DG-Font: Deformable Generative Networks for Unsupervised Font Generation

Yangchen Xie Xinyuan Chen* Li Sun Yue Lu

Shanghai Key Laboratory of Multidimensional Information Processing,
East China Normal University, 200241 Shanghai, China

ycxie0702@gmail.com xychen@cee.ecnu.edu.cn sunli@ee.ecnu.edu.cn ylu@cs.ecnu.edu.cn

Appendix

A. Implementation Detail

A.1 Training Strategy

We initial the weights of convolutional layers with He initialization [1], in which all biases are set to zero and the weights of linear layers are sampled from $N(0, 0.01)$. We use Adam optimizer with β_1 = 0.9 and β_2 = 0.99 for style encoder, and RMSprop optimizer with α = 0.99 for the content encoder and mixer. We train the whole framework 200K iterations and the learning rate is set to 0.0001 with a weight decay 0.0001. We train the model with a hinge version adversarial loss [2, 4] with R1 regularization [3] using $\gamma = 10$. In all experiments, we use the following hyper-parameters: λ_{img} = 0.1, λ_{cnt} = 0.1, λ_{offset} = 0.5. All the images are resized to 80×80 before training and testing and the batch size is set to 32. In testing process, we use ten reference images to compute an average style code for the generation process. The source code will be released after the paper is accepted.

A.2 Network Architecture

The proposed model is an encoder-decoder network. The style encoder whose architecture is based on VGG-11 aims to extract style information. The architecture of the content encoder and mixer are symmetrical, which help preserve domain-invariant information of content. The detailed architectures of style encoder, content encoder, and mix are shown in Table 3. The detailed information of discriminator is listed in Table 4.

B. Ablation Study

We evaluate our model by setting different hyper-parameter values in overall objective loss (Eq. 8). All the metrics are computed based on the 400 seen fonts mentioned in Sec 4.1.

	L1 loss	RMSE	SSIM	LPIPS	FID
N = 1	0.0563	0.2005	0.7549	0.850	47.99
N = 2	**0.0562**	**0.1994**	**0.7580**	**0.0814**	**46.15**
N = 3	0.0580	0.2039	0.7514	0.0832	47.12

Table 1. **The impact of the number of FDSC module**. N denotes the number of FDSC module.

Content reconstruction loss: We analyze the impact of the content reconstruction loss in Table 2. We observe that a large λ_{cnt} value leads to degradation, while the model begins even worse without the content reconstruction loss. It show that λ_{cnt} = 0.5 provides a good trade-off.

Image reconstruction loss: As shown in Table 2, we find that our method performs well when λ_{img} = 0.1. The image reconstruction loss helps preserve domain-invariant characteristics of the content input, but a large λ_{img} makes the model pay more attention to reconstruct input images and perform poorly on generating new characters.

Offset normalization: Table 2 presents results of loss term λ_{offset} in deformable offset normalization (Eq. 7). As demonstrated in the table, λ_{offset} = 0.5 significantly improve the generation quality.

Influence of the number of FDSC module: to evaluate the influence of the number of FDSC module on model performance, we conduct experiments that adding FDSC module transfer different levels of information from low-level to high level sequentially. From Table 1, we can observe that adding an FDSC module to transfer the deformed low-level feature significantly improve the performance of the model. However, when the second FDSC module is added to transfer the higher-level information, the improvement is not obvious. This may be because feature maps input to the second FDSC contains less spatial information. The third FDSC even reduces performance. The detailed results are shown in Table 1.

*Corresponding author: xychen@cee.ecnu.edu.cn

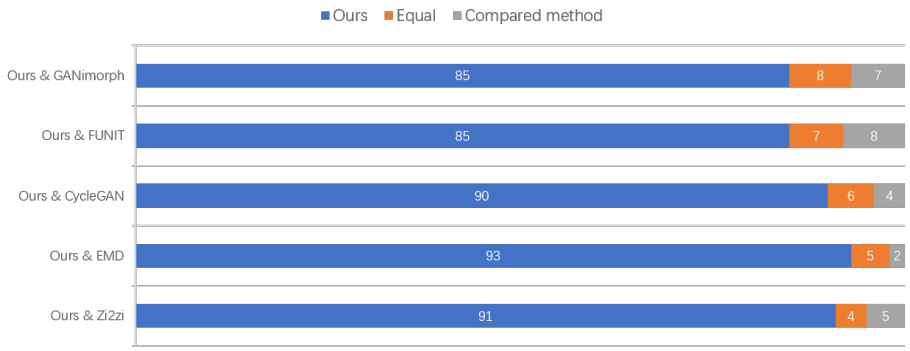

Figure 1. **Results of user study**. For each compared method, we randomly sample 100 generated images for participants' preferences test. The blue bar indicates the number of images that more participants prefer our results. The gray bar indicates the number of images that more participants prefer results of the compared methods. The orange bar indicates the number of images where two methods get an equal number of votes from 10 participants.

