Elegantly Written: Disentangling Writer and Character Styles for Enhancing Online Chinese Handwriting

Yu Liu[1,2], Fatimah binti Khalid[1], Lei Wang[1], Youxi Zhang[1], and Cunrui Wang[2]

[1] Faculty of Computer Science and Information Technology, University Putra Malaysia, 43400 UPM Serdang, Malaysia
{gs64481,fatimahk,gs68180,gs64680}@upm.edu.my
[2] Dalian Chinese Font Design Technology Innovation Center, Dalian Minzu University, 116600 Dalian, China
wcr@dlnu.edu.cn

A. 1 Reference Selection

In many font generation methods, at each iteration, one or more characters are randomly selected from the training set as style reference. This makes it challenging for the model to accurately extract styles from diverse combinations of style reference sets. Therefore, we introduce a strategy by setting a fixed style reference set $S = \{s_1, s_2, s_3, \ldots, s_n\}$, which is composed of 500 characters. Additionally, we design a content-style reference mapping to select a fixed K style reference characters for each content character.

A. 1.1 Decomposition of Chinese Character Structure

According to the structure decomposition table of Chinese characters, each character is decomposed into a component set T consisting of explicit components T^c and implicit components T^h. An example of character decomposition is shown in Figure 1. During the decomposition process, it is ensured that the character retains the integrity of its components, which are then further subdivided. The components obtained from the initial decomposition are defined as explicit decomposition $T^c = \{t_1^c, t_2^c, t_3^c, \ldots, t_n^c\}$, and those obtained from further decomposition are defined as implicit decomposition $T^h = \{t_1^h, t_2^h, t_3^h, \ldots, t_n^h\}$. The explicit decomposition contains structures that are more easily transferred from the style reference set to the target set, while the implicit decomposition ensures that each character can select K reference characters from the style reference set.

A. 1.2 Content-Style Reference Mapping:

We select 2000 high-frequency commonly used Chinese characters as the content character set $C = \{c_1, c_2, c_3, \ldots, c_n\}$. Our goal is to map K style references

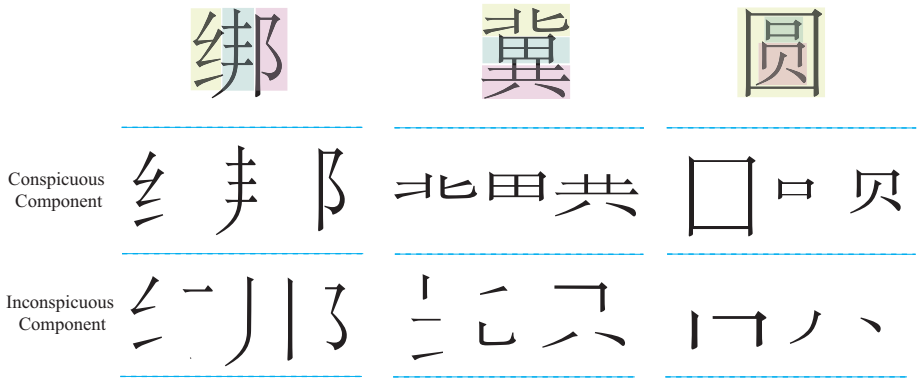

Fig. 1: An example of decomposition into explicit and implicit components according to the structure of Chinese characters.

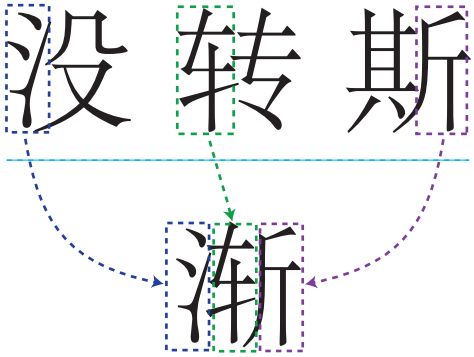

Fig. 2: Example of selecting style reference characters based on the component maximization principle.

for each content glyph based on the component maximization principle (see Appendix A.1.4 for more details). In this process, we first obtain the explicit decomposition $T^c_{c_i}$ of the content glyph set c_i, then obtain the explicit decomposition $T^c_{s_z}$ of each character s_z in the style reference set S. In each search step, we find s for c_i with the same explicit decomposition and include it in the style mapping set. If the style mapping set contains fewer than K glyphs, we search in the implicit decomposition. After selecting K style reference glyphs for each content glyph c_i, we sort the style reference set S by frequency to ensure coverage of each character. Through this process, each content glyph determines the corresponding K style reference glyphs. An example of style reference character selection is shown in Fig. 2 . The specific process is as follows algorithm 1.

