Font Creation Using Class Discriminative Deep Convolutional Generative Adversarial Networks

Kotaro Abe*, Brian Kenji Iwana*, Viktor Gösta Holmér†, Seiichi Uchida*
*Department of Advanced Information Technology, Kyushu University, Fukuoka, Japan
†School of Computer Science and Communication, Royal Institute of Technology, Stockholm, Sweden
{abe, brian, uchida}@human.ait.kyushu-u.ac.jp, vholmer@kth.se

Abstract

In this research, we attempt to generate fonts automatically using a modification of a Deep Convolutional Generative Adversarial Network (DCGAN) by introducing class consideration. DCGANs are the application of generative adversarial networks (GAN) which make use of convolutional and deconvolutional layers to generate data through adversarial detection. The conventional GAN is comprised of two neural networks that work in series. Specifically, it approaches an unsupervised method of data generation with the use of a generative network whose output is fed into a second discriminative network. While DCGANs have been successful on natural images, we show its limited ability on font generation due to the high variation of fonts combined with the need of rigid structures of characters. We propose a class discriminative DCGAN which uses a classification network to work alongside the discriminative network to refine the generative network. This results of our experiment shows a dramatic improvement over the conventional DCGAN.

1. Introduction

There is a wide variety of fonts which come with various styles and appearances. Fonts can have different weights, spacing, slopes, and proportions. There are also handwritten fonts, fonts made from outlines, decorated fonts, scripts, serif fonts, etc. The thousands of different fonts are valuable tools for document designers to portray messages in text. However, font creation is a manual process.

To remedy this, we explore generating fonts a Deep Convolutional Generative Adversarial Network (DCGAN) [7]. DCGANs use convolutional and deconvolutional layers to apply Generative Adversarial Networks (GAN) to the domain of images. Traditionally, DCGANs are unsupervised neural networks for generating images using adversarial training between a generative network and a discriminative network. The purpose of the discriminative network is to detect whether image samples are *generated* or *real*. Meanwhile, the generative network competes against the discriminative network to create undetectable images. This competition between the two networks continues until the generated images are indistinguishable to the discriminative network.

In an attempt to move towards font generation, the contributions of this paper is two-fold. First, we demonstrate the limited ability of DCGANs on text generation. Printed text has distinct classes, however, the discriminative network only considers if the generated fonts have text-like features and not actual character formulation. By including class consideration in the DCGAN architecture, we can force the generative network to produce more robust fonts. The second contribution is the proposal of a class discriminate DCGAN which introduces a classification network to guide the generative network. The results of our experiments show that the inclusion of the classification network has a considerable improvement on the generation ability of DCGAN.

The remaining of this paper is organized as follows. In Section 2, we discuss the related work in font generation and the state of DCGANs in literature. Section 3 details the DCGAN model and Section 4 reports our initial results. In Section 5, we propose an improvement to the conventional DCGAN by introducing a classification network into the architecture. We then demonstrate the effectiveness of the class discriminative DCGAN. Finally, Section 7 draws the conclusion and future work.

2. Related Work

Many attempts in automatic font generation have been made in the past. Some methods create fonts based on examples of manually inputed characters using predictive features [9, 11]. Another approach is to create mappings based on the shape of multiple fonts and generating interpolation between them [1, 12].

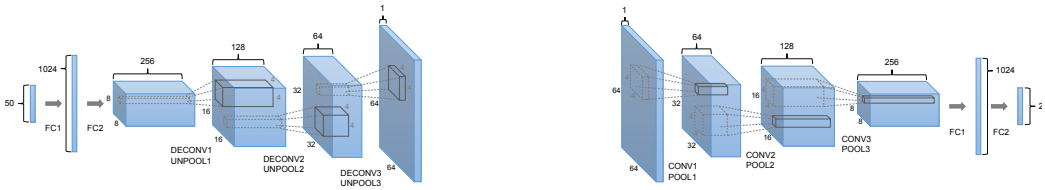


Figure 1. Topology of (left) the generative network G and (right) the discriminative network D used in the experiments.

The core concept that the proposed model is based on is the use of a GAN [2]. DCGANs [7] are a recent improvement on the original GAN that primarily deals with natural images. However, even with the successes of DCGANs, the generated natural images are typically low resolution and sometimes contain nonsensical or misplaced features. For natural images, this does not cause problems but font generation requires rigid structures to be considered valid characters. Categorical GAN (CatGAN) [8] is a class discriminating GAN that solves this by replacing the discriminative network with a classification network. The distinction between CatGAN and the proposed method maintains both the discriminative network and a second classification network.

