Appendix

We describe additional experimental results to complement the main paper (§A). The implementation details are in §B. Finally, we provide the detailed evaluation protocols (§C).

A. Additional experimental results

A.1. More visual examples

We show more generated glyphs in Figure A.2. MX-Font correctly synthesizes the strokes, dot, thickness and size of the ground truth glyphs. In the cross-lingual FFG, MX-Font can produce promising results in that they are all readable. Meanwhile, all other competitors provide inconsistent results, which are often impossible to understand. These results show a similar conclusion as our main paper.

A.2. Impact of the number of experts

In Table A.1, we report the performances by varying the number of experts, $k$. We observe that larger $k$ brings better performances until $k = 6$, but larger $k$, e.g., 8, shows slightly worse performance than $k = 6$. We presume that this is because there are no sufficient data having more than or equal to eight components for training all the eight experts to capture different concepts. Figure A.1 illustrates the frequency of the number of components. From this graph, we find that the most characters have less than 8 components in our Chinese dataset. Moreover, larger $k$ means the number of parameters are increased, resulting in more training and inference runtime. Hence, in the paper, we choose $k = 6$ for all experiments.

B. Implementation details

B.1. Network architecture

Each localized expert $E_i$ has 11 layers including convolution, residual, global-context [3], and convolutional block attention (CBAM) [18] blocks. The multiple localized experts share the weights of their first five blocks. The two feature classifiers $Cls_x$ and $Cls_u$ have the same structure; a linear block following two residual blocks. The weights of the first two residual blocks are shared. The generator $G$ consists of convolution and residual blocks. Please refer our code for the detailed architecture.

B.2. Component allocation problem to weighted bipartite B-matching problem

Given a bipartite graph $G = (V, E)$, where $V$ is a set of vertices, $E$ is a set of edges and $W$ is the weight values for each edge $e \in E$, the weighted bipartite B-matching (WBM) problem [13] aims to find subgraph $H = (V, E')$ maximizing $\sum_{e \in E'} W(e)$ with every vertex $v \in V$ adjacent to at most the given budget, $B(v)$, edges. WBM problem can be solved by the Hungarian algorithm [14], a typical algorithm to solve combinatorial optimization in a polynomial time, in $O(|V||E|) = O(|V|^3)$. For curious readers, we refer recent papers solving variants of WBM problems [5, 1].

<table>
<thead>
<tr>
<th>$k$</th>
<th>Acc (S)↑</th>
<th>Acc (C)↑</th>
<th>Acc (B)↑</th>
<th>LPIPS ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>72.2</td>
<td>98.7</td>
<td>71.4</td>
<td>0.133</td>
</tr>
<tr>
<td>2</td>
<td>79.0</td>
<td>99.3</td>
<td>78.5</td>
<td>0.128</td>
</tr>
<tr>
<td>4</td>
<td>78.3</td>
<td>99.5</td>
<td>78.0</td>
<td>0.125</td>
</tr>
<tr>
<td>6</td>
<td>78.9</td>
<td>99.5</td>
<td>78.7</td>
<td>0.120</td>
</tr>
<tr>
<td>8</td>
<td>75.5</td>
<td>99.5</td>
<td>75.2</td>
<td>0.123</td>
</tr>
</tbody>
</table>

Table A.1. Impact of the number of experts $k$. The models with different number of heads are compared on in-domain Chinese transfer benchmark. We used $k = 6$ for all experiments.
Figure A.2. Generation samples. We provide more generated glyphs with four reference glyphs.
We recall the component allocation problem described in the main paper:

\[
\begin{align*}
\max_{w_{ij} \in \{0,1\}^k} \sum_{i=1}^{k} \sum_{j \in U_c} w_{ij} p_{ij}, \\
\text{s.t.} \sum_{i=1}^{k} w_{ij} \geq 1 \text{ for all } j, \\
\sum_{j \in U_c} w_{ij} \geq 1 \text{ for all } i,
\end{align*}
\] (B.1)

We replace the last condition, \(\sum_{i=1}^{k} \sum_{j \in U_c} w_{ij} = \max(k, m)\) to the upper bound condition where \(\lceil \cdot \rceil\) denotes the ceiling function. For example, if \(k = 3\) and \(m = 4\), the budget for each expert is 2, while the budget for each component is 1. We build a bipartite graph where the vertex set contains all experts and all valid components, and the edge weights are the prediction probability \(p_{ij}\). Now (B.1) can be re-formulated by the WBM problem.

