CF-Font: Content Fusion for Few-shot Font Generation
Supplementary Material

Chi Wang[1,2]*, Min Zhou[2], Tiezheng Ge[2], Yuning Jiang[2], Hujun Bao[1], Weiwei Xu[1]†
[1] State Key Lab of CAD&CG, Zhejiang University [2] Alibaba Group
`wangchi1995@zju.edu.cn, {yunqi.zm, tiezheng.gtz, mengzhu.jyn}@alibaba-inc.com`
`{bao, xww}@cad.zju.edu.cn`

This supplementary material contains more results and a discussion of potential negative societal impacts, which are not included in the main paper due to space limitation. We also provide a video clip to demonstrate the font style interpolation effect. In addition, the code will be uploaded to https://github.com/wangchi95/CF-Font for reproduction of the results in our paper.

Code for Reproduction. We also provide the code needed to reproduce the main experimental results in our main paper. Our code is based on the official code repository of DG-Font [2] (https://github.com/ecnuycxie/DG-Font).

Visualization of Basis. Fig. 1 provides an illustration of the cluster-based selection method (in Subsection. 3.2 of the paper). We plot images of the example character "Tong" from 240 fonts and highlight images from the automatically selected basis fonts with red boxes. The visualizations show that our basis fonts are quite different in content from each other, which is beneficial for our content fusion module (CFM).

Visualization of CF-Font on the full GB2312 standard. In the *Demo* folder, we show some results generated by our network (with 16 reference images) on challenging fonts and challenging characters. The results include 8 demos, each containing 16 reference character images and some corresponding generated characters, which form a short essay and two poems. In order to better show the effect, we also provide the corresponding TrueType Font File (generated with the output 6763 characters of our network). The TrueType Font File is obtained by converting bitmap images to vector images, so there may be slight differences caused by vectorization. The generated TrueType Font File works well with Word/Excel/PPT/Sketch and other apps on mac OS, but only works with system font viewer on windows (it can be viewed with FontForge or Glyphs).

Detailed Qualitative Comparison. In Figs.2-14, we provide a more detailed (Relative to Fig. 5 and 6 in the paper) qualitative comparisons on both seen and unseen fonts. For methods without CFM, we plot all generated results with the font "Kai" and all basis fonts as the source. The results show that the generated images are closely related to the source font, and not all fonts are suitable as a source for the few-shot font generation task. Methods with "Kai" (the first row in each subfigure) and "Song" (the third-to-last row in each subfigure) as source fonts produce relatively better results than with other fonts, but often fail on skeleton transfer. In contrast, our method stably produces high-quality results, both in terms of style and content.

Network Efficiency. We evaluate the efficiency of CF-Font in terms of speed, computation, and the number of parameters with batch size set as one. The speed is an average value of 200 experiments with random inputs. As Tbl. 1 shows, the computation significantly increases (246%) from DG-Font to CF-Font, since the fused content features in CFM are based on ten basis fonts. But the FPS drop is smaller (8.8%), for CFM can be highly parallelized on GPUs. The FPS drop is further reduced to only 1.7% during training, for the reason that the content encoder is fixed and does not need back-propagation to update parameters. Also, the increase in the number of parameters for CF-Font compared to DG-Font is almost negligible (17.1M→17.1M+10).

Use PC-WDL to find the basis fonts and calculate fusion weights. Since PC-WDL can also measure the similarity of skeletons through the observation of Fig. 5 in the paper, we design an experiment to further explore the effectiveness of PCL. We evaluate the PC-WDL based CF-Font, which use PC-WDL to find the basis fonts and calculate the content fusion weights, on the unseen fonts, and obtain 0.07432, 0.2355, 0.6999, 0.1162, and 25.28 on the L1, RMSE, SSIM, LPIPS, and FID metrics, which is closely comparable to the original CF-Font (0.07394, 0.2354, 0.7007, 0.1182, and 26.51 on these metrics). We believe both methods are feasible and we can choose either one in practice.

Evaluation of one-hot content features. Font generation is a variant of image generation tasks. It has some unique characteristics, such as sharp edges, a huge number of categories, style transfer on both skeleton (or structure) and stroke, etc. Taking one-hot class vectors as the content inputs is not a good choice for font generation, since it is tedious for one-hot encoding to represent thousands of characters. In addition, during training, all one-hot class vectors have to be input for the network for the synthesis of the corresponding characters.

To reveal the ability of font generation with one-hot class vectors, we design a content-id-based network. Given a reference image and a target character id, the model should output an image of the corresponding character with the same style as the reference image. To complete this task, we modify the architecture and supervision of the classical Zi2zi model [1]. In detail, we change the training data pairs (from content image and style image to style image and content embedding extracted from character ID) and remove the long-distance skip connections which may confuse the skeleton structure.

We train the model on 20 characters for 200 epochs and the dataset is generated by sampling 1000 pairs per character among 240 fonts (20000 pairs in total). The results are shown in the *part B* of the PDF file *CID-based Net.pdf* (in the folder *others*), which indicates that it is hard for the content-id-based network to handle these only 20 characters on seen or unseen fonts, and the output images are noisy. We also experiment with the content-id-based network on 50 characters, the results in the *part C* of the PDF file *CID-based Net.pdf* (in the folder *others*) indicate the same conclusion.

Dataset. We obtain fonts from the following font platforms under a personal non-commercial academic research license: 1. Foundertype (https://www.foundertype.com) 2. Makefont (https://www.makefont.com) 3. Hanyi (http://www.hanyi.com.cn) 4. Tsanger (http://www.tsanger.cn) 5. 17Font https://www.17font.com/) 6. Tensentype (http://www.tensentype.com) 7. Hellofont (https://www.hellofont.cn)

Potential Negative Societal Impacts. Since our method can generate an entire font library with consistent style from a few reference characters, it can be abused to forge personal signatures and handwritten documents.

References

[1] Yuchen Tian. Zi2zi. https://github.com/kaonashi-tyc/zi2zi. 2

[2] Yangchen Xie, Xinyuan Chen, Li Sun, and Yue Lu. Dg-font: Deformable generative networks for unsupervised font generation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5130–5140. Computer Vision Foundation / IEEE, 2021. 1

Table 1. The efficiency of CF-Font compared to DG-Font.

Method	Mult-Adds (G)	Params (M)	FPS[1]	Training FPS[2]
DG-Font	7.44	17.1	66.0	5.9
CF-Font	18.27	17.1	60.2	5.8

[1] There is only one forward pass for the generator and discriminator.

[2] There are more than one forward-backward pass for both the generator and discriminator in alternating periods for GAN training.

