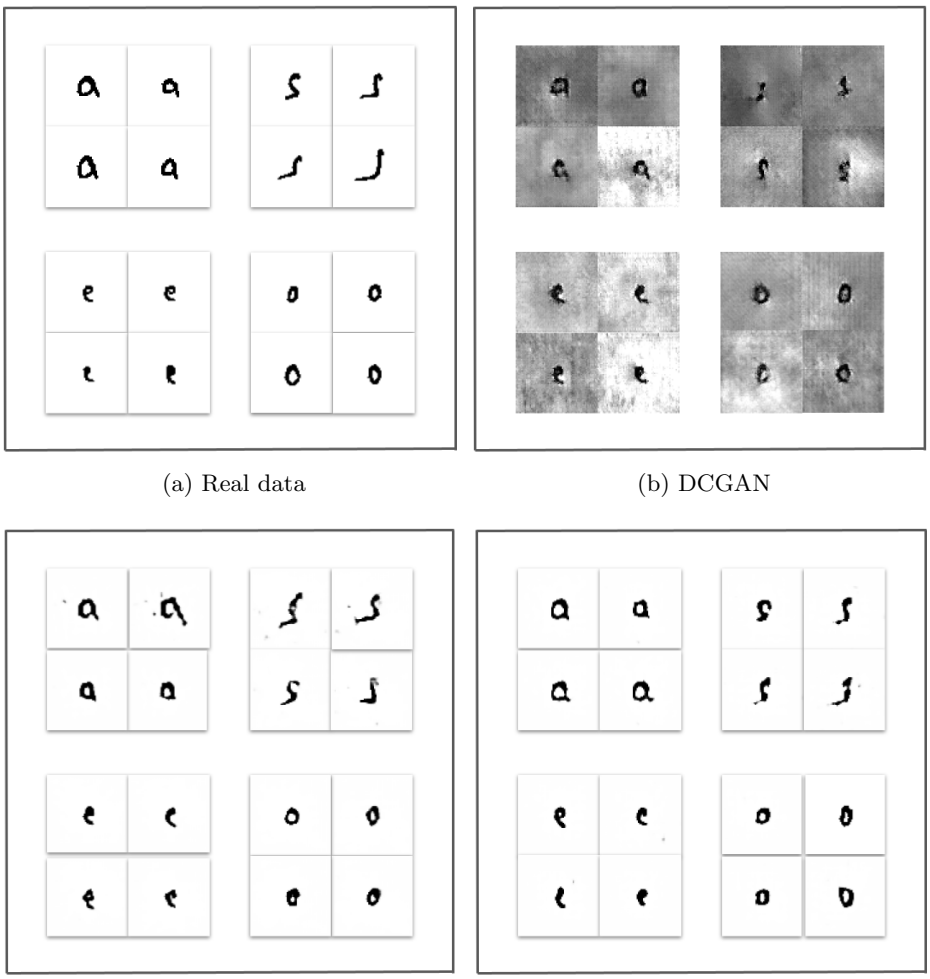
To measure the diversity and quality of synthetic images produced by the generator, Fréchet Inception Distance (FID) [14] score is used. Benny et al. [15] generalized FID score for evaluating generative models in the class-conditional image generation setting. They proposed Within Class FID (WCFID) and Between Class FID (BCFID) for comparing the performances of cGANs. FID, WCFID and BCFID scores are computed to measure the performance of our proposed models.

7.1 Fréchet Inception Distance (FID)

FID score computes resemblance between synthetic (S) data distribution and actual (A) data distribution. The lower FID score indicates that the diversity and quality of synthetic images generated by the generator is more similar to the actual ones. The lower FID score indicates a better GAN model.

If μ^A, Σ^A and μ^S, Σ^S are the centers and covariance matrices of actual data distribution ($\mathcal{D}_A$) and synthetic data distribution ($\mathcal{D}_S$) obtained using a pre-trained feature extractor, then FID score is calculated using (3).

$$FID\left(\mathcal{D}_A, \mathcal{D}_S\right) = \left\|\mu^A - \mu^S\right\|^2 + \mathrm{Tr}\left(\Sigma^A + \Sigma^S - 2\left(\Sigma^A \Sigma^S\right)^{\frac{1}{2}}\right) \tag{3}$$



(a) Real data (b) DCGAN

(c) R-cDCGAN (d) I-cDCGAN

Fig. 7: Alphabets 'a', 'e', 'o' and 's' collected/synthesized using (a) Real data (b) DCGAN (c) R-cDCGAN (d) I-cDCGAN.

