A Tree-Structure Analysis Network on Handwritten Chinese Character Error Correction

Yunqing Li, Jun Du, *Senior Member, IEEE*, Jianshu Zhang, and Changjie Wu

***Abstract*—Existing researches on handwritten Chinese characters are mainly based on recognition network designed to solve the complex structure and numerous amount characteristics of Chinese characters. In this paper, we investigate Chinese characters from the perspective of error correction, which is to diagnose a handwritten character to be right or wrong and provide a feedback on error analysis. For this handwritten Chinese character error correction task, we define a benchmark by unifying both the evaluation metrics and data splits for the first time. Then we design a diagnosis system that includes decomposition, judgement and correction stages. Specifically, a novel tree-structure analysis network (TAN) is proposed to model a Chinese character as a tree layout, which mainly consists of a CNN-based encoder and a tree-structure based decoder. Using the predicted tree layout for judgement, correction operation is performed for the wrongly written characters to do error analysis. The correction stage is composed of three steps: fetch the ideal character, correct the errors and locate the errors. Additionally, we propose a novel bucketing mining strategy to apply triplet loss at radical level to alleviate feature dispersion. Experiments on handwritten character dataset demonstrate that our proposed TAN shows great superiority on all three metrics comparing with other state-of-the-art recognition models. Through quantitative analysis, TAN is proved to capture more accurate spatial position information than regular encoder-decoder models, showing better generalization ability.**

***Index Terms*—Handwritten Chinese character error correction, CNN, tree-structure analysis network, triplet loss, quantitative analysis.**

I. Introduction

LEARNING Chinese characters is a difficult task for children, due to the huge variety of existing Chinese characters and the complex internal structure of them. Thus mistakes are easy to make when elementary school students learn to write. In response to this scenario, the task of handwritten Chinese character error correction (HCCEC) came into being. We define the wrongly written characters as misspelled characters and the rightly written ones as right characters. The goal of HCCEC is twofold: to assess the correctness of the handwritten characters and to correct the misspelled ones, which we call Assessment and Correction for short. The Assessment subtask means to determine whether a given handwritten isolated character is correctly written. The Correction subtask means to locate the specific errors in character and correct them, which requires sophisticated analysis ability of models.

The misspelled characters are quite similar to the right ones, only differing in some details. Since the misspelled characters can be various and countless, we roughly summarize the character errors into three categories: stroke-level error, radical-level error and structure disorder. Stroke-level error means addition, deletion or misuse of one stroke; radical-level error means addition, deletion or misuse of some radicals; structure disorder means correct radicals composed in the wrong structural order. As shown in Fig. 1, we list a few examples of three kinds of misspelled characters. Considering the errors can occur in radical or structure, the diagnosis model needs to have the ability of modeling the internal primitives of Chinese characters, not treating the character as a whole.

Unlike the commonly used characters, the misspelled characters samples are extremely rare and the categories are unpredictable and countless. We assume the samples in training set to be right characters and expect the transfer learning ability of model to handle unseen misspelled characters. Moreover, the characters to be diagnosed can be correct-spelled or misspelled, so the task of HCCEC can be attributed to a generalized zero-shot learning problem (GZSL). GZSL is an extension of zero-shot learning, extending the test set labels to both seen and unseen classes, which makes it more challenging and realistic [1].

Compared with the handwritten Chinese character recognition (HCCR) task [2][3], the challenges of HCCEC are mainly reflected in the following three aspects: 1. The test set simultaneously contains seen and unseen classes, which sets a higher demand on the generalization ability of models. 2. The misspelled characters could be quite similar to the right ones and the training set only contains the right characters, which can cause strong bias [4] problem i.e., instances of misspelled characters (unseen classes) are more likely to be misclassified as one of the seen classes. 3. Other than assessing the correctness, the HCCEC task has a follow-up subtask, that is to correct errors, which has never been discussed in HCCR.