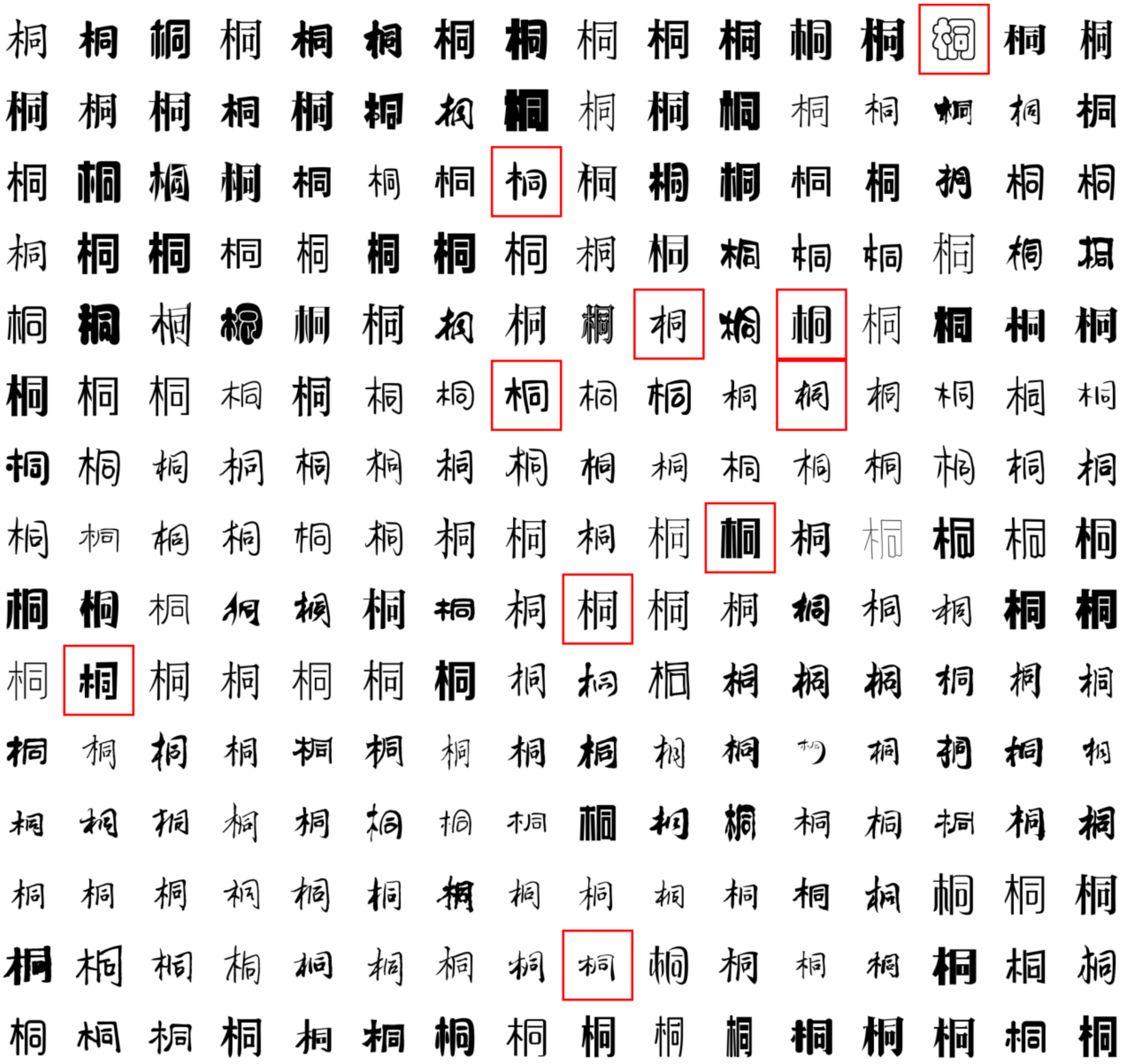

Figure 1. Visualization of basis, taking the character “Tong” of all fonts from the training set as an example. The images bounded by red boxes are from the basis fonts, which are automatically selected by our cluster-based selection method.

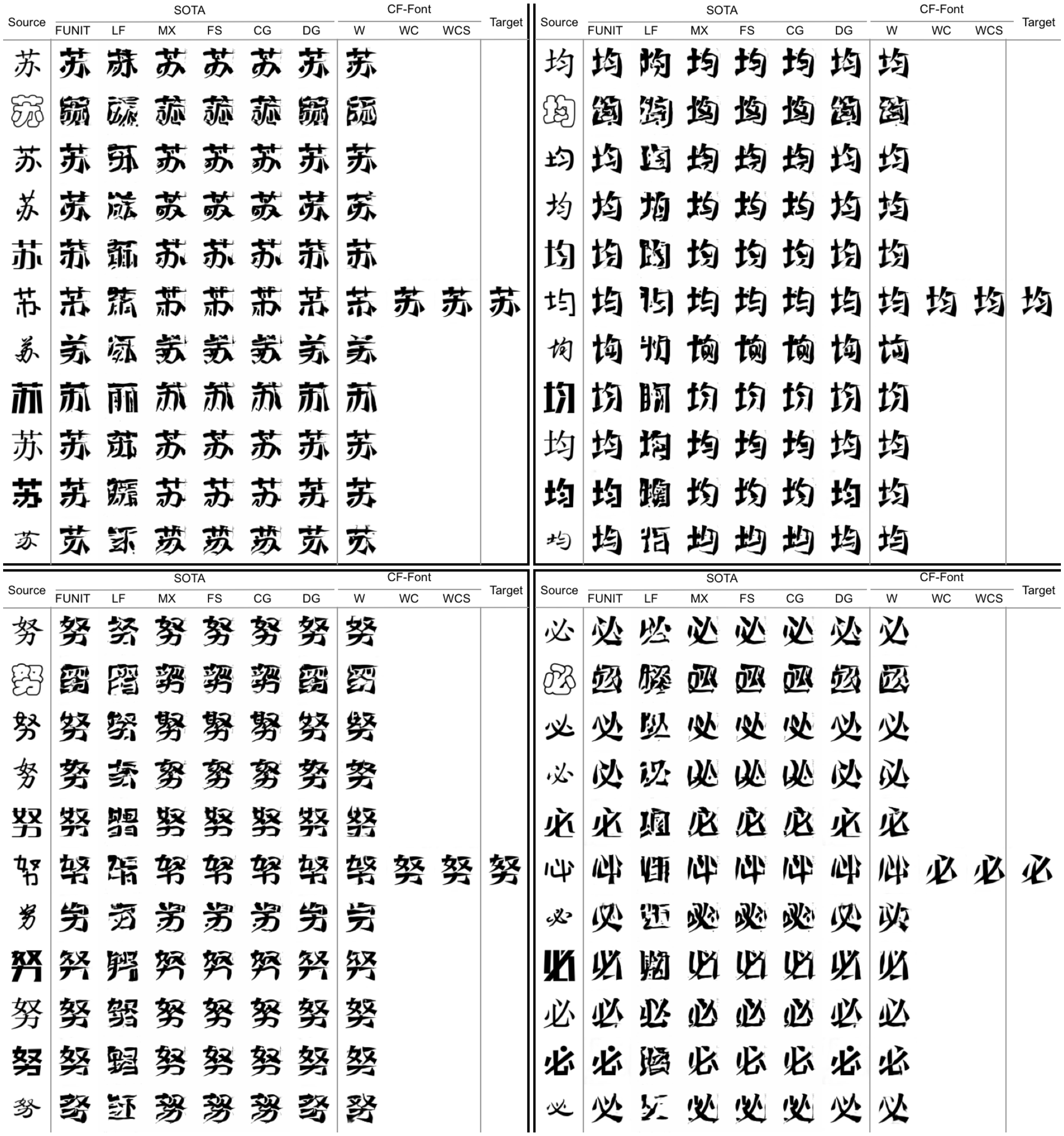

Figure 2. Detailed qualitative comparisons on a seen font with state-of-the-art methods. As mentioned in the subsection 4.3 of the paper, we use “Kai” and all basis fonts as source for these comparisons methods. LF, MX, FS, CG, and DG represent LF-Font, MX-Font, Fs-Font, CG-GAN, and DG-Font respectively.