The principles of DCGANs are described in detail in Section 3, however at its core, DCGANs rely on an underlying Convolutional Neural Network (CNN) structure. While CNNs boast high accuracies in character recognition, they are flaws. In literature [10, 3], adversarial examples are presented to CNNs and by merely adding slight noise, they can be fooled. Also, there are examples [6] of images with no human recognizable objects, but still classified with high probabilities by CNNs. Considering these examples, it is thought that generating ideal images for human beings by machine learning is a difficult task. Humans and CNNs emphasize different features and have different methods of perception, thus they have dramatically different ways of image recognition.

3. Deep Convolutional Generative Adversarial Network

3.1. Definition

DCGAN [7] is an image generation model composed of two neural networks, a generative network G and a discriminative network D. Figure 1 is an illustration of the G and D networks. The G network is a deconvolutional neural network that generates images from d-dimensional vectors using deconvolutional layers and unpooling. On the other side, a D network has the same structure as a conventional CNN that discriminates whether the input is a real image from a predefined dataset or whether it is an image generated by G.

Specifically, Fig. 2 demonstrates the learning process of a DCGAN. It is performed by repeating the procedure in Fig. 2. First, the training of the D network is performed using images from the dataset x. Next, the generative network produces generated input images $G(z)$ from an random vector z. Finally, the discriminative D network is updated from the generated image. The idea of this process is that over repeated iterations, the training of G will produce images more and more indistinguishable from images from the dataset.

The training of DCGAN is expressed as follows [2]:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\log D(x)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log(1-D(G(z)))\right], \quad (1)$$

where x is the original image, z is a d-dimensional vector consisting of random numbers, and $p_{\text{data}}(x)$ and $p_z(z)$ are the probability distributions of x and z respectively. $D(x)$ is the probability of the input being a generated image from $p_{\text{data}}(x)$ and $1-D(G(z))$ is the probability of being generated from $p_z(z)$. D is trained to maximize the correct answer rate and G is trained to minimize $\log(1-D(G(z)))$ in order to deceive D.

3.2. Discriminative Network

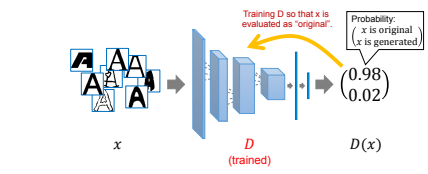
The discriminative network D determines whether the given image is an original image or an image generated by G. The structure of D is similar to a conventional CNN which consists of convolutional layers, pooling layers, and fully-connected layers. D is trained by the following two procedures.

First, training is done using original images x from the dataset as input patterns. This is illustrated in Fig. 2 (a). The ground truth of this training step for each x is set to "original" and the cross entropy error for $D(x)$ is calculated. Subsequently, D applies backpropagation to train the network.

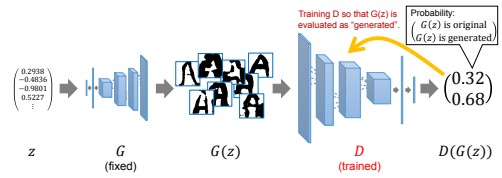
Next, as shown in Fig. 2 (b), D is given an image generated by G with the ground truth set as "generated." The cross entropy error for $D(G(z))$ is then calculated and backpropagated through D.

3.3. Generative Network

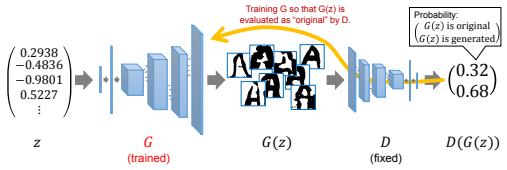
A generative network generates images from a given d-dimensional vector$z$. The network uses a combination of



(a)



(b)



(c)

Figure 2. The training process of a DCGAN. (a) is the training of D using real images x from the dataset. (b) is a second training iteration of D, however this time using the generated images as the input. (c) is the training of G using the evaluation of the generated images $G(z)$ in the network D given a random vector z.

deconvolutional layers and unpooling layers to extrapolate the provided vector into an image. During the training of G, the vector z consists of random variables of a uniform distribution within a predetermined range. Furthermore, shown in Fig. 2 (c), G learns from the evaluation made by D on the generated image $G(z)$. Unlike the training of D, this training step designates the ground truth of $G(z)$ as "original" in an attempt to deceive D and to have G learn from the cross entropy of $D(G(z))$ through backpropagation. Also, during this training step of G, the parameters of D are fixed.

3.4. Duplicate Removal

To reduce the chance of overfitting, we reduce the dataset by removing similar fonts before training. Radford et al. [7] showed the value of duplicate removal by removing images with a de-noising AutoEncoder.