C. User Study

For further comparing the quality of images generated with other methods, we conduct an experiment on a human study by pairwise A/B tests. There are four tasks and for each task, we randomly select 100 characters from the test set to make 100 paired images generated by DG-Font and another compared method. The participants are 10 people who use Chinese characters every day. The participates are asked to select a more similar image compared to ground-truth from each pair within seven seconds. For each pair, we choose the image with more votes as the judgment result. Fig. 1 shows the participants' preference among the four tasks. We observe that more than 85 results of our methods outperform the results of compared methods.

D. More Results

We select twenty-four fonts including calligraphy and handwriting to prove the superiority of our method. These fonts present different styles in geometric transformation, stroke thickness, tips, and joined-up writing patterns. The results show that the proposed method outputs high-quality characters.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1026–1034. IEEE Computer Society, 2015.

[2] Jae Hyun Lim and Jong Chul Ye. Geometric GAN. *CoRR*, abs/1705.02894, 2017.

[3] Lars M. Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3478–3487. PMLR, 2018.

[4] Dustin Tran, Rajesh Ranganath, and David M. Blei. Deep and hierarchical implicit models. *CoRR*, abs/1702.08896, 2017.

	L1 loss	RMSE	SSIM	LPIPS	FID
$\lambda_{cnt} = 0$	0.0592	0.2065	0.7441	0.0874	52.91
$\lambda_{cnt} = 0.1$	**0.0562**	**0.1994**	**0.7580**	**0.0814**	**46.15**
$\lambda_{cnt} = 1$	0.0591	0.2068	0.7456	0.0849	46.60
$\lambda_{img} = 0$	0.0593	0.2065	0.7439	0.0860	59.29
$\lambda_{img} = 0.1$	**0.0562**	**0.1994**	**0.7580**	**0.0814**	46.15
$\lambda_{img} = 1$	0.0627	0.2155	0.7340	0.0908	**44.86**
$\lambda_{offset} = 0$	0.0568	0.2007	0.7532	0.0870	52.60
$\lambda_{offset} = 0.5$	**0.0562**	**0.1994**	**0.7580**	**0.0814**	**46.15**
$\lambda_{offset} = 1$	0.0569	0.2006	0.7538	0.0838	50.23

Table 2. **Impact of the hyper-parameters**.

	Operation	Kernel size	Resample	Padding	Feature maps	Normalization	Nonlinearity
Style encoder	Convolution	3	MaxPool	1	64	BN	ReLU
	Convolution	3	MaxPool	1	128	BN	ReLU
	Convolution	3	-	1	256	BN	ReLU
	Convolution	3	MaxPool	1	256	BN	ReLU
	Convolution	3	-	1	512	BN	ReLU
	Convolution	3	MaxPool	1	512	BN	ReLU
	Convolution	3	-	1	512	BN	ReLU
	Convolution	3	MaxPool	1	512	BN	ReLU
	Average pooling	-	-	-	128	-	-
	FC	-	-	-	128	-	-
Content encoder	Deformable conv	7	-	3	64	IN	ReLU
	Deformable conv	4	stride-2	1	128	IN	ReLU
	Deformable conv	4	stride-2	1	256	IN	ReLU
	Residual block $\times 2$	3	-	1	256	IN	ReLU
Mixer	Residual block $\times 2$	3	-	1	256	AdaIN	ReLU
	Convolution	5	Upsample	2	128	AdaIN	ReLU
	Convolution	5	Upsample	2	64	AdaIN	ReLU
	Convolution	7	-	3	3	-	tanh

Table 3. **Generative network architecture**. BN, IN, AdaIN denote the batch normalization, Instance normalization, and Adaptive instance normalization, respectively. FC means the fully connected layer

	Operation	Kernel size	Resample	Padding	Feature maps	Normalization	Nonlinearity
Discriminator	Convolution	3	-	1	64	-	-
	Residual block	3	-	1	64	FRN	-
	Residual block	3	AvgPool	1	128	FRN	-
	Residual block	3	-	1	128	FRN	-
	Residual block	3	AvgPool	1	256	FRN	-
	Residual block	3	-	1	256	FRN	-
	Residual block	3	AvgPool	1	512	FRN	-
	Residual block	3	-	1	512	FRN	-
	Residual block	3	AvgPool	1	1024	FRN	LeakyReLU
	Convolution	4	-	1	1024	-	LeakyRuLU
	Convolution	1	AvgPool	1	400	-	-

Table 4. **Discriminator architecture**. AvgPool denotes the averge pooling. The slope of LeakyRuLU is set to 0.2.