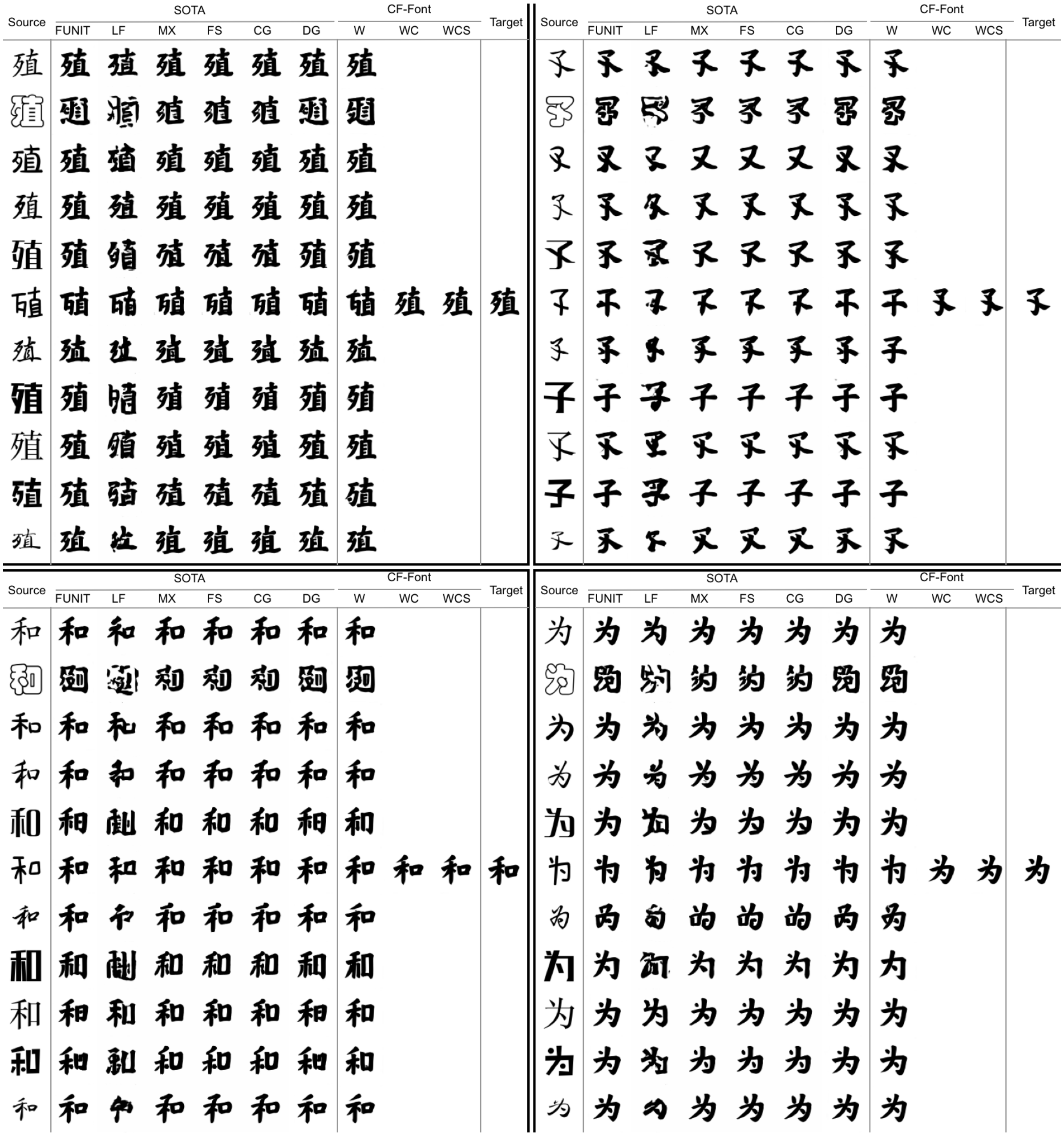

Figure 3. Detailed qualitative comparisons on a seen font with state-of-the-art methods. LF, MX, FS, CG, and DG represent LF-Font, MX-Font, Fs-Font, CG-GAN, and DG-Font respectively.

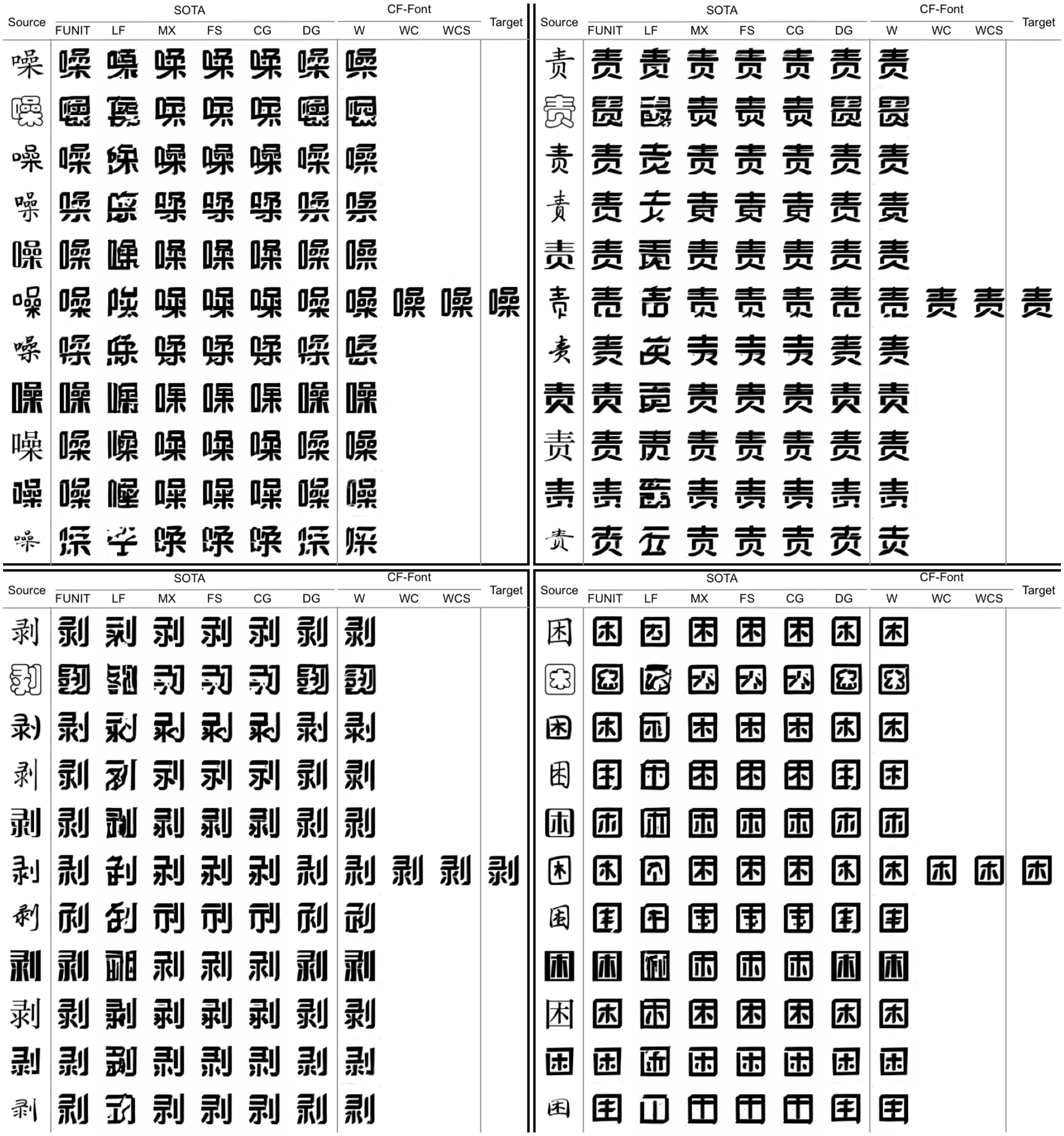

Figure 4. Detailed qualitative comparisons on a seen font with state-of-the-art methods. LF, MX, FS, CG, and DG represent LF-Font, MX-Font, Fs-Font, CG-GAN, and DG-Font respectively.