Algorithm 1: Content-Style Reference Set Mappings

Input: $C = \{c_i\}$: High-frequency characters list.
$S = \{s_j\}$: Style reference character list.
$T = \{T^c; T^h\}$: Chinese character structure component set.
K: Number of style reference characters.
Output: $\hat{U}$: Content-Style Reference Set.

Function `mapping`$(c_i, \hat{T})$**:**
  $R_{c_i} \leftarrow \emptyset$; $k \leftarrow 0$;
  $T_{c_i} \leftarrow \hat{T}(c_i)$; $S \leftarrow Sort(S) \uparrow$;
  for $t_l \in T_{c_i}$ **do**
    for $s_z \in S$ **do**
      $T_{s_z} \leftarrow \hat{T}(s_z)$;
      for $t_m \in T_{s_z}$ **do**
        if $t_l = t_m$ ***and*** $s_i \notin R$ **then**
          $R_{c_i} \leftarrow s_i$; $k \leftarrow k + 1$;
          $S(s_i) \leftarrow S(s_i) + 1$;
        end
        if $k \geq K$ **then**
          return R_{c_i}
        end
      end
    end
  end
return R_{c_i}
Function `Main()`**:**
  $R \leftarrow \emptyset$;
  for $c_i \in C$ **do**
    $R_{c_i} \leftarrow$ `mapping`(c_i, T^c)
    if $Len(R_{c_i}) < K$ **then**
      $R_{c_i} \leftarrow$ `mapping`(c_i, T^h)
      if $Len(R_{c_i}) < K$ **then**
        $R_{c_i} \leftarrow random(S, K - k)$
      end
    end
    $R \leftarrow R_{c_i}$
  end
return R

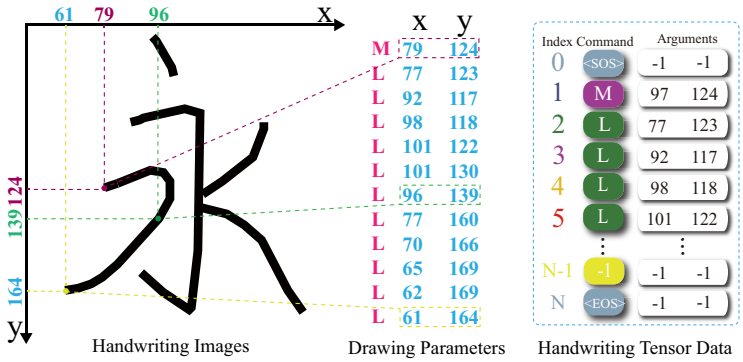

Fig. 3: Graphical parameter data structure visual description. Take one of the strokes as an example, the beginning of the drawing instruction is marked by $<$ SOS $>$, a command in the upper left cornerM moves the starting point of the drawing, drawing $<$SOS$>$a series of trajectories, and then the $<$ EOS $>$ command indicates the end of the drawing instruction.

A. 2 Data Structure

We treat the user's handwriting trajectory as a vector image drawn by Bezier curves. Bezier curve drawing instructions are used to simulate the dynamic process of handwriting. Specifically, the instruction M (MoveTo) indicates when the user's pen leaves the writing surface and moves to a new position in the air, while the instruction L (LineTo) represents the action of the user's pen touching down and moving on the paper. In this way, the user's handwriting process is transformed into a series of combinations of M and L instructions, accurately depicting the trajectory of handwriting. This representation preserves not only the geometric features of the handwriting trace but also captures the dynamic changes during the handwriting process.