7.2 Between Class FID (BCFID)

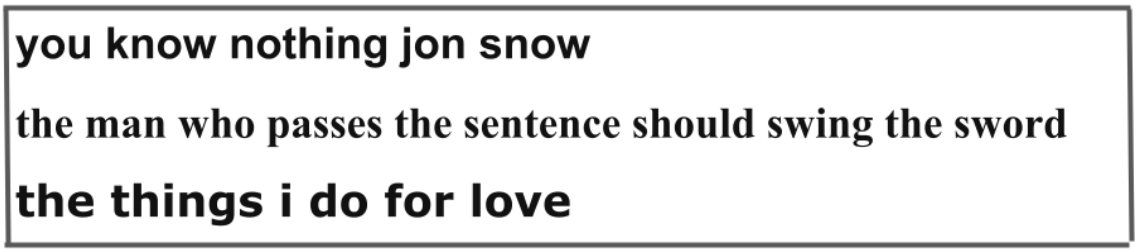
BCFID measures the spread of conditioned classes over actual classes. It is the measure of FID between the distributions of mean feature vector of conditioned classes in synthetic (S) data distribution and mean feature vector of actual (A) classes in actual data distribution. Like FID, the lower BCFID score indicates a better GAN model.

If μ_B^A, Σ_B^A and μ_B^S, Σ_B^S are the mean of per class means and covariance matrices of all per class means of actual data distribution ($\mathcal{D}_A$) and synthetic data distribution ($\mathcal{D}_S$) obtained using a pre-trained feature extractor, then BCFID score is calculated using (4).

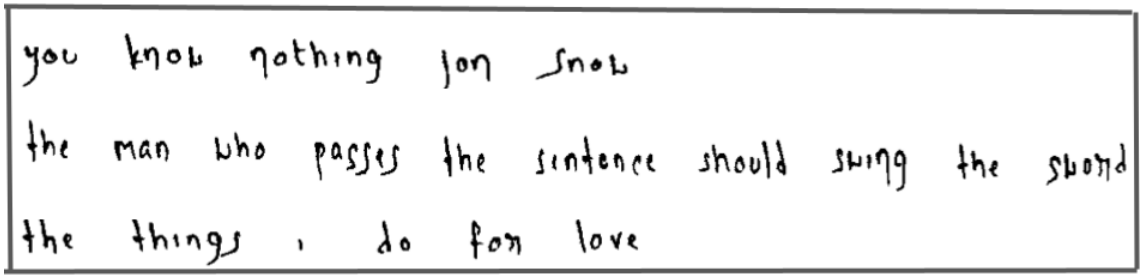
$$BCFID\left(\mathcal{D}_A, \mathcal{D}_S\right) = \|\mu_B^A - \mu_B^S\|^2 + \mathrm{Tr}\left(\Sigma_B^A + \Sigma_B^S - 2\left(\Sigma_B^A \Sigma_B^S\right)^{\frac{1}{2}}\right) \tag{4}$$