4. Font Generation Experiment by DCGAN

4.1. Dataset

The dataset was created using the 26 uppercase alphabet letters from 6,628 different fonts. The characters were rendered into 64px by 64px binary images with size normalization. From the dataset, duplication removal (see Section 3.4) was performed reducing the dataset to 200 unique fonts. Also, we ensured that no outline fonts or lowercase fonts were used. Figure 3 shows a sample letter "A" from each of the fonts used as the x set for the D network.

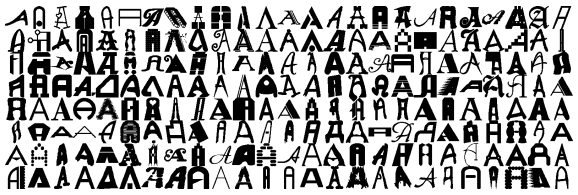


Figure 3. The letter "A" in the 200 fonts selected by the duplicate removal process.

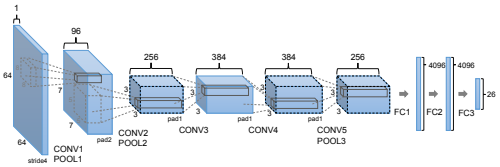


Figure 4. The classification network used in the experiments

4.2. Real Character Recognition

Before the generation experiments, we conducted a character recognition experiment with our database to determine how much could be recognized by a CNN. The purpose of the character recognition experiment is to compare the results to the recognition rate of the generated font in order to ascertain the quantity and quality of the generated fonts. Figure 4 illustrates the classification network used for the experiment. It should be noted that the classification network is a separate network from the other two networks. The classification network is a modification of AlexNet [5], made to match our the experiment's image size. Using a dataset split of 90% training and 10% test, the classification experiment had a recognition rate of 92.14%.

4.3. DCGAN Architecture

Figure 1 details the structure and the hyperparameters of the G and D networks. The size of the d-dimensional input vector of G was set to 50 and the output is a deconvoluded 64px by 64px image. The D network performs the opposite and takes in the 64px $\times$ 64 image and outputs the probability of the input being an "original" image or a "generated" one.

In literature [7], a Rectified Linear Unit (ReLU) or LeakyReLU activation function is used, however for the experiment we used a hyperbolic tangent activation function. This is because characters only need to consider binary im-

"A" "B" "C"

Figure 5. 25 example characters generated by DCGAN after 5,000 training iterations.

ages and do not need to consider the degree of shading. In addition to using Adam optimizer [4] with the same parameters to [7], the learning rate was set to decay by 0.001 after each iteration. We trained the network with batch sizes of 100 for 10,000 iterations.

4.4. Generation Results

The result of font generation is shown in Fig. 5. However, many of them are difficult to recognize as characters. This is because during the training of D, discrimination is only done on whether the images are "original" or "generated," and G only learns to correct the error. As a result, the network only learns objects that contain scattered features, making for disfigured fonts. Also, due to the nature of characters, even if the shape is slightly distorted, the readability becomes very low.

Figure 6 shows the training process of fonts for the character "A." In the early stages, there are many fonts with black backgrounds and white characters, despite only using input images with white backgrounds and black characters. This indicates that D uses part of the early stages to learn the white-black boundary instead of actual font color. Through the process, those characters gradually invert themselves to the proper style. Little change can be seen between iteration 8,000 to 10,000 due to the considerably smaller learning rate at that stage.

4.5. Impact of Duplicate Removal

The duplicate removal process had a dramatic effect on the generated fonts. Figure 7 is the result of using the entire set of 6,628 fonts without duplicate removal. From the figure, the results did not learn properly and is unrecognizable as letters. The reason for this is the high variation causes D to struggle distinguishing between real and fake.

4.6. Generated Character Recognition Results

To test the generated fonts, we performed a recognition experiment. As test patterns, a total of 10,000 generated fonts were prepared, each consisting of 2,500 characters from "A," "B," "C," and "D." The experiment used the classification network described in Section 4.2. As a result, the experiment obtained a recognition rate of 48.15%. The results were poor due to having many disfigured or ambiguous characters.

5. Class Discriminative DCGAN

The problem with generating fonts with DCGAN is that the readability is not actively considered. In other words, the G and D networks only care about recognizing the difference between "original" and "generated." D only discriminates based on the images containing features of characters and not the actual classes of the characters. Therefore, we propose a DCGAN which considers interclass discrimination in order to generate more structurally sound fonts.