In this paper, we present a diagnosis system to solve the HCCEC task, which is composed of decomposition, judgement and correction stages. Given an image of Chinese character, the decomposition stage produces its radical expression. With

Manuscript received 15 November 2021; revised 23 February 2022; accepted 16 March 2022. Date of publication 31 March 2022; date of current version 8 September 2023. The Associate Editor coordinating the review of this manuscript and approving it for publication was Prof. Xiaochun Cao. *(Corresponding author: Jun Du.)*

Yunqing Li, Jun Du, and Changjie Wu are with the National Engineering Research Center of Speech and Language Information Processing, University of Science and Technology of China, Hefei 230026, China (e-mail: lyq123@mail.ustc.edu.cn; jundu@ustc.edu.cn; wucj@mail.ustc.edu.cn).

Jianshu Zhang is with AI Research, iFLYTEK, Hefei 230088, China (e-mail: jszhang6@iflytek.com).

Digital Object Identifier 10.1109/TMM.2022.3163517

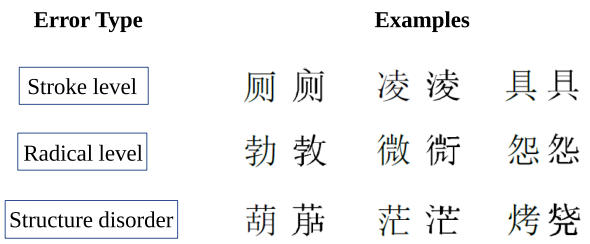

Fig. 1. Examples of three misspelled error types. The right one of each pair is the misspelled character and the left one is the corresponding right character.

the decomposition result, the correctness of character can be judged. For the misspelled chars, the following correction stage can provide detailed feedback on error locations and guidance on how to rectify the errors. Specifically, we propose a novel tree-structure analysis network (TAN) to decompose a character into radical-tree layout. The proposed TAN mainly consists of a CNN-based encoder and a tree-structure based decoder (TD), which can reduce the high dependence on contextual information of string decoder and alleviates the bias problem to a certain extent, showing better model generalization on misspelled characters. For more discriminative radical feature representation, we adopt bucketing triplet mining strategy and successfully apply triplet loss to Chinese character task in radical level. Moreover, probability-based decoding is employed, which is more flexible.

Considering the lack of literatures dedicated to HCCEC task, we also design three metrics to thoroughly measure model performance, i.e. F_1-score, accuracy and correction rate. Observing that there is no publicly dataset available for HCCEC task, we collect a large dataset that contains 401,400 handwritten samples for 5,500 common characters and 570 misspelled characters and annotate their character-level and radical-level labels. Since that we hope the diagnosis system can generate a feedback for error locations and corrections, the radical-level labels can be helpful.

The main contributions of this study are as follows:

1. We propose an integrated diagnosis system specifically for HCCEC task, which mainly consists of decomposition, judgement and correction stages.
2. We present a novel tree-structure analysis network for the task to alleviate the high dependance on contextual information and prove TAN captures better spatial information through quantitative analysis.
3. We propose a new bucketing mining strategy to apply triplet loss on radical for more discriminative feature. Additionally a new probability decoding method is also introduced to combine the strength of sequence decoding and metric learning.
4. We successfully apply several radical-based recognition models in this error correction task with designed post-processing. Experiment results show that our proposed system achieves the best performance on three metrics with an extra ability of correcting errors.
5. The source codes of our proposed TAN are available at https://github.com/yqingli123/TAN.

II. Related Works

The application requirement for HCCEC task has existed for a long time, but it has not attracted enough attention of academia, and we have not yet found any targeted model. So in this section, we describe the closely-related topics, including handwritten Chinese character recognition, Chinese grammatical error correction and generalized zero-shot learning.