7.3 Within Class FID (WCFID)

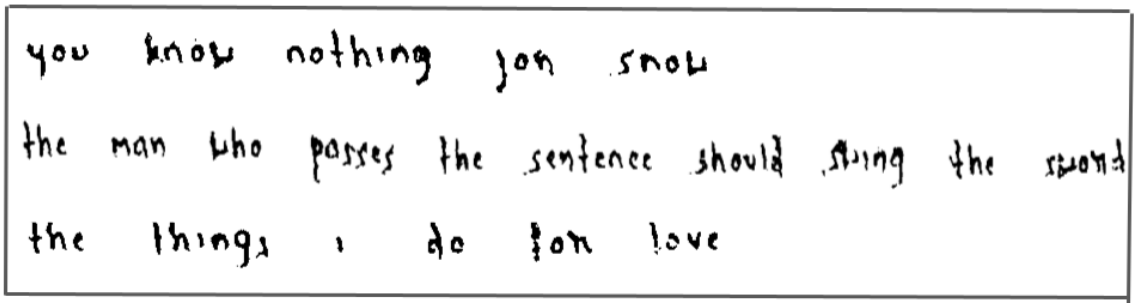
WCFID measures the resemblance between each conditioned class to its corresponding real class. WCFID is the average of all FID scores within all the classes in synthetic (S) data distribution and actual data distribution. The lower WCFID score, similar to FID and BCFID, denotes a superior GAN model.



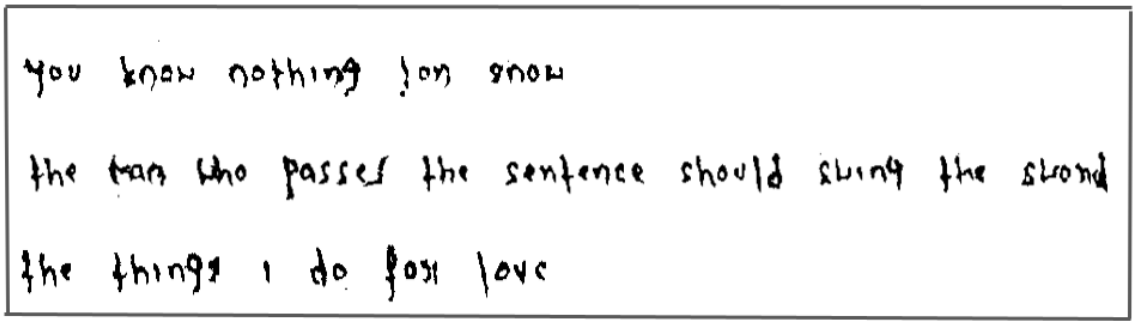
(a) Normal Font



(b) Real data



(c) R-cDCGAN



(d) I-cDCGAN

Fig. 8: Text generated with alphabets collected/synthesized using (a) Normal fonts (b) Real data (c) R-cDCGAN (d) I-cDCGAN.

If μ_c^A, $\Sigma_{W,c}^A$ and μ_c^S, $\Sigma_{W,c}^S$ are within class means and within class covariance matrices of actual data distribution ($\mathcal{D}_A$) and synthetic data distribution ($\mathcal{D}_S$) of class (C) obtained using a pre-trained feature extractor, then WCFID score is calculated using (5).

$$WCFID\left(\mathcal{D}_A, \mathcal{D}_S\right) = \mathbb{E}_{c\sim\mathcal{D}_C}\left[\left\|\mu_c^A - \mu_c^S\right\|^2 + \mathrm{Tr}\left(\Sigma_{W,c}^A + \Sigma_{W,c}^S - 2\left(\Sigma_{W,c}^A\Sigma_{W,c}^S\right)^{\frac{1}{2}}\right)\right] \tag{5}$$

7.4 Experiments

For the feature extractor model, we developed a simple CNN classifier (with 2 convolutional layers and 2 fully connected layers), trained it on IIITA-ALPHABETS data set, and got a test accuracy of 99.35%. This pre-trained CNN classifier is used as a feature extractor to calculate FID score. From real and generated distributions, 50 samples are randomly collected from each class for measuring FID score. The activation of the penultimate layer is employed as the extracted features. The extracted feature dimension is 4096.

In Table 2, real data FID scores are obtained not from synthetic data but from real data itself, two distributions used for FID score calculations are collected from real data itself. Real data FID scores serves as upper bound to evaluate the performance of GANs. Since, DCGAN generated only a few classes of alphabets, so there is high diversity between actual data distribution and synthetic data distribution, resulting in poor FID score.

WCFID score of I-cDCGAN is better than R-cDCGAN because, in real data, the appearance of the alphabet depends on its previous and next alphabet. I-cDCGAN mimics this behavior by considering previous and next alphabet class label along with current alphabet class label as input while training and generating alphabet. So within each class, it produces more diverse data than R-cDCGAN which considers only the current alphabet class label.