**B.3. HSIC Formulation**

When training MX-Font, we let the two feature outputs from different experts, or content and style features independent of each other. To measure the independence between content feature and style feature, we first assume that the content features \(f_c\) and the style features \(f_s\) are drawn from two different random variables, \(Z_c\) and \(Z_s\), i.e., \(f_c \sim Z_c\) and \(f_s \sim Z_s\). We employ Hilbert Schmidt independence criterion (HSIC) [7] to measure the independence between two random variables. For two random variables \(Z_c\) and \(Z_s\), HSIC is defined as \(\text{HSIC}_{k,l}^1(Z_c, Z_s) : = \| C_{Z_c, Z_s} \|_{\text{HS}}^2 \) where \(k\) and \(l\) are kernels, \(C_{k,l}^{1}\) is the cross-covariance operator in the Reproducing Kernel Hilbert Spaces (RKHS) of \(k\) and \(l\), \(\| \cdot \|_{\text{HS}}\) is the Hilbert-Schmidt norm [7, 8]. If we use radial basis function (RBF) kernels for \(k\) and \(l\), HSIC is zero if and only if two random variables are independent.

Since we only have the finite number of samples drawn from the distributions, we need a finite sample estimator of HSIC. Following Bahng et al. [2], we employ an unbiased estimator of HSIC, \(\text{HSIC}_{k,l}^1(Z_c, Z_s)\) [17] with \(m\) samples. Formally, \(\text{HSIC}_{k,l}^1(Z_c, Z_s)\) is defined as:

\[
\begin{align*}
\text{HSIC}_{k,l}^1(Z_c, Z_s) = \frac{1}{m(m-3)} & \left( \text{tr}(\tilde{Z}_c \tilde{Z}_s^T) + \right. \\
& \left. \frac{1^T \tilde{Z}_c 1 1^T \tilde{Z}_s 1}{(m-1)(m-2)} - \frac{2}{m-2} 1^T \tilde{Z}_c 1 \right)
\end{align*}
\] (B.2)

where \((i, j)\)-th element of a kernel matrix \(\tilde{Z}_c\) is defined as, \(\tilde{Z}_c(i, j) = (1 - \delta_{ij}) k(f_c^i, f_s^j)\), and the \(i\)-th feature in the mini-batch \(f_c^i\), is assumed to be sampled from the \(Z_c\), i.e., \(\{f_c^i\} \sim Z_c\). We similarly define \(Z_s(i, j) = (1 - \delta_{ij}) l(f_s^i, f_s^j)\).

In practice, we compute \(\text{HSIC}_{k,l}^1(Z_c, Z_s)\) in a mini-batch, i.e., \(m\) is the batch size. We use the RBF kernel with kernel radius 0.5, i.e., \(k(f_c^i, f_s^j) = \exp(-\frac{1}{2} \| f_c^i - f_s^j \|_2^2)\).

**B.4. GAN objective details**

We employ two conditional discriminators \(D_s\) and \(D_t\) which predict a style label \(y_s\) and a content label \(y_c\), respectively. In practice, we employ a multitask discriminator \(D\), and different projection embeddings for content labels and style labels, following the previous methods [15, 4, 16]. The hinge loss [20] is employed to high fidelity generation:

\[
\mathcal{L}_{adv}^D = \mathbb{E}_{(x, y_c, y_s)} \left[ \left| \left| 1 - D(x, y_s) \right| \right|_1 + \left| \left| 1 - D(x, y_c) \right| \right|_1 \right] + \mathbb{E}_{(\tilde{x}, y_c, y_s)} \left[ \left| \left| 1 - D(\tilde{x}, y_s) \right| \right|_1 + \left| \left| 1 - D(\tilde{x}, y_c) \right| \right|_1 \right]
\]

\[
\mathcal{L}_{adv}^C = -\mathbb{E}_{(\tilde{x}, y_c, y_s)} \left[ D(\tilde{x}, y_s) + D(\tilde{x}, y_c) \right],
\] (B.3)

where \(\tilde{x}\) is the generated image by combining a content feature extracted from an image with content label \(y_c\) and a style feature extracted from an image with style label \(y_s\).

The feature matching loss \(\mathcal{L}_{fm}\) and the reconstruction loss \(\mathcal{L}_{recon}\) are formulated as follows:

\[
\mathcal{L}_{fm} = \mathbb{E}_{(x, \tilde{x})} \left[ \sum_{l=1}^{L-1} \left| \left| D^l(x) - D^l(\tilde{x}) \right| \right|_1 \right],
\]

\[
\mathcal{L}_{recon} = \mathbb{E}_{(x, \tilde{x})} \left[ \left| \left| x - \tilde{x} \right| \right|_1 \right],
\] (B.4)

where \(L\) is the number of layers in the discriminator \(D\) and \(D^l\) denotes the output of \(l\)-th layer of \(D\).