A. Handwritten Chinese Character Recognition

Handwritten Chinese character recognition has received intensive attention since 1980s [5] considering the great diversity of handwriting styles and large number of character classes. Early works based on traditional methods mainly involves three procedures: pre-processing [6]–[9], feature extracting [10] and classifying [11], [12]. With the development of deep learning, mainly researches based on neural networks can be divided into character-based models and radical-based models. Character-based models treat recognition task as a classification problem. Ciresan [13], [14] proposed Multi-column deep neural networks, which was the first successful application of CNN on offline HCCR. Later, [15] proposed to combine traditional feature and GoogleNet [16] and achieved high accuracy that exceeded human performance. Since then, researches on character-based modeling have turned to other ideas, such as writer adaptation [17] and low computational cost [18], [19]. Different from character-based models, radical-based models treat a character into a combination of radicals. [20] first separated radicals in a recursive hierarchical scheme and proposed hierarchical radical matching to identify character. [21] detected position-dependent radicals with deep residual network. Since the proposal of encoder-decoder based models, there have been many applications, such as machine translation [22], mathematical expression recognition [23], [24], image captioning [25], [26], etc. At that time, Zhang *et al.* [27] proposed encoder-decoder based model RAN to decompose a Chinese character into a sequence of radicals and structures, which also achieved great success in HCCR [2]. This radical-level modeling methods have the ability of zero-shot learning, which is necessary to HCCEC task. FewshotRAN [28] further combined deep prototype learning for more robust feature extraction, but needed support samples of test set. Otherwise, RCN [29] treated a character as a vector of radical probability and radical numbers. Recently Cao *et al.* [3] proposed a hierarchical decomposition embedding method to represent a Chinese character with a semantic vector.

B. Chinese Grammatical Error Correction

Chinese grammatical error correction (CGEC) is an important task in natural language processing (NLP), aiming to detect and correct grammatical errors in Chinese texts. The errors discussed in CGEC task are 'real-word' error [30] appearing in texts or arrays, not 'non-word' error in handwritten splitted characters. Traditional works were mainly based on statistical language models and rules [31], [32] due to the lack of corpus. Since Yu *et al.* [33] organized a shared task on Chinese Grammatical Error Diagnosis (CGED), methods based on machine learning

have received more attention. They mainly categorized the errors into four types, including redundant words, missing words, bad word selection, disordered words. Huang and Wang [34] treated CGEC task as a sequence labeling problem and proposed a Bi-LSTM based model. For different errors, they simultaneously trained three Bi-LSTM models sharing word embeddings to solve them. Li *et al.* [35] proposed a two-stage hybrid system to not only detect but also correct the errors. The system included a BiLSTM-CRF model of detection and three grammatical error correction models to generate correction. Ren *et al.* [36] treated CGEC as a translation task that translated from "bad" Chinese texts to "good" texts by presenting a convolutional sequence-to-sequence model with attention mechanism. Recently, Liang *et al.* [37] applied BERT-fused Neural Machine Translation model on CGEC and achieved great performance.

C. Zsl and Gzsl

Zero-shot learning aims to recognize objects whose instances may not have been seen during training, thus tackles challenging problems: tedious human annotations and unseen testing categories. A strong restriction that the test samples only come from unseen classes makes ZSL task be criticized [1], [4], since the test set is unlimited in the realistic scenario. Comparing with ZSL, GZSL demonstrates to be more practical by including training classes in the search space [1]. GZSL aims at recognizing both seen and unseen classes, which is greatly consistent with our HCCEC task. Accordingly, Xian *et al.* [1] proposed a unified benchmark, including dataset splits and evaluation protocol, to provide a scientific standard for this area. The key to GZSL is to establish relations between seen and unseen classes through auxiliary information, such as attributes [38], word embeddings [39]. [40] proposed to learn latent representations for images and classes and convert sparse labels to continuous label embeddings. [41] proposed a co-representation network to learn a more uniform visual embedding space. Another research line is to solve the problem of training data imbalance by generating feature of unseen class. [42] proposed f-CLSWGAN to generate unseen feature conditioned on class-level semantic information. [43] utilizes meta-representation of each class for more authentic synthesized feature.

Considering the difference between HCCEC and HCCR task, radical-based recognition models can be transferred to this error correction task with post-processing but perform poorly (discussed in Section V). Encoder-decoder based model RAN has high dependance problem on contextual information, which causes severe bias on training classes. Metrics-based models [3] [29] directly encode an image of character into elaborate embedding and the intermediate steps of model lack interpretability, which cannot provide detailed information for correction stage. Inspired by the proposal of tree-structured decoder [24] in handwritten mathematical expression recognition, we model the Chinese character into radical-tree layout and combine the characteristics of characters to simplify the tree decoder. Our proposed TAN also combines the strength of encoder-decoder based models and metrics-based models, which has impressive performance and good model interpretability.