To enable parallel processing, we convert the non-structural drawing parameters into structural tensor data. Each handwriting trace c consists of N drawing parameters V. Each drawing parameter $v_i = (h_i, p_i)$ consists of its drawing instruction type and drawing coordinates. The drawing instruction type $h_i \in \{<\text{SOS}>, \text{M}, \text{L}, <\text{EOS}>\}$, where $<$ SOS $>$ and $<$ EOS $>$ represent special markers for the beginning and end of the sequence. The drawing coordinate parameter $p_i = (x, y)$. If the length of the drawing instruction sequence V is less than N, padding with -1 is used.

Raster images are represented in fixed sizes, while vector images are composed of unstructured data of drawing parameters. The length of drawing parameters is influenced by the complexity of the character shape, resulting in the inability to align parameters of different font shapes. To enable parallel processing of the model, this paper converts the unstructured drawing parameters into structured tensor data. An example of a handwriting trace vector image and its tensor representation is shown in Fig. 3.

A. 3 Handwriting Feature Embedding

Handwriting traces are discrete data, and we preprocess them using a feature embedding method similar to natural language processing. For each drawing parameter sequence V of a handwriting trace, we project it into a continuous embedding space $\mathbb{R}^{N\times d}$.

First, the drawing command type h_i is represented using a 4-dimensional one-hot vector δ_{h_i}, where only one dimension represents the current instruction type index. Then, it is mapped to a d-dimensional vector $e^i_{\text{cmd}} = W_{\text{cmd}}\delta_{h_i}$ using a learnable embedding matrix $W_{\text{cmd}} \in \mathbb{R}^{d\times 4}$.

$$e^i_{\text{cmd}} = W_{\text{cmd}}\delta_{h_i} \tag{1}$$

Additionally, the drawing coordinate parameters p are embedded. First, each coordinate p_i is represented using a one-hot vector, where only one dimension represents the current coordinate index. Then, each coordinate is embedded by a learnable weight matrix $W_x \in \mathbb{R}^{d\times 128}$, followed by linear layer projection to a d-dimensional vector e^i_{coord} using,

$$e^i_{\text{coord}} = W_{\text{coord}}\ \text{vec}\left(W_x p_i\right). \tag{2}$$

where $\text{Vec}(\cdot)$ denotes flattening the matrix into a vector, and $W_{\text{coord}} \in \mathbb{R}^{d\times 2d}$. Each drawing command S_i is embedded into $e^i \in \mathbb{R}^d$. Absolute positional encoding $e^i_{\text{pos}} \in \mathbb{R}^d$ encodes the position and order information of all commands in a drawing sequence. The command type embedding, coordinate parameter embedding, and positional embedding of each S_i are summed, and their sum is projected into a vector $e^i \in \mathbb{R}^d$,

$$e^i = e^i_{\text{cmd}} + e^i_{\text{coord}} + e^i_{\text{pos}}. \tag{3}$$

A. 4 Character Dataset

We decompose Chinese characters into sets of components consisting of dominant components based on the Chinese character structure decomposition table. Based on the coverage between components, we filter out components that can cover commonly used characters. And 500 characters were screened as style reference characters based on the combination of these components. In addition, another 2000 Chinese characters were selected as content characters based on the frequency of use of Chinese characters in the "List of Commonly Used Characters in Modern Chinese".

A. 5 Visualization of Style Aggregation Module

To further demonstrate the effectiveness and high flexibility of the style aggregation module, we reveal how the module deeply aggregates features of style and

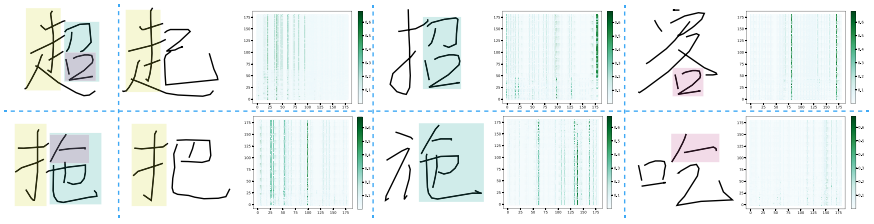

Fig. 4: Feature fusion effect of the style aggregation module under multiple style reference characters.