5.1. Introducing a Classification Network to DCGAN

In order to apply class discrimination to the DCGAN, we introduce the same classification network Section 4.2 denoted as C. With the introduction of C, the process flow is changed to,

1. D is trained from "original" fonts x (Fig. 2 (a)).
2. G generates images $G(z)$ and they are used to further train D (Fig. 2 (b)).
3. G is updated from the results of D (Fig. 2 (c)).
4. C classifies the generated image $G(z)$ and G learns based on the evaluation (Fig. 8).

The cross entropy error for $C(G(z))$ is computed with the correct label of $G(z)$ and in turn G is trained through backpropagation. During this time, backpropagation is calculated through C, but the parameters of C are not changed. This new process flow creates generated images more reflective of class differentiation.

5.2. Classification Network Influence

Simply adding the full gradient of C puts too much emphasis on class discrimination and the DCGAN loses some of the adversarial aspects from D. We can control this by adjusting the rate that the gradient of C effects G. The training of the influence controlled class discriminative DCGAN is expressed as:

$$\begin{aligned}\min_G \max_D V(D,G) = &\mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\log D(x)\right] \\ &+ (1-\alpha)\mathbb{E}_{z\sim p_z(z)}\left[\log(1-D(G(z)))\right] \\ &+ \alpha\mathbb{E}_{z\sim p_z(z)}\left[\log(C_k(G(z)))\right] \qquad (2)\end{aligned}$$

where C_k is a function that returns the probability that the given image is of class k and where α represents the rate that the gradient affects G. In other words, the proposed improved DCGAN trains G by learning to maximize $\log(C_k(G(z)))$.

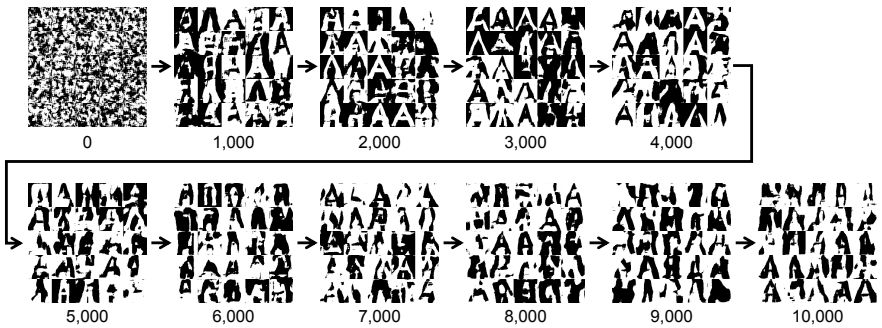


Figure 6. Examples of the letter "A" generated by DCGAN over the course of training. The numbers indicate the training iteration.

Figure 7. The fonts for the letter "A," generated after 5,000 training iterations without duplicate removal.

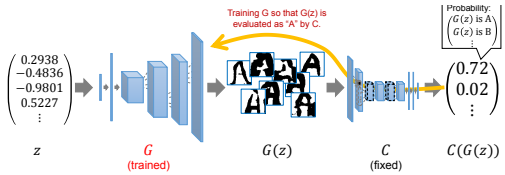


Figure 8. Training of the classification network C integrated into a DCGAN.

6. Font Generation Experiment by Class Discriminative DCGAN

6.1. Class Discriminative DCGAN Architecture

The class discriminative DCGAN uses the same optimization methods and hyperparameters for the D and G networks defined in Section 4.3. The C network classifies the 26 alphabet classes using the same configuration as Section 4.2, which was trained on the full 6,629 fonts. In addition, the experiment uses the classification network gradient influence rate of $\alpha = 0.01$.

6.2. Generation Results

Figure 9 is the result of the font generation with the class discriminative DCGAN for four example classes. When compared to Fig. 5, the results of the proposed method are much more clear and character-like. Moreover, while similar in structure, the generated fonts are distinct with unique details and features. By comparing the previous process in Fig. 6 to the class discriminative process in Fig. 10, the results show a clear improvement. After about 5,000 training iterations, the fonts become formalized characters, and the subsequent iterations do not improve significantly.

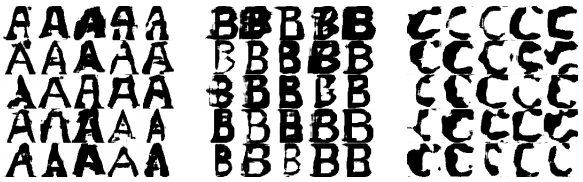


Figure 9. Fonts generated by the class discriminative DCGAN when $\alpha = 0.01$.