III. System Description

Here in this section, we propose a novel diagnosis system, which consists of three stages: decomposition, judgement and correction. The illustration of proposed system is shown in Fig. 2. In the decomposition stage, given a picture of a Chinese character (can be right or misspelled), the model generates a predicted tree layout. Through judgement, the decomposed tree will be sent to correction module if it is misspelled. Then a feedback of error location and candidate corrections will be produced.

A. Motivation

It is well-known that the primitives of characters include stroke, radical and structure. One or a few strokes can compose of a radical. One or a few radicals and structures listed in a certain order can compose of a character. As discussed in Section I, the character errors can be divided into three types. And the addition or deletion of strokes can be transformed into difference between similar radicals. Therefore, we model Chinese characters at radical level and utilize the shared radicals to transfer learning from right characters (seen classes) to misspelled characters (unseen classes).

Unlike English or Arabic characters, Chinese characters have intern structures. There are ten ways to decompose characters, leading to 10 structures [44]. In Fig. 3(a), the 10 structures are listed: (1) above-below, (2) top-left-surround, (3) left-right, (4) top-right-surround, (5) bottom-surround, (6) bottom-left-surround, (7) top-surround, (8) full surround, (9) left-surround, (10) overlaid. Thanks to these structures, all Chinese characters can be hierarchically decomposed to corresponding tree layout composed of radicals and structures [3][44]. As examples shown in Fig. 3(b), following the principle of decomposition from the whole to the part, character 'tiao' firstly belongs to a left-right structure. We set the structure as parent node and the left and right parts as two child nodes. Then the two substructures can be recursively decomposed until the child node is an indecomposable radical. As a result, each Chinese character can be decomposed into hierarchical tree layout composing of radicals and structures.

Through observation, we conclude three laws of the tree layouts:

1. All tree layouts are binary tree so that the relations between each parent-child pair are only left and right. That can greatly simplify the decoding process since the relations can be determined without special modeling.
2. All the parent nodes are structures and all leaf nodes are radicals. This rule makes it easy to determine when current branch ends during decoding. If current node belongs to structures, it has more child nodes and this tree branch needs following decoding. If current node belongs to radicals, it is the leaf node of tree, indicating the end of current branch.
3. The parent node can express the spatial relationship of its child nodes. For example, if the parent node is up-down structure, its two child nodes are spatially located above and below in the character. This gives us the inspiration to

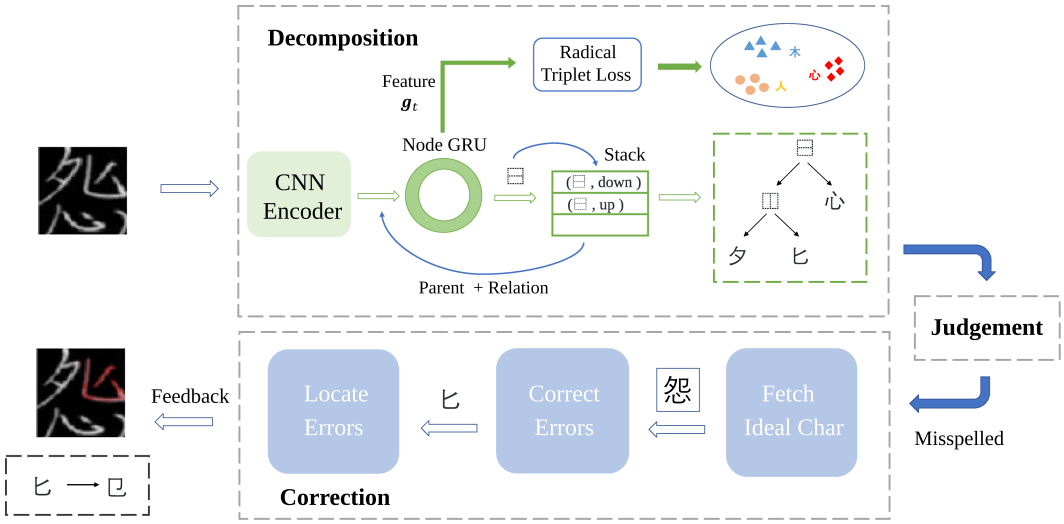

Fig. 2. The flow chart of our proposed diagnosis system, which contains three parts: decomposition, judgement and correction.