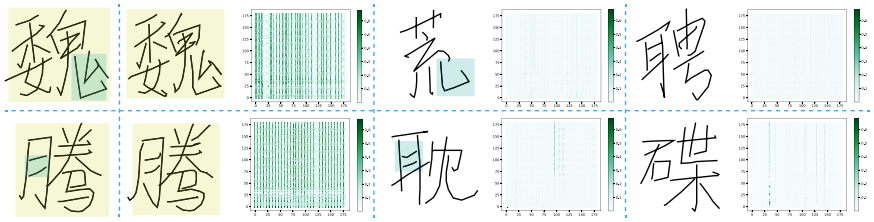

Fig. 5: Special case style reference character in the Style Aggregation module.

content through attention feature maps. In Fig. 4, three style reference characters are selected for each character, among which the third style reference character maintains very little visual similarity with the content character. Specifically, in Fig. 5, we select the same content character and style character, and additionally choose two other style reference characters with very little similarity. Such selection strategies aim to demonstrate the effectiveness and flexibility of the style aggregation module. The results show that the style aggregation module can flexibly select style reference characters based on the structure of content characters. Even when there is very little similarity between style and content characters, the module can still accurately capture and transfer those subtle yet crucial style features.

A. 6 More Online Generation Results

Figure 6 shows the results of our proposed method in beautifying handwriting traces. For the same input handwriting trace, we apply transformations to verify the ability of our method in online handwriting trace beautification. Two best matching style reference characters are selected for each character of the user's handwriting based on the principle of maximizing components. Our method improves the aesthetic appeal of characters while preserving the user's unique writing style. Importantly, the model does not directly learn style from style reference characters but learns the average style from all characters, resulting in significant improvement in the overall aesthetic appeal of the generated handwriting fonts.

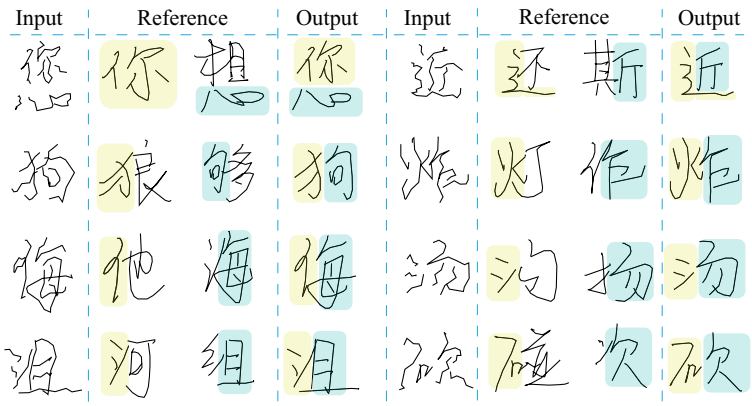

Fig. 6: Generation results of handwriting characters corrected by content-reference mapping.

Table 1: Comparisons with state-of-the-art methods on Chinese dataset. Unseen writing style seen characters.

Method	Style Score ↑	Content Score ↑	DTW ↓	User Prefer.(%) ↑
Drawing [5]	33.38	64.56	1.6021	8.42
FontRNN [4]	39.04	70.21	1.4531	10.23
DeepImitator [6]	52.88	87.18	1.3011	16.62
WriteLikeYou [3]	76.21	94.255	1.1101	47.53
SDT [1]	85.31	95.71	0.9554	55.24
Ours	89.10	96.04	0.8135	59.12

A. 7 Quantitative Comparison

We use two settings, Unseen Writing Style Seen Characters (USSC) and Unseen Writing Style Unseen Characters (USUC), to test these methods. USSC refers to the 2000 characters in the training set, while USUC refers to characters not in the training set. Table 1 shows the performance of our method and compared methods in USSC setting. Due to DeepImitator [6], WriteLikeYou [3], and SDT [1] learning style features across multiple images, all evaluation metrics are superior to FontRNN [4] and Drawing [5].