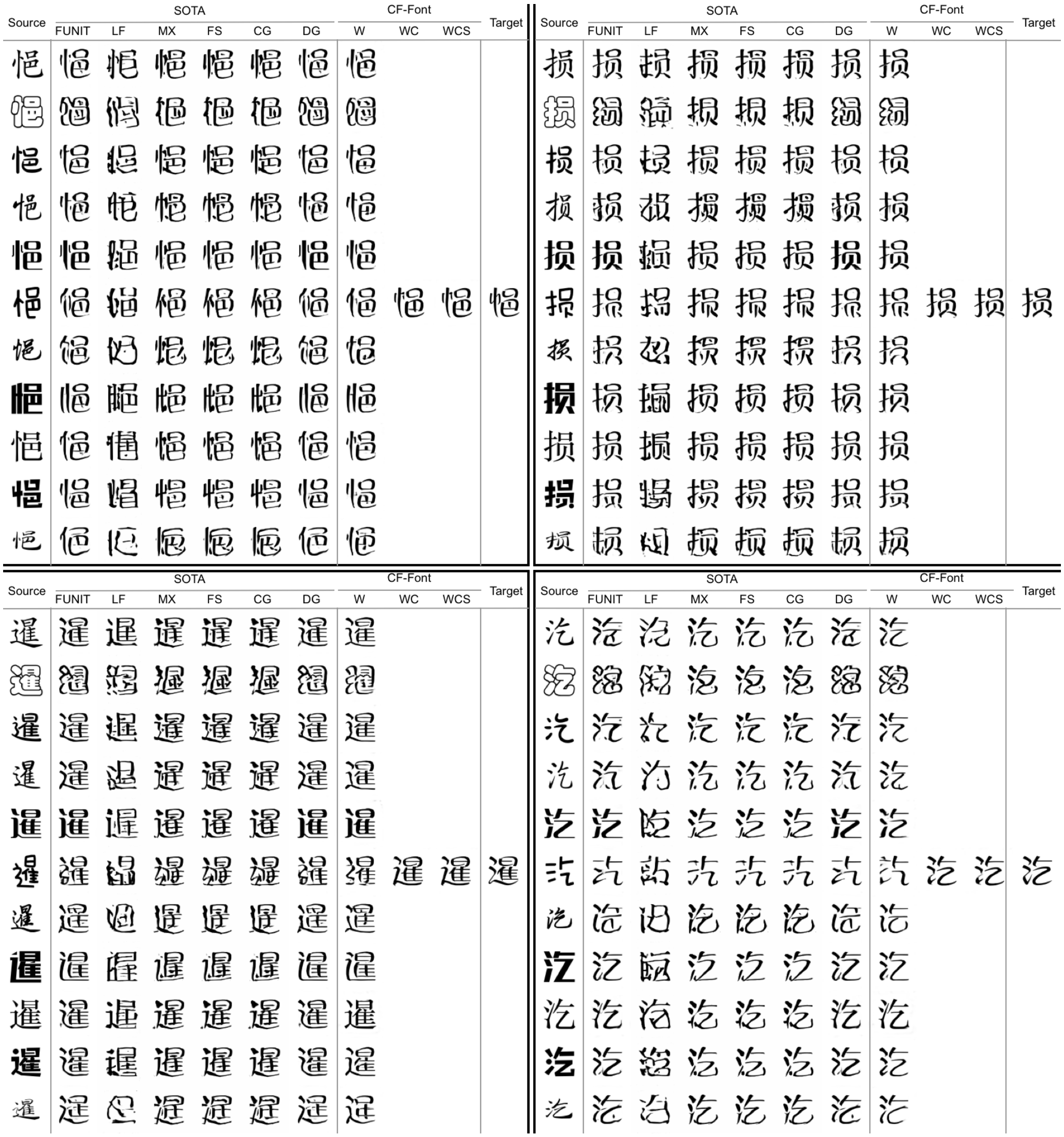

Figure 5. Detailed qualitative comparisons on a seen font with state-of-the-art methods. LF, MX, FS, CG, and DG represent LF-Font, MX-Font, Fs-Font, CG-GAN, and DG-Font respectively.

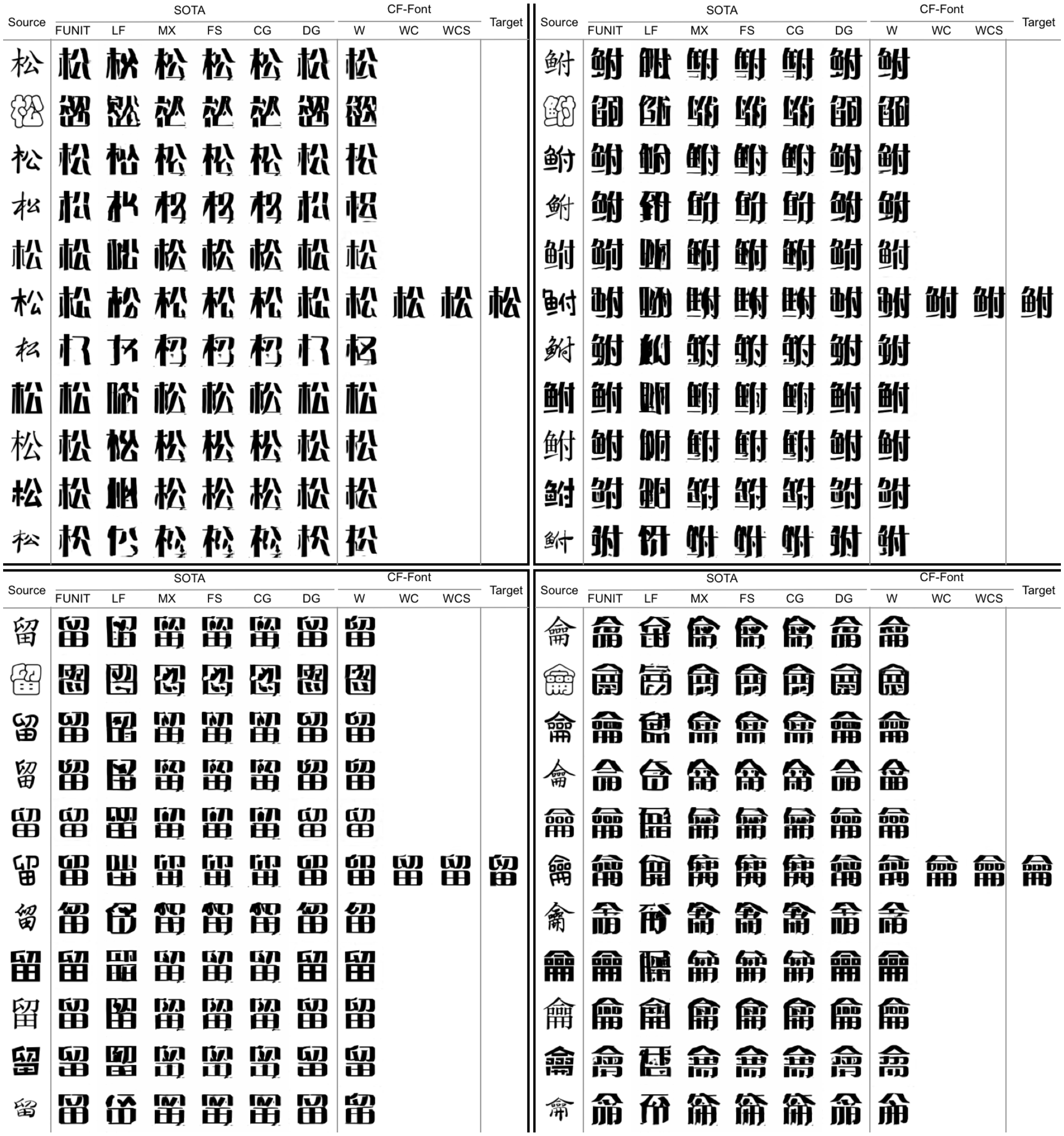

Figure 6. Detailed qualitative comparisons on a seen font with state-of-the-art methods. LF, MX, FS, CG, and DG represent LF-Font, MX-Font, Fs-Font, CG-GAN, and DG-Font respectively.