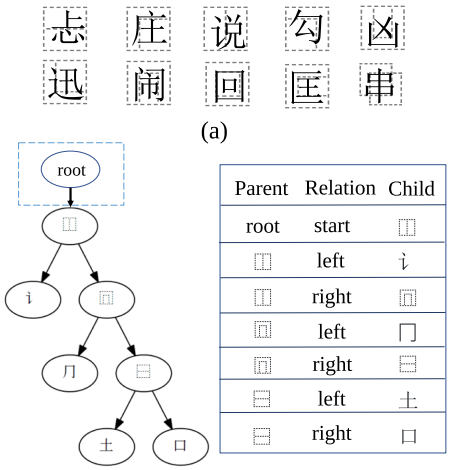

Fig. 3. (a) Graphical representation of ten radical structures. (b) Hierarchical tree layout of Chinese character 'tiao'. (c) The generated tree-structure label of character 'tiao'.

add spatial location guidance for attention of each parent-child pair.

Inspired by the above three laws, we innovatively propose to model each character into an exclusive tree layout directly, unlike [27], which serialized the tree structure into one-dimensional sequence. We fully exploit the location information that structures can provide and model Chinese characters from the perspective of two-dimensional structure, which have been ignored in previous studies [27]–[29].

B. Decomposition Model

Given an image of handwritten Chinese character, our proposed TAN decomposes the char into a tree layout. TAN mainly consists of a CNN-based encoder for feature extraction and a tree-structure decoder for character modeling. A guidance on spatial relation between parent and child nodes is proposed for better attention. Additionally, we propose bucketing triplet mining strategy to apply triplet loss on radical for more distinctive feature. Probability-based decoding is proposed which is more flexible than match decoding.

1) Tree-Structure Label: Each character can be represented into radical-tree layout like Fig. 3(b). Following the depth-first traversing order, the tree structure can be disassembled into an ordered sequence of parent-child node pairs, consisting of two nodes with their edge. As shown in Fig. 3(c), we list the transformed tree-structure label of character 'tiao'. The auxiliary symbol 'root' and relation 'start' are added for unified expression. The relation is the relative position of parent-child pair in the tree. Left-child node corresponds to relation 'left' and right-child node corresponds to relation 'right'. Each radical/structure in the radical-tree layout is treated as child node once and can be decoded at one step. From the tree-structure label table in Fig. 3(c), we can see all parent nodes except auxiliary symbol 'root' are structures, which are consistent with the law 2.

Hence the object of tree-structure decoder is converted to produce the ordered sequence of parent-child pairs. Among each pair, there are three elements needed to produce: parent, child, relation. Thanks to the first two laws, we can neatly model the tree-structure decoder into node prediction module and a stack. The stack is novelly utilized to store the decoded nodes and pop the accurate parent node with the help of the stack operation at each step. In parent-child pair, once the parent node is given, the relation can be easily surmised through law 1. So there is no need to model the relations specifically. In each time step, given the parent node and corresponding relation, we design a node prediction module for child node decoding.

2) CNN Encoder: Given an image I of character (right or misspelled), we first use a CNN-based encoder composed of densely connected convolutional layers in DenseNet [45] to extract high-level visual feature. We introduce the three-dimensional feature obtained from encoder as $\boldsymbol{B}$, whose size is $H \times W \times D$, where H denotes the height, W denotes the width,

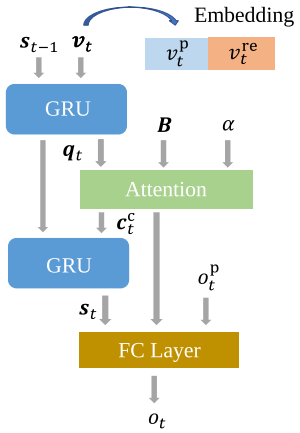

Fig. 4. The illustration of node prediction module.