6.3. Generated Character Recognition Results

In order to determine the qualitative effect of C on the DCGAN, we compared the recognition results of the conventional DCGAN to the class discriminate DCGAN. A similar trial to Section 4.6 was performed with 10,000 generated fonts consisting of "A"s, "B"s, "C"s, and "D"s. We achieved the much higher recognition rate of 98.95% compared to the previous evaluation of 48.15% from the conventional DCGAN. The recognition rate is much higher because C incorporates class distinction and class recognizability into the learning process. This means that the network produces more realized characters with higher readability.

6.4. Effect of Classification Network Influence Rate

It is important to study the effect of modifying the influence rate from the gradient of C. We trained multiple networks at different values of α. Figure 11 reveals the results. When $\alpha = 0.1$, the influence of the gradient of C has on the network overpowers the effect of the discriminative network D. However, when α is too small, such as 0.005, the benefits of C are suppressed. We find that the most effective rate of α are between 0.05 and 0.01.

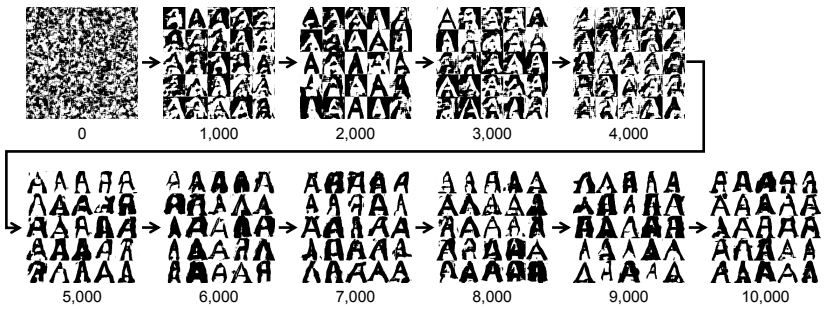


Figure 10. Examples of the letter "A" generated by the class discriminative DCGAN through the training. The numbers indicate the training iteration.

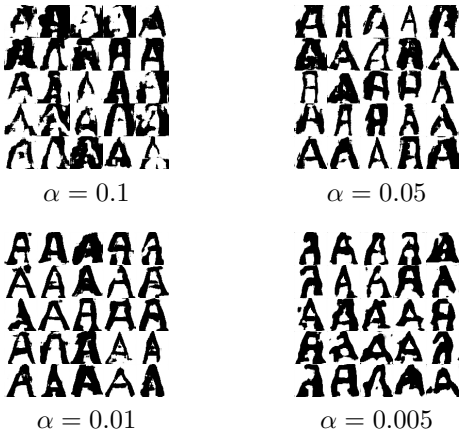


Figure 11. Generated "A"s at different influence rates α of the gradient of C.

7. Conclusion

In this paper, we attempted automatic font generation using a DCGAN. Experiments with a conventional DCGAN produces sub-par results. We propose a method of incorporating a classification network in the DCGAN architecture, resulting in a class discriminative DCGAN. Using the class discriminative DCGAN, we confirm the improved effectiveness of the proposed method over a conventional DCGAN for text generation.

However, is still room for improvement. Future improvements to the network include: optimization of the discriminative network, creating a more robust generative network, and setting criteria for stopping the learning process. Another future task will be to extend the class discrimination to being able to distinguish a non-character class in order to produce more robust characters.

Acknowledgments

This research was partially supported by MEXT-Japan (Grant No. 26240024 and 17H06100) and Kakihara Foundation.

References

[1] N. D. Campbell and J. Kautz. Learning a manifold of fonts. *ACM Trans. Graphics*, 33(4):91, 2014. 1

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Inform. Process. Syst.*, pages 2672–2680, 2014. 2

[3] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 2

[4] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Inform. Process. Syst.*, pages 1097–1105. Curran Associates, Inc., 2012. 3

[6] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, pages 427–436, 2015. 2

[7] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Int. Conf. Learning Representations*, 2016. 1, 2, 3, 4

[8] J. T. Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015. 2

[9] R. Suveeranont and T. Igarashi. Example-based automatic font generation. In *Int. Symp. Smart Graphics*, pages 127–138. Springer, 2010. 1

[10] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 2

[11] T. Tsuchiya, T. Miyazaki, Y. Sugaya, and S. Omachi. Automatic generation of kanji fonts from sample designs. In *Tohoku-Section Joint Conv. of Inst. of Elect. and Inform. Engineers*, 2014. 1

[12] S. Uchida, Y. Egashira, and K. Sato. Exploring the world of fonts for discovering the most standard fonts and the missing fonts. *Int. Conf. Document Anal. and Recognition*, 2015. 1