

MSA²: Multi-task Framework with Structure-aware and Style-adaptive Character Representation for Open-set Chinese Text Recognition

Supplementary Material

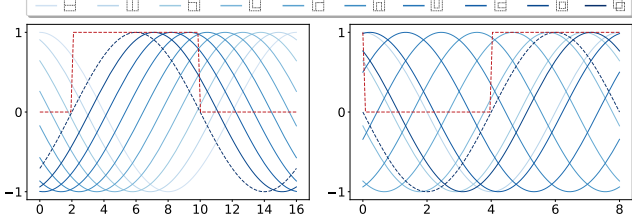


Figure 13. Illustration of the sinusoidal encoding for structures with varying representation spaces, where the red dashed line denotes a type of structure code generated through binarization.

A. Implement details

Structure Encoding Algorithm In existing methods, all structures within a character are uniformly represented using 4-bit binary encoding. In contrast, SACE employs different encoding strategies for structures at varying depth layers to emphasize the distinction between primary and secondary structures, as detailed in Algorithm 1. Notably, SACE presents a sinusoidal encoding algorithm to represent the primary structures. As shown in Figure 13, this algorithm utilizes the phase to describe the structure type and uses the period to control the representation space size. The binarized sinusoidal function is then defined as the structure code. In this way, SACE can generate unique and informative codes of arbitrary length for different structures.

Network Architecture For fair comparisons, we consider two configurations of different scales for the recognizer. The first configuration uses ResNet50 [8] as the backbone, paired with a transformer decoder that includes two multi-head attention layers for sequence modeling. In this transformer decoder, the number of attention heads is set to 4, the model dimension is 1024, and attention loss is used as recognition loss.

The second configuration employs ResNet34 as the backbone, along with two BiLSTM [9] or BiGRU [5] layers, each with 512 hidden states, serving as the sequence model. Specifically, for handwritten recognition, we use BiGRU for sequence modeling, while BiLSTM is applied for other scenarios. A CTC decoder is implemented as the sequence decoder described in Equation (3) to align the predictions with the ground-truth labels during training.

For both configurations, we rectify the input images to ensure that text lines are horizontal and remove white padding from the top and bottom of the images to emphasize the text. Additionally, two linear layers followed by

Algorithm 1 Structure Encoding in SACE

```

1: Input: index of nodes:  $i$ , index of structure types:  $s_i \in [0, 1, 2, \dots, 10]$ , max depth of the binary tree:  $D > 1$ .
2: Initialization: structure code  $\mathcal{I}_s = []$ 
3: for  $i \in \{1, 2, \dots, 2 \cdot (D-1) - 1\}$  do
4:    $n_i = \max(16 / (2 \cdot \lfloor \log_2(i) \rfloor), 1)$  # length
5:   if  $s_i = 0$  then # non-structure node
6:      $\mathcal{I}_s^i = \text{zeros}(n_i)$  # zero padding
7:   else
8:     if  $n_i \in [8, 16]$  then
9:        $t = [0, 1, \dots, n_i - 1]$ 
10:       $\mathcal{I}_s^i = \text{binarize}(\cos(32\pi(s_i + t)/n_i \cdot 2))$ 
11:    else if  $n_i = 4$  then # binary encoding
12:       $\mathcal{I}_s^i = \text{list}(\text{map}(\text{'int'}, \text{format}(s_i, \text{'04b'})))$ 
13:    else if  $n_i = 2$  then
14:      if  $s_i \in \text{Composite}$  then  $\mathcal{I}_s^i = [0, 1]$ 
15:      else if  $s_i \in \text{Surround}$  then  $\mathcal{I}_s^i = [1, 0]$ 
16:      else  $\mathcal{I}_s^i = [1, 1]$  # Overlaid
17:    else if  $n_i = 1$  then
18:       $\mathcal{I}_s^i = [1]$ 
19:    end if
20:  end if
21:   $\mathcal{I}_s \leftarrow \text{Concat}(\mathcal{I}_s, \mathcal{I}_s^i)$  # concatenating
22: end for
23: Output: structure code  $\mathcal{I}_s$ 

```

LeakyReLU activation are used as the linguistics and glyph decoder. Notably, the first configuration is only used for regular CTR evaluation, *i.e.*, MSA²,[†] reported in Table 1.