D denotes the channels. The feature map $\boldsymbol{B}$ contains visual information of input, which contributes to latter decoding. Moreover, three-dimensional feature $\boldsymbol{B}$ can be flatten into the variable length of grid with size of $L \times D$, in which $L = H \times W$.

$$\boldsymbol{B} = \mathrm{CNN}(I) \tag{1}$$

$$\boldsymbol{B} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots \boldsymbol{b}_L\}, \boldsymbol{b}_i \in \mathbb{R}^D \tag{2}$$

3) Node Prediction Module: As discussed in Section III-B1, suppose we now have current parent node o_t^{p} and relation o_t^{re}, current child node is decoded by node prediction module. As shown in Fig. 4, the module mainly consists of a gated recurrent unit (GRU) [46] of two layers, attention module and a classifier. Through two embedding layers, high dimensional vectors of parent node o_t^{p} and the relation o_t^{re} are represented as $\boldsymbol{v}_t^{\mathrm{p}}$ and $\boldsymbol{v}_t^{\mathrm{re}}$. The concat of $\boldsymbol{v}_t^{\mathrm{p}}$ and $\boldsymbol{v}_t^{\mathrm{re}}$ works as the contextual information $\boldsymbol{v}_t$. D_m denotes the dimension of embedding.

$$\boldsymbol{v}_t^{\mathrm{p}} = \mathrm{Emb}_{\mathrm{p}}(o_t^{\mathrm{p}}), \boldsymbol{v}_t^{\mathrm{p}} \in \mathbb{R}^{D_m} \tag{3}$$

$$\boldsymbol{v}_t^{\mathrm{re}} = \mathrm{Emb}_{\mathrm{re}}(o_t^{\mathrm{re}}), \boldsymbol{v}_t^{\mathrm{p}} \in \mathbb{R}^{D_m} \tag{4}$$

$$\boldsymbol{v}_t = \mathrm{Concat}(\boldsymbol{v}_t^{\mathrm{p}}, \boldsymbol{v}_t^{\mathrm{re}}) \tag{5}$$

Through the first GRU layer, we compute the current query vector $\boldsymbol{q}_t$ from the previous hidden state $\boldsymbol{s}_{t-1}$ and current contextual information $\boldsymbol{v}_t$. Let D_n denotes the dimension of GRU decoder.

$$\boldsymbol{q}_t = \mathrm{GRU}_1(\boldsymbol{s}_{t-1}, \boldsymbol{v}_t) \tag{6}$$

Then we employ the coverage-based attention mechanism f_{att} [27], in which the visual feature $\boldsymbol{B}$ as the key and value and $\boldsymbol{q}_t$ as the query, to locate the related region of current step.

$$\boldsymbol{c}_t^{\mathrm{c}} = f_{\mathrm{att}}(\boldsymbol{q}_t, \boldsymbol{B}) \tag{7}$$

Specifically, in addition to visual feature $\boldsymbol{B}$ and query vector $\boldsymbol{v}_t$, the summation of former attention probabilities are also taken into consideration. The coverage vector $\boldsymbol{F}$ is employed to avoid problems with over-parsing (some radicals are decoded more than once) and under-parsing (some radicals are never decoded) [27].

$$\boldsymbol{F} = \boldsymbol{Q} * \sum_{l=1}^{t-1} \boldsymbol{\alpha}_l \tag{8}$$

$$e_{ti} = \boldsymbol{V}^{\top} \tanh(\boldsymbol{W}_{\mathrm{att}} \boldsymbol{q}_t + \boldsymbol{U}_{\mathrm{att}} \boldsymbol{b}_i + \boldsymbol{U}_{\mathrm{f}} \boldsymbol{f}_i) \tag{9}$$

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_k \exp(e_{tk})} \tag{10}$$

Let D_a denotes the attention dimension, D_f denotes the number of feature maps of filter $\boldsymbol{Q}$. Then $\boldsymbol{V} \in \mathbb{R}^{D_a}$, $\boldsymbol{W}_{\mathrm{att}} \in \mathbb{R}^{D_a \times D_n}$, $\boldsymbol{U}_{\mathrm{att}} \in \mathbb{R}^{D_a \times D}$, $\boldsymbol{U}_{\mathrm{f}} \in \mathbb{R}^{D_a \times D_f}$.