BCFID score of R-cDCGAN is better than I-cDCGAN because it is the measure of FID between the distributions of the mean feature vector of conditioned classes in synthetic data distribution and the mean feature vector of actual classes in actual data distribution. As discussed earlier, R-cDCGAN within each class produces less diverse data, but more mean-centric data compared to I-cDCGAN as it considers only the current alphabet class label while training and generating the alphabet.

However, the overall performance of I-cDCGAN is better than R-cDCGAN, as it has a better FID score, indicating that the resemblance between synthetic data distribution and actual data distribution is more in the case of I-cDCGAN than R-cDCGAN.

Table 2: FID, WCFID and BCFID metrics on IIITA-ALPHABETS data set for DCGAN, R-cDCGAN and I-cDCGAN

Model	FID	WCFID	BCFID
Real data	-0.8097	-0.0002064	-9.26e-10
DCGAN	137692	N/A	N/A
R-cDCGAN	46.1970	1.1398	0.0213
I-cDCGAN	43.4696	1.0302	0.0327

8 Conclusion and Future Work

R-cDCGAN and I-cDCGAN are proposed, two-class conditioned versions of DCGAN for generating random font and intelligent font, respectively. The alphabet class is passed in the R-cDCGAN generator to generate random font, whereas the previous, current, and next alphabet classes are passed in the I-cDCGAN generator to generate intelligent font. The experimental results showed that the sentences formed using synthetic fonts generated from proposed models are incorporated with uncertainty and randomness, giving them a more natural depiction than standard fonts. Instead of random TXT documents, lexicons can be created based on the most frequently used words in English to reduce the number of handwritten documents that the author needs to write. Numbers, special characters, and uppercase alphabets can also be added to training data.

Acknowledgment

The present research is partially funded by the I-Hub foundation for Cobotics (Technology Innovation Hub of IIT-Delhi set up by the Department of Science and Technology, Govt. of India).

References

1. Goodfellow, Ian J., Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron C., and Bengio, Yoshua. Generative adversarial nets. NIPS, 2014.
2. Y. Tian, zi2zi: Master Chinese calligraphy with conditional adversarial networks, 2017, https://kaonashi-tyc.github.io/2017/04/06/zi2zi.html. (Accessed 16 April 2019).
3. 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, pp. 1125–1134.
4. A. Odena, C. Olah, J. Shlens, Conditional image synthesis with auxiliary classifier GANs, in:Proceedings of the 34th International Conference on Machine Learning, ICML, 2017, pp. 2642–2651.
5. Y. Taigman, A. Polyak, L. Wolf, Unsupervised cross-domain image generation, arXiv preprint arXiv:1611.02200.
6. J. Chang, Y. Gu, Chinese typography transfer, arXiv preprint arXiv:1707.04904.
7. Ronneberger, O., Fischer, P. and Brox, T., 2015, October. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention (pp. 234-241). Springer, Cham.
8. S. Azadi, M. Fisher, V. Kim, Z. Wang, E. Shechtman, T. Darrell, Multi-content GAN for few-shot font style transfer, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 7564–7573.
9. Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.
10. Y. Guo, Z. Lian, Y. Tang, J. Xiao, Creating new Chinese fonts based on manifold learning and adversarial networks, in: O. Diamanti, A. Vaxman (Eds.), Proceedings of the Eurographics - Short Papers, The Eurographics Association, 2018.
11. X. Lin, J. Li, H. Zeng, R. Ji, Font generation based on least squares conditional generative adversarial nets, Multimedia Tools and Applications (2018) 1–15.
12. Radford, A.; Metz, L.; Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint http:// arxiv. org/ abs/ 1511. 06434, (2015)
13. Kingma, Diederik P and Ba, Jimmy Lei. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.
14. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Advances in neural information processing systems (pp. 6626–6637).
15. Benny, Y., Galanti, T., Benaim, S. et al. Evaluation Metrics for Conditional Image Generation. Int J Comput Vis 129, 1712–1731 (2021). https://doi.org/10.1007/s11263-020-01424-w