Description of Glyph Forms In this work, we consider 10 different types of glyphs: cursive (T1), artistic (T2), handwritten (T3), SimHei (*i.e.*, bold) (T4), SimSun (*i.e.*, SongTi) (T5), FangSong (T6), round (T7), KaiTi (T8), LiShu (T9), and others (T10). To enhance the diversity of glyph styles, we collect 7 variants for each type, as presented in Table 8. The glyph from each variant is rendered on a black background with a size of 96×96 . Notably, some complex Chinese characters cannot be accurately rendered using these specific forms. Therefore, to ensure the robustness of the glyph representations generated through GCCL, only glyphs that can be rendered with these forms are considered in calculating the pseudo-representation, as in Equation (8).

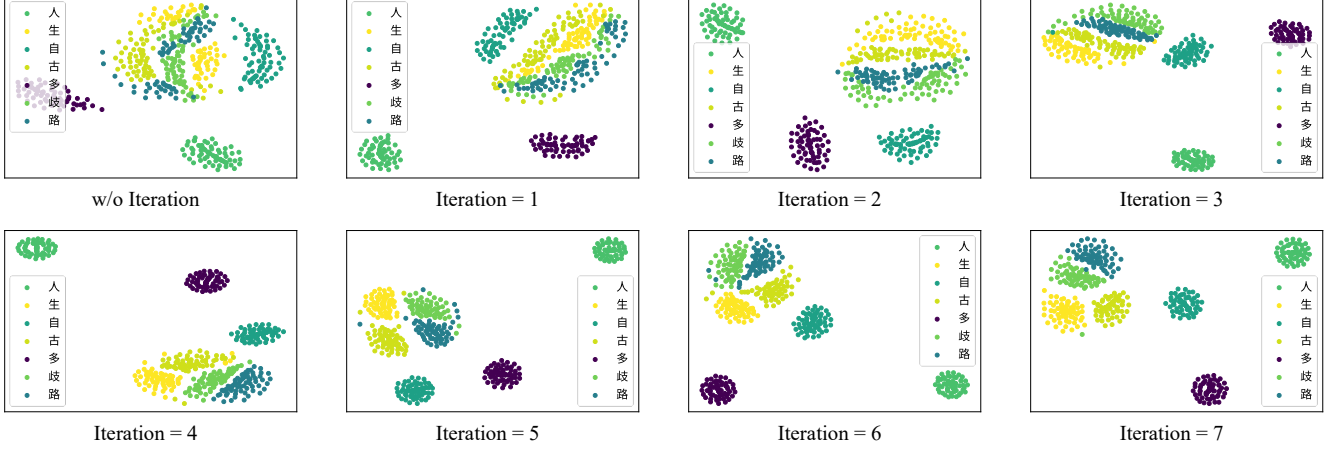


Figure 14. Glyph embedding distribution visualization with varying iteration of the GCCL.

B. Ablation Study

In this section, we present two additional ablation experiments, focusing on the structure encoding algorithm and the radical codes to validate their effectiveness.

Influence of Structures Encoding Algorithm To validate the effectiveness of the introduced sinusoidal encoding algorithm, we compare it with two widely used encoding algorithms: binary encoding and random multi-hot encoding. As shown in Table 6, our proposed sinusoidal encoding algorithm achieves the best result. This is likely due to the fact that random multi-hot encoding produces complex patterns, making predictions by the linguistic decoder more challenging. Additionally, when the representation space is much larger than $\lceil \log_2(10) \rceil$, the binary encoding generates sparse codes containing many zero-padding bits, which negatively impacts their expressive ability.

Influence of Radical Code We explore various combinations of the number of encoded radicals (R) and encoding length (L_R). Additionally, we generate a random radical code of equal length for each character as a baseline, which cannot express any radical information. As shown in Table 7, random encoding leads to a significant decrease in recognition performance compared to encoding based on character radicals, highlighting the importance of radicals for recognition. Furthermore, we observe that preserving too few radicals (e.g., $R = 12$) substantially degrades the recognition performance of the CTR model due to excessive loss of radical information. Meanwhile, since most characters do not contain many radicals, considering too many radicals increases ineffective padding bits within the linguistic representation, thereby decreasing recognition performance (as discussed in Section 4.5). Moreover, shorter encoding lengths (e.g., $L_R = 36$) slightly decline the recognition performance due to their limited expressive ability. As the encoding length increases, recognition performance

Encoding Strategy	AR	RR
Binary Encoding	97.07	94.78
Random multi-hot Encoding	97.22	94.90
Sinusoidal Encoding	97.35	95.12

Table 6. Ablation study on encoding algorithm of structures in terms of accurate rate (%) (AR) and recall rate (%) (RR).