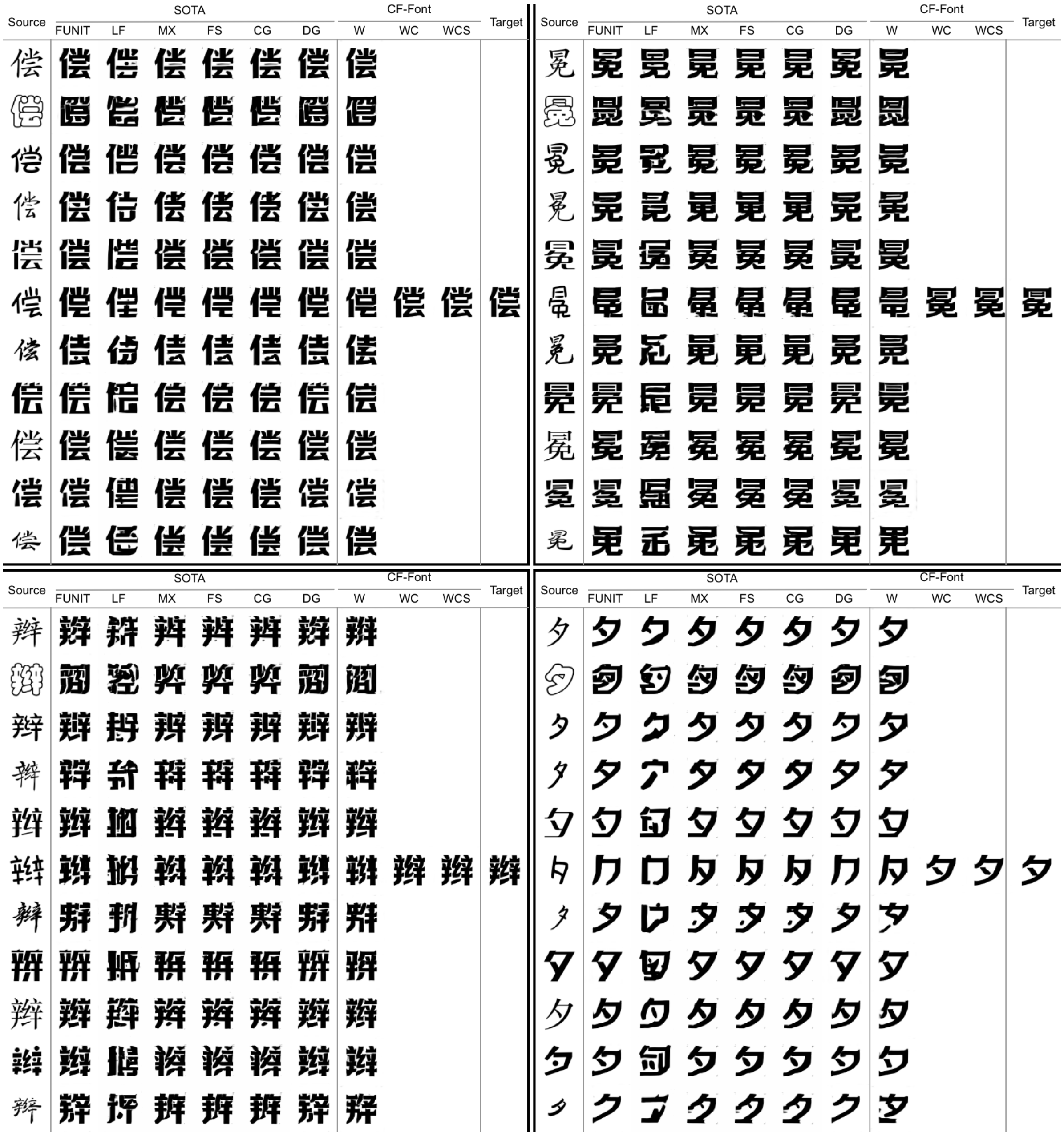

Figure 7. Detailed qualitative comparisons on a seen font with state-of-the-art methods. LF, MX, FS, CG, and DG represent LF-Font, MX-Font, Fs-Font, CG-GAN, and DG-Font respectively.

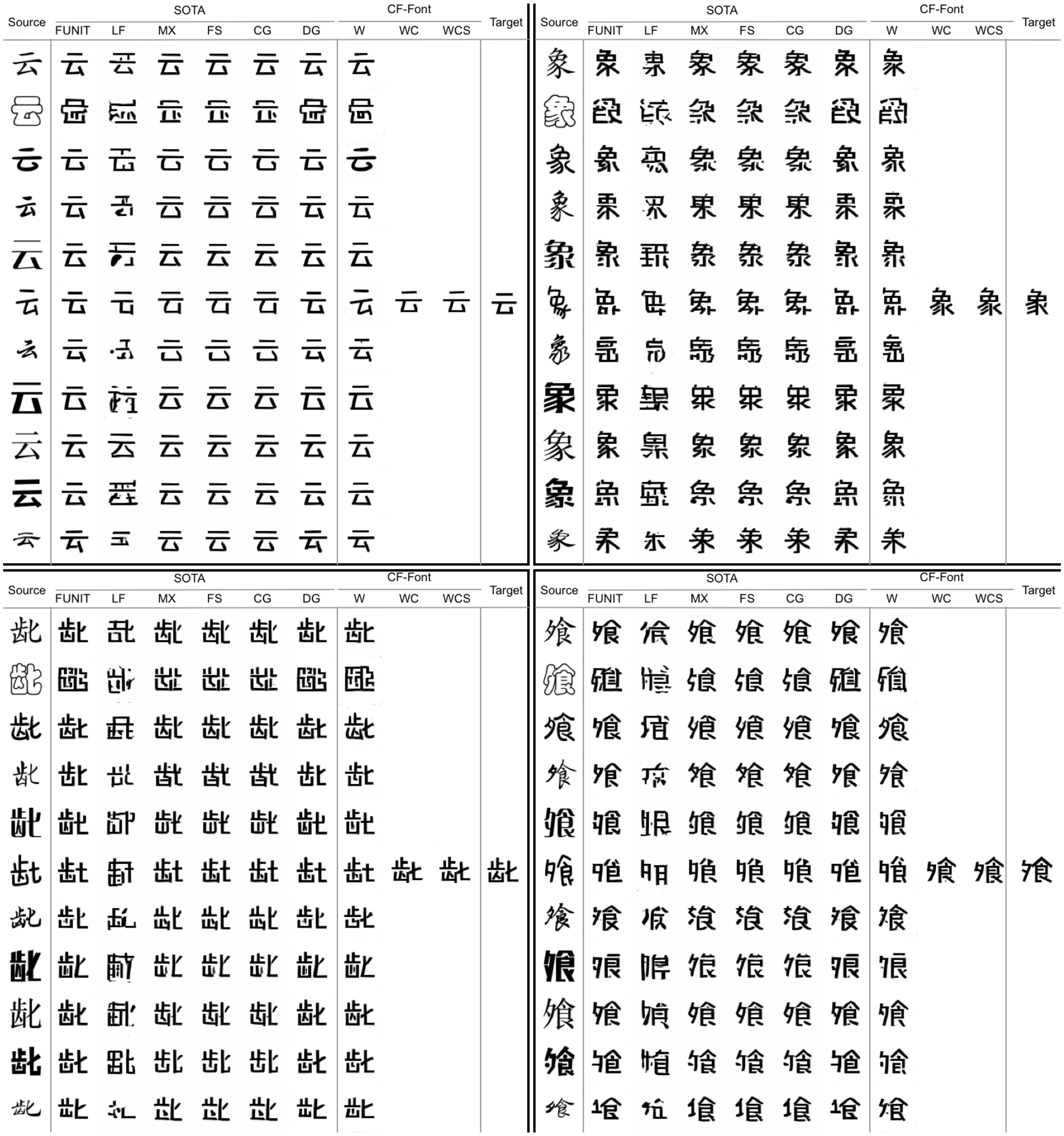

Figure 8. Detailed qualitative comparisons on a seen font with state-of-the-art methods. LF, MX and DG represent LF-Font, MX-Font and DG-Font respectively.