**B.5. Training details**

We use Adam [12] optimizer to optimize the MX-Font. The learning rate is set to 0.001 for the discriminator and 0.0002 for the remaining modules. The mini-batch is constructed with the target glyph, style glyphs, and content glyph during training. Specifically, we first pick the target glyph randomly, then we randomly select \(n\) style glyphs with the same style as the target glyph, and \(n\) content glyphs with the same character as the target glyph for each target glyph. Here, the target glyph is excluded from the style and content glyphs selection. We set \(n = 3\) during training. We set the number of heads \(k\) to 6 and train the model for 650k iteration with the full objective functions for the Chinese glyph generation. For the Korean, we set the number of heads \(k\) to 3 and train the model for 200k iteration with the all objective functions except \(\mathcal{L}_{\text{indp exp}}\). We do not employ the \(\mathcal{L}_{\text{indp exp}}\) during training for the Korean glyph generation, due to the special characteristic of the Korean script; always decomposed to fixed number of components, e.g., 3.
C. Evaluation details

C.1. Classifiers

Three classifiers are trained for the training; the style classifier, the Chinese character classifier, and the Korean character classifier. The style classifier and the Chinese character classifier are trained with the same Chinese dataset, including 209 Chinese fonts and 6428 Chinese characters per font. Besides, we used the Korean dataset that DM-Font [4] provides to train the Korean character classifier. The classifiers have ResNet-50 [9] structure. We optimize the classifiers using AdamP optimizer [10] with learning rate 0.0002 for 20 epochs. During training, the CutMix augmentation [19] is adopted and the mini-batch size is set to 64.

C.2. LF-Font modification

Since LF-Font [16] cannot handle the unseen components in the test time due to its component-conditioned structure, we modify its structure to enable the cross-lingual font generation. We loose the component-condition of LF-Font in the test time only, by skipping the component-condition block when the unseen component is given. Note that, we use original LF-Font structure for the training to reproduce its original performance.

C.3. User study examples

We show the sample images used for the user study in Figure C.1. Five methods, including EMD [21], AGIS-Net [6], FUNIT, LF-Font [16], and MX-Font are randomly displayed to users for every query.

C.4. FID

We measure the style-aware and content-aware Fréchet inception distance (FID) [11] between generated images and rendered images using the style and content classifier. For the Chinese glyphs, the style-aware and content-aware FIDs are measured with the generated glyphs and the corresponding ground truth glyphs. Since the ground truth glyphs of cross-lingual generation do not exist, the style-aware FID is measured the generated glyphs and all the available rendered glyphs having the same style with the generated images. The content-wise FID is measured similar to the style-aware FID. The style-aware (S) and the content-aware (C) FID values and their harmonic mean (H) are reported in Table C.1. Despite that MX-Font shows the slight degradation in FID for Chinese font generation, these results are not consistent with the user study and qualitative evaluation. For quantifying the image quality, we tend to trust the user study more because it better reveals the user’s preference.

References


Table C.1. We provide style-aware (S), content-aware(C) FIDs measured by the style and content classifiers. The harmonic mean (H) of the style-aware and the content-aware FIDs values are identical to the values reported in the main table.

<table>
<thead>
<tr>
<th>Method</th>
<th>FIDs</th>
<th>CN → CN</th>
<th>CN → KR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S</td>
<td>C</td>
<td>H</td>
</tr>
<tr>
<td>EMD</td>
<td>145.5</td>
<td>51.1</td>
<td>79.7</td>
</tr>
<tr>
<td>AGIS-Net</td>
<td>91.0</td>
<td>10.8</td>
<td>19.2</td>
</tr>
<tr>
<td>FUNIT</td>
<td>50.6</td>
<td>11.8</td>
<td>19.2</td>
</tr>
<tr>
<td>LF-Font</td>
<td>43.5</td>
<td>9.0</td>
<td>14.8</td>
</tr>
<tr>
<td>MX-Font</td>
<td>50.5</td>
<td>13.9</td>
<td>21.8</td>
</tr>
</tbody>
</table>

Figure C.1. User study examples. The example images that we provide to the candidates are shown. Each image includes the reference images, source images, and the generated images.