As shown in Table 2, except for our method and SDT [1] , other methods experience a significant decrease in performance in the USUC setting, indicating the generalization capability of our method to new characters. Although these methods perform well in evaluation metrics, human perception is sensitive to minor differences, and humans can easily detect the effects of these differences. Our method significantly outperforms previous SOTA methods, indicating outstanding style imitation performance of the proposed method.

Table 2: Comparisons with state-of-the-art methods on Chinese dataset. Unseen writing style unseen Characters.

Method	Style Score ↑	Content Score ↑	DTW ↓	User Prefer.(%) ↑
Drawing [5]	18.38	42.52	2.6241	1.52
FontRNN [4]	27.42	54.01	2.2185	3.19
DeepImitator [6]	40.24	71.79	1.8123	5.48
WriteLikeYou [3]	64.21	90.24	1.3923	37.54
SDT [1]	80.31	93.51	1.1954	50.23
Ours	85.10	94.12	0.8978	56.32

Table 3: Quantitatively evaluated on Japanese and Korean datasets.

Datasets	Methods	Style Score ↑	DTW ↓
Japanese	Drawing [5]	74.45	1.3598
	FontRNN [4]	80.23	1.2173
	DeepImitator [6]	87.24	0.9532
	WriteLikeYou [3]	90.18	0.8142
	SDT [1]	94.12	0.7412
	Ours	95.12	0.7034
Korean	Drawing [5]	68.42	1.4032
	FontRNN [4]	76.12	1.2941
	DeepImitator [6]	84.32	0.9931
	WriteLikeYou [3]	88.21	0.8823
	SDT [1]	93.81	0.7571
	Ours	94.50	0.7271

A. 8 Applications to Other Languages

For Japanese handwriting font generation tasks, experiments were conducted on the TUAT HANDS [2] database. A Korean skeleton was extracted as a Korean handwriting sample using a skeleton extraction algorithm. Random style reference characters were chosen for each content character in the experiment. Table 3 validate the effectiveness of our method in Japanese and Korean handwriting font generation, demonstrating outstanding performance in multilingual scenarios. Due to the fewer and simpler strokes of characters in Korean and Japanese Hiragana handwriting fonts compared to Chinese, they are easier to imitate.

References

1. Dai, G., Zhang, Y., Wang, Q., Du, Q., Yu, Z., Liu, Z., Huang, S.: Disentangling writer and character styles for handwriting generation. In: CVPR. pp. 5977–5986. IEEE (2023). https://doi.org/10.1109/cvpr52729.2023.00579
2. Matsumoto, K., Fukushima, T., Nakagawa, M.: Collection and analysis of on-line handwritten japanese character patterns. In: Proceedings of Sixth International

Conference on Document Analysis and Recognition. pp. 496–500. IEEE (2001). https://doi.org/10.1109/ICDAR.2001.953839
3. Tang, S., Lian, Z.: Write like you: Synthesizing your cursive online chinese handwriting via metric-based meta learning. Computer Graphics Forum **40**(2), 141–151 (2021). https://doi.org/10.1111/cgf.142621
4. Tang, S., Xia, Z., Lian, Z., Tang, Y., Xiao, J.: Fontrnn: Generating large-scale chinese fonts via recurrent neural network. Computer Graphics Forum **38**(7), 567–577 (2019). https://doi.org/10.1111/cgf.13861
5. Zhang, X.Y., Yin, F., Zhang, Y.M., Liu, C.L., Bengio, Y.: Drawing and recognizing chinese characters with recurrent neural network. PAMI **40**(4), 849–862 (2017). https://doi.org/10.1109/tpami.2017.2695539
6. Zhao, B., Tao, J., Yang, M., Tian, Z., Fan, C., Bai, Y.: Deep imitator: Handwriting calligraphy imitation via deep attention networks. Pattern Recognition **104**, 107080 (2020). https://doi.org/10.1016/j.patcog.2019.107080