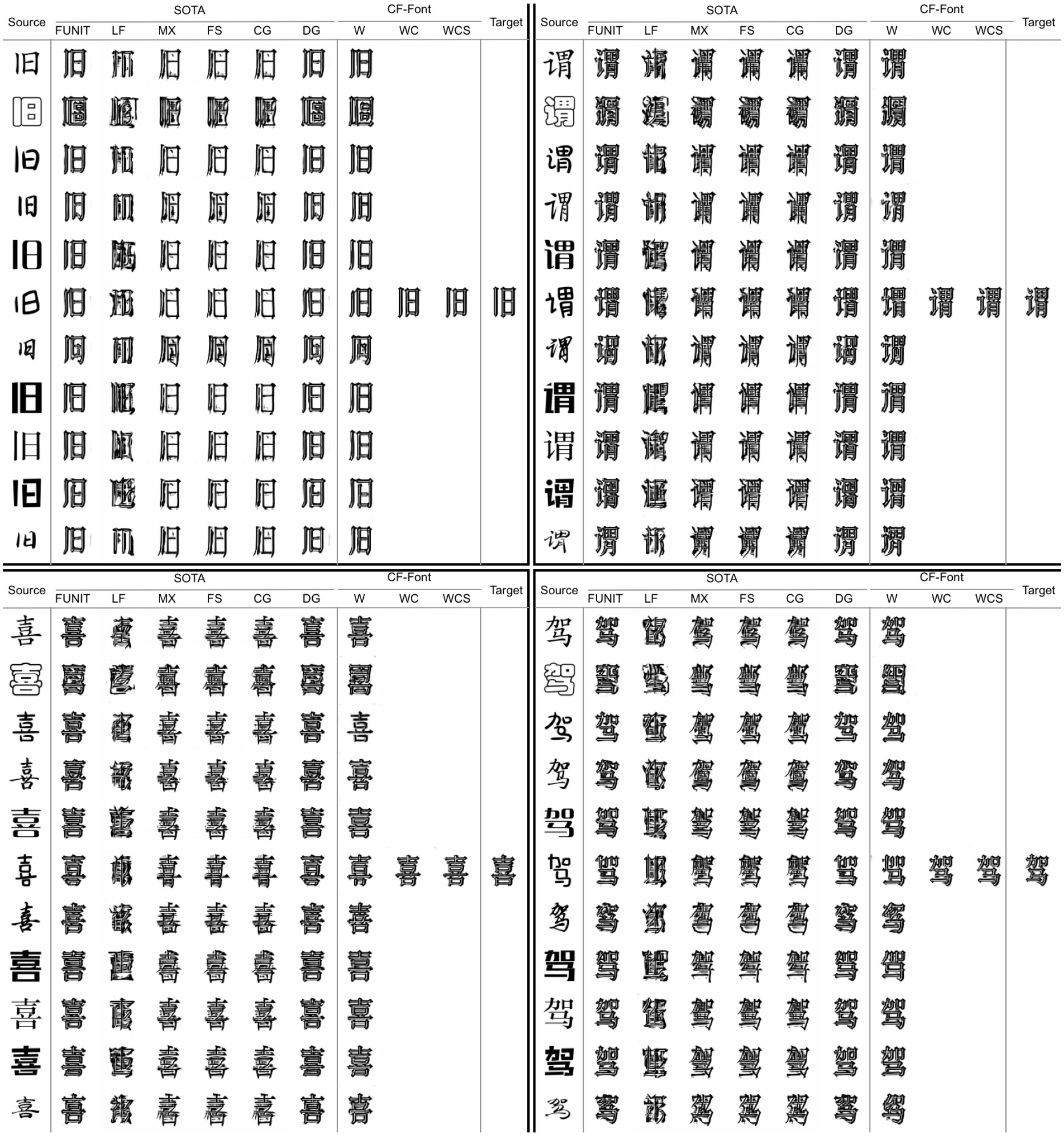

Figure 9. Detailed qualitative comparisons on a seen font with state-of-the-art methods. LF, MX, FS, CG, and DG represent LF-Font, MX-Font, Fs-Font, CG-GAN, and DG-Font respectively.

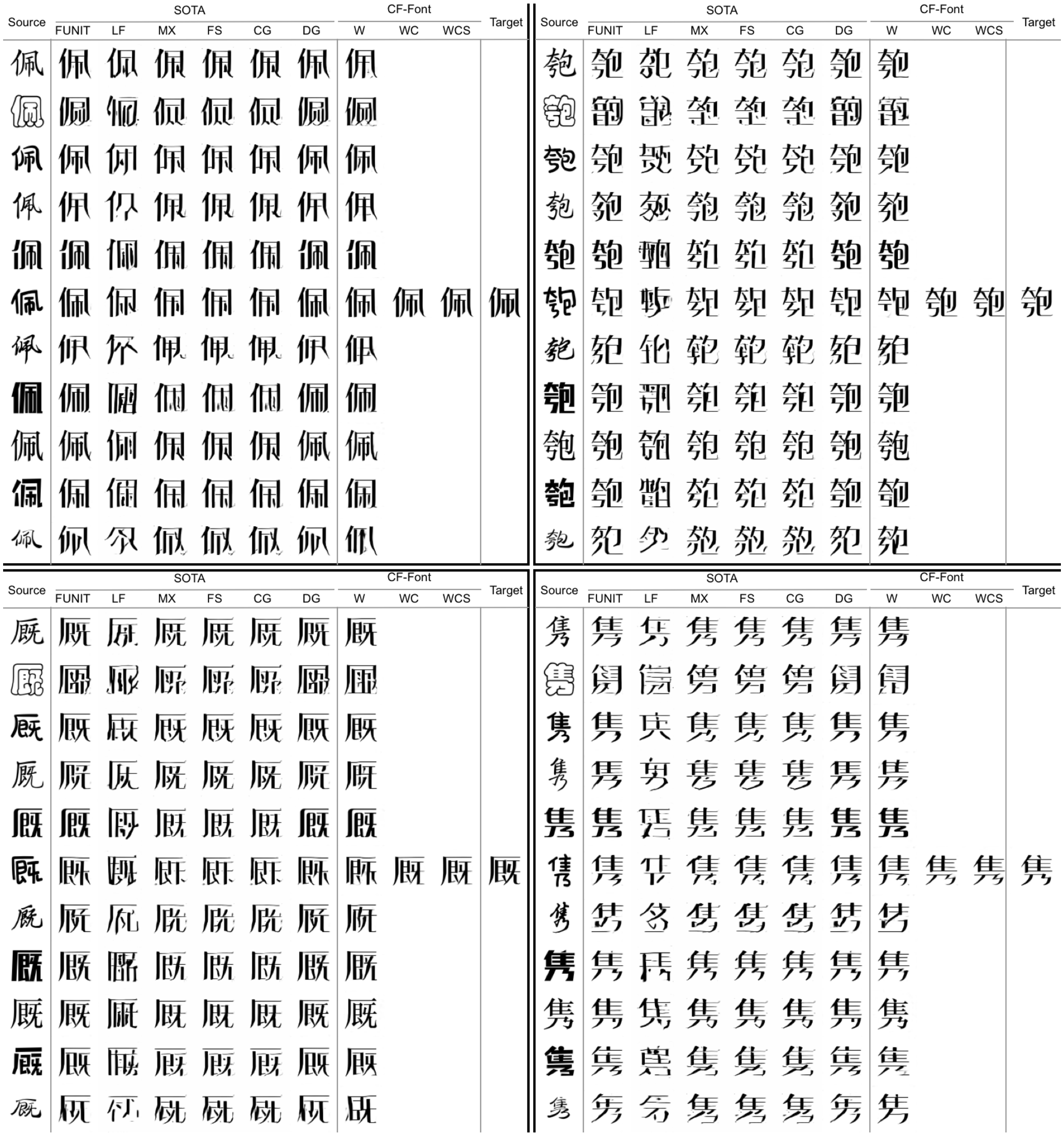

Figure 10. Detailed qualitative comparisons on an unseen font with state-of-the-art methods. LF, MX, FS, CG, and DG represent LF-Font, MX-Font, Fs-Font, CG-GAN, and DG-Font respectively.