Thus we can obtain the attention coefficients α_{ti}, which denotes the attention score of feature vector $\boldsymbol{b}_i$ at step t. Then current context vector $\boldsymbol{c}_t^{\mathrm{c}}$ is computed by the weighted summation of all feature vectors.

$$\boldsymbol{c}_t^{\mathrm{c}} = \sum_{i=1}^{L} \alpha_{ti} \boldsymbol{b}_i \tag{11}$$

With the context vector $\boldsymbol{c}_t^{\mathrm{c}}$ and query vector $\boldsymbol{q}_t$, we can compute the current hidden state $\boldsymbol{s}_t$:

$$\boldsymbol{s}_t = \mathrm{GRU}_2(\boldsymbol{q}_t, \boldsymbol{c}_t^{\mathrm{c}}) \tag{12}$$

Finally, a classifier is utilized for node prediction :

$$\boldsymbol{g}_t = \boldsymbol{W}_{\mathrm{e}} \boldsymbol{v}_t^{\mathrm{p}} + \boldsymbol{W}_{\mathrm{s}} \boldsymbol{s}_t + \boldsymbol{W}_{\mathrm{c}} \boldsymbol{c}_t^{\mathrm{c}} \tag{13}$$

$$\boldsymbol{p}_t = \mathrm{softmax}(\boldsymbol{W}_{\mathrm{out}}(\mathrm{maxout}(\boldsymbol{g}_t))) \tag{14}$$

Let D_r and M denotes the dimension of radical representation and the size of radical dictionary. Then $\boldsymbol{W}_e \in \mathbb{R}^{D_r \times D_m}$, $\boldsymbol{W}_s \in \mathbb{R}^{D_r \times D_n}$, $\boldsymbol{W}_c \in \mathbb{R}^{D_r \times D}$, $\boldsymbol{W}_{\mathrm{out}} \in \mathbb{R}^{M \times D_r}$. $\boldsymbol{g}_t$ denotes the feature vector of t-th node o_t, which contains the most representive information of current radical. $\boldsymbol{p}_t$ denotes the predicted probability distribution of step t.

Suppose the one-hot label of t-th step is $\boldsymbol{y}_t$, the child node classification loss is computed as :

$$L_{\mathrm{c}} = -\sum_t \log \boldsymbol{p}_t \, \boldsymbol{y}_t \tag{15}$$

4) Spatial Relation Guidance: As the third law discussed in Section III-A, the parent nodes are all structures and express the spatial relationship between parent and child nodes. For example, the parent node is an above-below structure, meaning that its left child node is above it in space, and its right child node is below it. Considering this rule, we propose a spatial relation guidance on parent-child node pair. As the context vector $\boldsymbol{c}_t$ contains both the contextual information and spatial information, we utilize them to predict the spatial relation between parent and child nodes.

$$\boldsymbol{p}_t^{\mathrm{re}} = \mathrm{softmax}(\boldsymbol{W}_{\mathrm{re}}[\boldsymbol{c}_t^{\mathrm{c}}, \boldsymbol{c}_t^{\mathrm{p}}]) \tag{16}$$

Let M_{re} denotes the size of spatial relation dictionary, then $\boldsymbol{W}_{\mathrm{re}} \in \mathbb{R}^{M_{\mathrm{re}} \times 2D}$. $\boldsymbol{c}_t^{\mathrm{p}}$ denotes the context vector of parent node.

Given the ground-truth spatial relations $\boldsymbol{y}_t^{\text{sre}}$, the guidance loss can be computed as:

$$L_{\text{g}} = -\sum_t \log \boldsymbol{p}_t^{\text{re}}\, \boldsymbol{y}_t^{\text{sre}} \tag{17}$$

5) Radical Triplet Loss: During decoding, the feature vector $\boldsymbol{g}_t$ of radical r is calculated in Eq.(13). Ideally, the distance between features of the same radical should be smaller than those of different radicals. However, influenced by writing styles and radical positions (detailed discussed in Section VI-A), the feature space is relatively chaotic. So in this section, we introduce triple loss on Chinese characters at the radical level for the first time. We impose constraints on the feature vector to make the intra-class distance closer than the inter-class distance. The target is as follows:

$$||E_x - E_p|| + m < ||E_x - E_n|| \tag{18}$$

(x, p, n) masks a triplet. E_x is the anchor feature of one radical. E_p is the positive feature from the same radical and E_n is the negative feature from other radical. m denotes a margin that increases the intra-class distance.