R	L_R	AR	RR
Random Code		95.81	94.65
16	36	97.12	94.92
16	48	97.25	95.08
16	60	97.35	95.12
16	72	97.33	95.11
12	60	97.04	94.87
20	60	97.26	95.04

Table 7. Ablation study on the numbers of encoded radical nodes, i.e., R and the length of the radical codes, i.e., L_R in terms of accurate rate (%) (AR) and recall rate (%) (RR).

gradually improves and approaches saturation. Ultimately, we selected $R = 16$ and $L_R = 60$ as the standard settings to balance performance and computational efficiency.

C. Visualization

To further visualize the effectiveness of GCCL, we sample seven characters and visualize their glyph embedding distribution with different iterations in a 2-D space using t-SNE, where each character class is represented by a distinct color. As shown in Figure 14, as iterations progress, the glyph embeddings become increasingly discriminative, and the number of outliers decreases. Consequently, the pseudo-representations derived from these embeddings become more representative, strongly demonstrating the effectiveness of GCCL.

Type	No.	Forms	Samples	Type	No.	Forms	Samples
T1	1	F101255	欲把西湖比西子	T6	36	F101228	欲把西湖比西子
	2	F103112	欲把西湖比西子		37	F101170	欲把西湖比西子
	3	F101104	欲把西湖比西子		38	F101282	欲把西湖比西子
	4	F101140	欲把西湖比西子		39	F100714	欲把西湖比西子
	5	F100582	欲把西湖比西子		40	F100568	欲把西湖比西子
	6	F102844	欲把西湖比西子		41	F103164	欲把西湖比西子
	7	F102836	欲把西湖比西子		42	F100333	欲把西湖比西子
T2	8	F101144	欲把西湖比西子	T7	43	F102457	欲把西湖比西子
	9	F101245	欲把西湖比西子		44	F100573	欲把西湖比西子
	10	F102391	欲把西湖比西子		45	F101295	欲把西湖比西子
	11	F101274	欲把西湖比西子		46	F101112	欲把西湖比西子
	12	F100099	欲把西湖比西子		47	F102530	欲把西湖比西子
	13	F100583	欲把西湖比西子		48	F101207	欲把西湖比西子
	14	F102804	欲把西湖比西子		49	F101213	欲把西湖比西子
T3	15	F102427	欲把西湖比西子	T8	50	F101103	欲把西湖比西子
	16	F100542	欲把西湖比西子		51	F102338	欲把西湖比西子
	17	F102840	欲把西湖比西子		52	F102820	欲把西湖比西子
	18	F101155	欲把西湖比西子		53	F102862	欲把西湖比西子
	19	F100403	欲把西湖比西子		54	F103216	欲把西湖比西子
	20	F100413	欲把西湖比西子		55	F101154	欲把西湖比西子
	21	F101100	欲把西湖比西子		56	F102942	欲把西湖比西子
T4	22	F103287	欲把西湖比西子	T9	57	F102819	欲把西湖比西子
	23	F101216	欲把西湖比西子		58	F103225	欲把西湖比西子
	24	F101105	欲把西湖比西子		59	F102976	欲把西湖比西子
	25	F101279	欲把西湖比西子		60	F101260	欲把西湖比西子
	26	F103260	欲把西湖比西子		61	F103121	欲把西湖比西子
	27	F100706	欲把西湖比西子		62	F102403	欲把西湖比西子
	28	F101261	欲把西湖比西子		63	F101259	欲把西湖比西子
T5	29	F101253	欲把西湖比西子	T10	64	F100587	欲把西湖比西子
	30	F101190	欲把西湖比西子		65	F103263	欲把西湖比西子
	31	F102806	欲把西湖比西子		66	F100170	欲把西湖比西子
	32	F101183	欲把西湖比西子		67	F101133	欲把西湖比西子
	33	F100698	欲把西湖比西子		68	F100001	欲把西湖比西子
	34	F102323	欲把西湖比西子		69	F102398	欲把西湖比西子
	35	F102324	欲把西湖比西子		70	F102807	欲把西湖比西子

Table 8. Visualizations of the 70 forms of glyphs employed in this work through the ancient Chinese poem ‘欲把西湖比西子’. The font files are available at: <https://www.51miz.com/fonts/>.