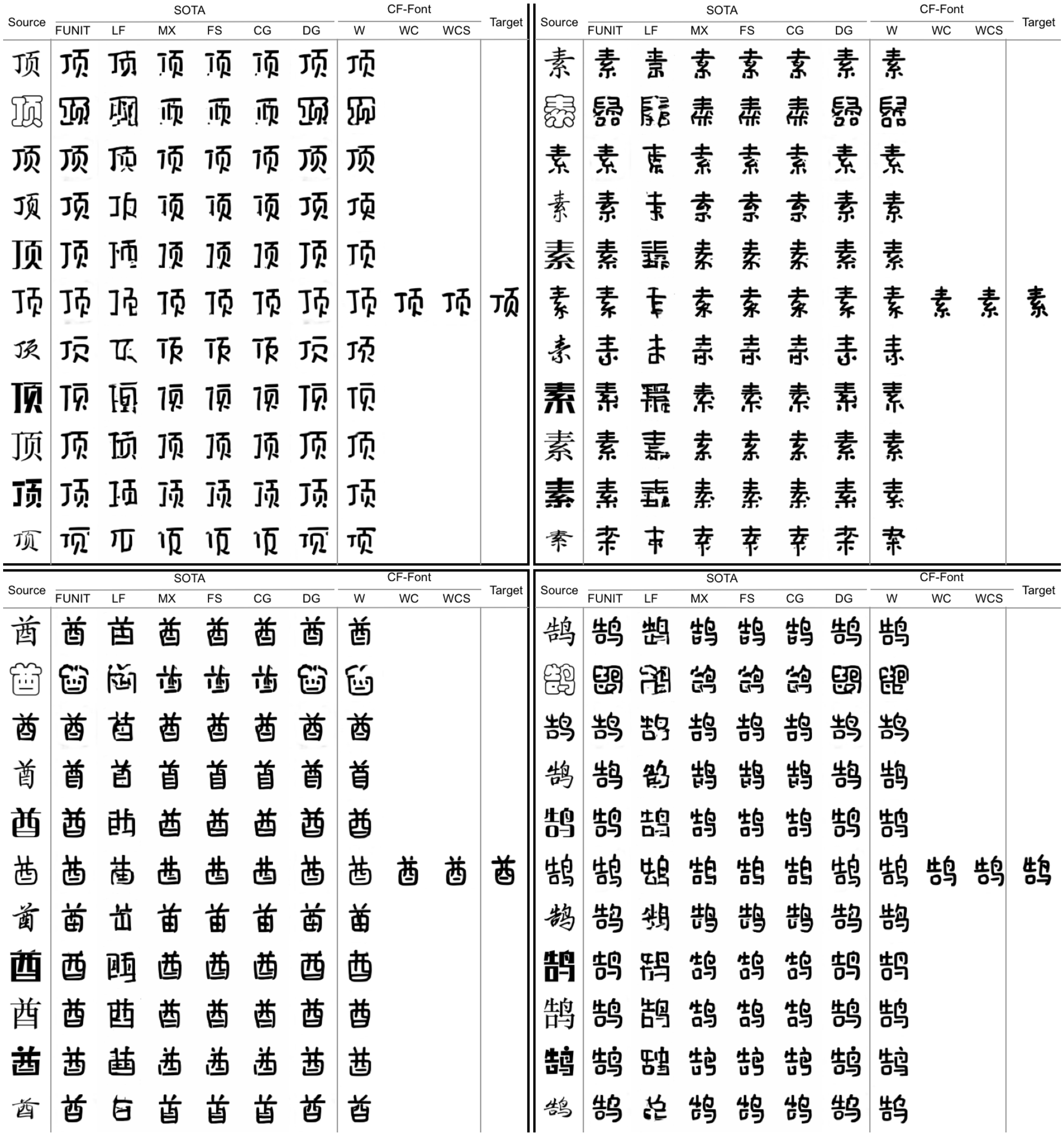

Figure 11. Detailed qualitative comparisons on an unseen font with state-of-the-art methods. LF, MX, FS, CG, and DG represent LF-Font, MX-Font, Fs-Font, CG-GAN, and DG-Font respectively.

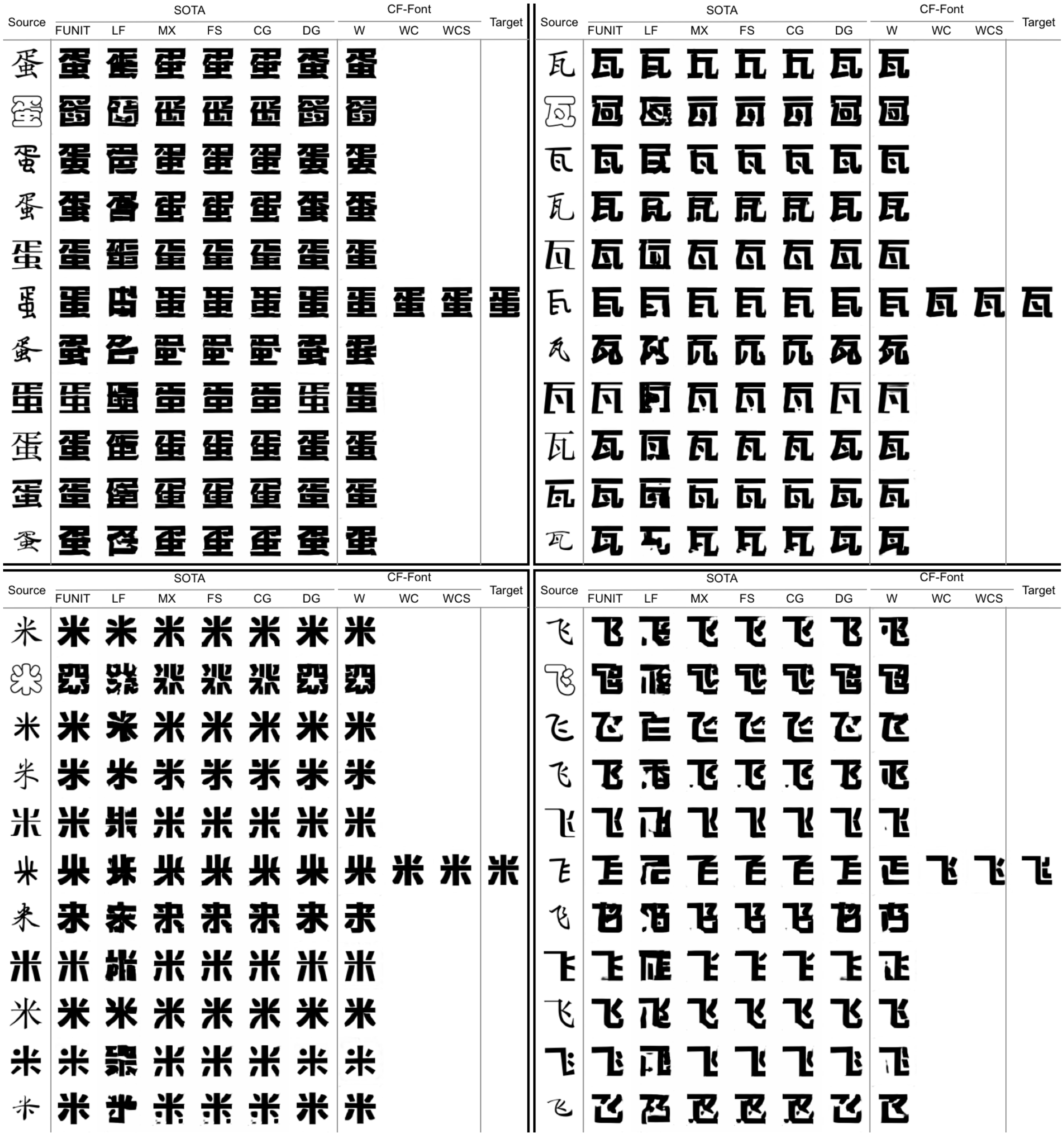

Figure 12. Detailed qualitative comparisons on an unseen font with state-of-the-art methods. LF, MX, FS, CG, and DG represent LF-Font, MX-Font, Fs-Font, CG-GAN, and DG-Font respectively.

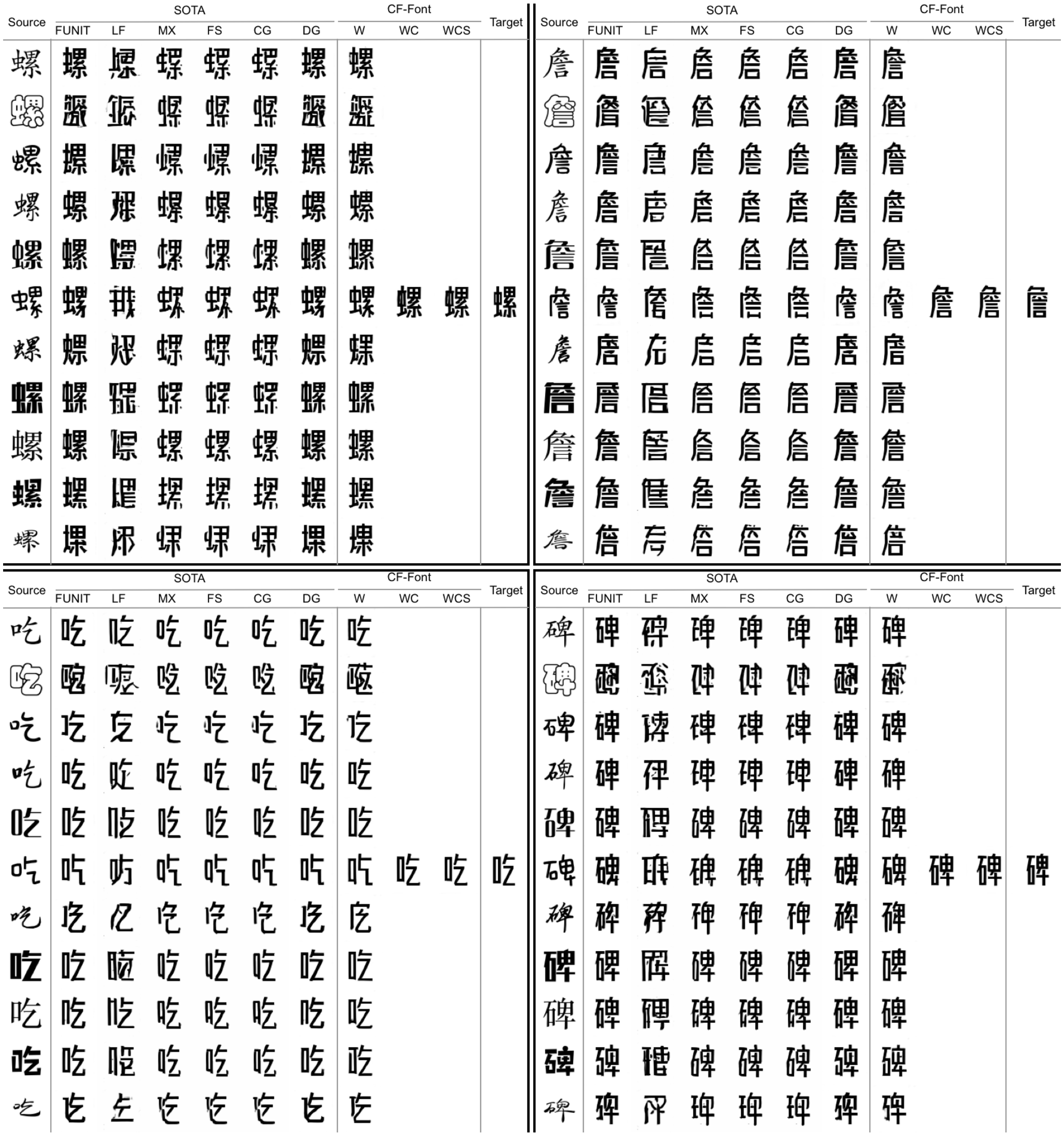

Figure 13. Detailed qualitative comparisons on an unseen font with state-of-the-art methods. LF, MX, FS, CG, and DG represent LF-Font, MX-Font, Fs-Font, CG-GAN, and DG-Font respectively.

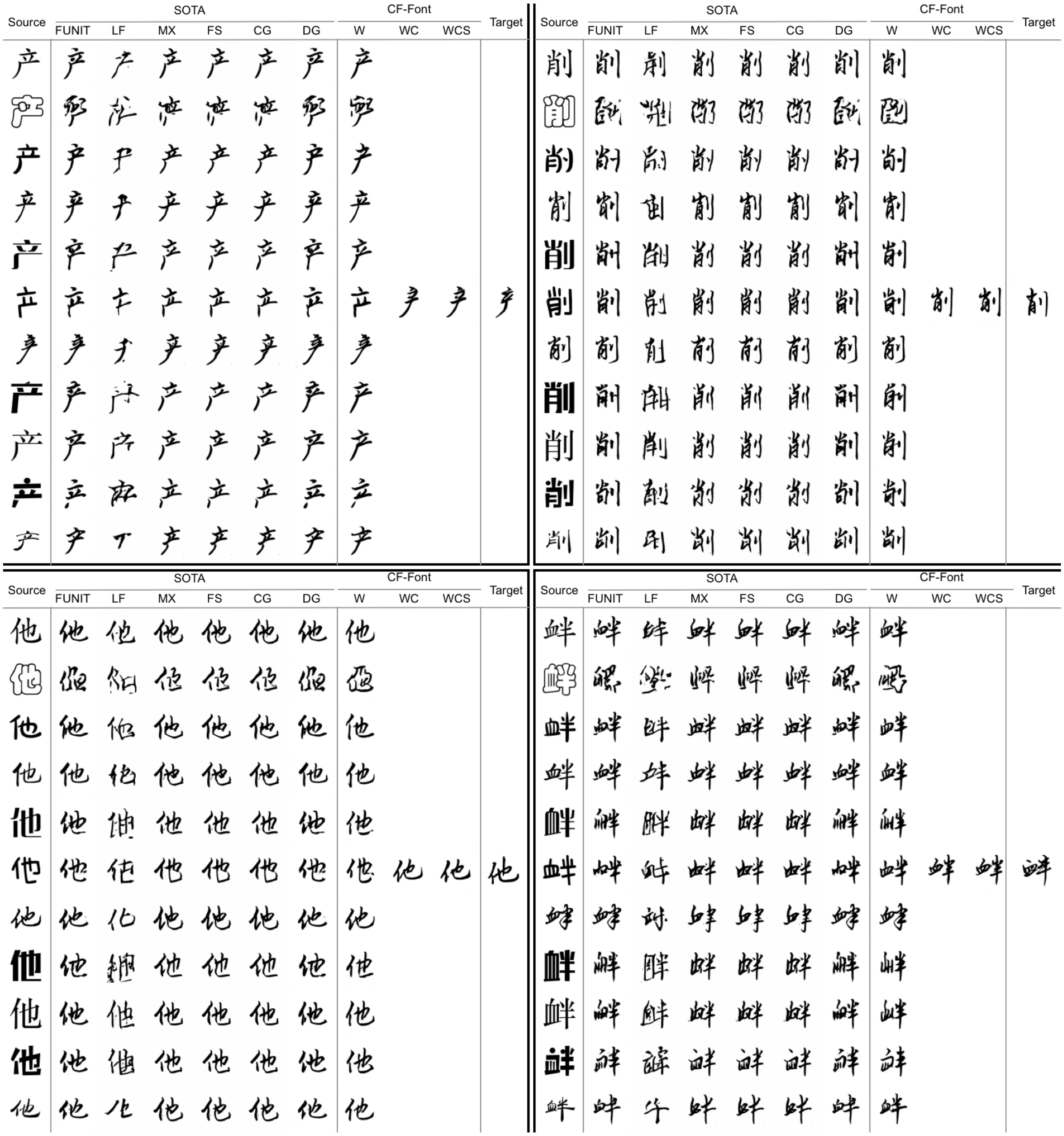

Figure 14. Detailed qualitative comparisons on an unseen font with state-of-the-art methods. LF, MX, FS, CG, and DG represent LF-Font, MX-Font, Fs-Font, CG-GAN, and DG-Font respectively.