We apply online mining method [47] to generate triplets in each mini-batch N. Considering each Chinese character c is composed of several radicals $(r^1, r^2, \ldots, r^n)$, it is a problem to ensure each radical sample r_i^k ($i \in [1, N]$) has at least one positive and one negative sample. Moreover, the inter-class distance is caused by both radical positions and writing styles. Accordingly we propose a novel bucketing mining strategy.

We first pack all Chinese characters containing a certain radical r into a bucket and there will be M radical buckets (M denotes the size of radical dictionary). Note that one character can appear in several buckets at the same time. In each mini-batch, we randomly select P characters in a random bucket, then randomly select K samples of each character. In this way, we can ensure the same radical from different characters and different writers can be both considered. There will be $K\sum_{i=1}^{P} R_i$ radical samples in each batch. R_i denotes the radical number of the i-th character. Hence, for each radical sample, there will always be at least PK positive samples and at most $K\sum_{i=1}^{P} R_i - PK$ negative ones. We use batch-hard [48] method, choosing the hardest positive sample and the hardest negative one to compose a triplet for each radical sample. The triplet loss is calculated as :

$$L_t = \sum_{i=1}^{K\sum_{j=1}^{P} R_j} \left[\max_p D(E_i, E_p) + m - \min_n D(E_i, E_n)\right] \tag{19}$$

where $D(x, y) = (x - y)^2$, which denotes the Euclidean distance between x and y.

6) Probability-Based Decoding: Instead of matching the output sequence/tree with character-radical lexicon, we propose a probability-based decoding method since matching decoding requires strict consistence between prediction and groundtruth. Our proposed probability-based decoding method is based on metric learning and relaxes the requirements.

Through node prediction module, we can get the predicted child node sequence $(o_1, o_2, \ldots, o_T)$ and the corresponding probabilities $(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_T)$. Since $\boldsymbol{p}_i$ represents the probability distribution of all radicals at current i-th step, together with the depth information of current node, we perform a weighted summation on T probability vectors, yielding the probability embedding $\boldsymbol{w}$ of predicted character.

$$\boldsymbol{w} = \sum_{i=1}^{T} \boldsymbol{p}_i * \alpha^{d_i} \tag{20}$$

where α denotes the weight decay with the increase of depth. We restrict $\alpha < 1.0$ since as the depth gets larger, the influence on the whole character gets smaller. d_i denotes the depth of i-th node.

Similarly, we can preprocess labels of all characters into embeddings $\boldsymbol{W}^{\text{gt}}$ once given the radical-level label. We can adopt one-hot embedding $\boldsymbol{y}^{\text{oh}}$ for each radical. Thus the label embedding of character c can be computed as follows:

$$\boldsymbol{w}_c^{\text{gt}} = \sum_{i=1}^{T} \boldsymbol{y}_i^{\text{oh}} * \alpha^{d_i} \tag{21}$$

Through calculating the Euclidean distance between the predicted $\boldsymbol{w}$ and all label embeddings $\boldsymbol{W}^{\text{gt}}$, we choose the one with the minimum distance as the predicted character c^{p}.

$$d(\boldsymbol{w}, \boldsymbol{w}_c^{\text{gt}}) = (\boldsymbol{w} - \boldsymbol{w}_c^{\text{gt}})^2 \tag{22}$$

$$c^{\text{p}} = \arg\min_{c \in C} d(\boldsymbol{w}, \boldsymbol{w}_c^{\text{gt}}) \tag{23}$$

where C denotes the set of all possible characters (including all right characters and misspelled characters).

C. Correction

HCCEC task is not only to assess whether the character is written correctly, but also to correct the errors. Next we introduce the error correction process of system, which mainly includes three steps.

Step I. Fetch the ideal char. After character decomposition, we need to figure out the candidate ideal characters first, i.e. which character the user intended to write. Considering that the characters are written without context, we select the top-5 characters as candidate ideal set. Given the output probability embedding $\boldsymbol{w}$ and the embeddings of all right characters set $\boldsymbol{W}_{\text{r}}^{\text{gt}} = \{\boldsymbol{w}^{\text{gt}}{}_1, \ldots, \boldsymbol{w}^{\text{gt}}{}_n\}$, we can calculate the Euclidean distance d among them using (22). Then the ideal chars are specified as the ones with 5 shortest distances.

Step II. Correct the errors. With the candidate ideal chars as reference, we can correct the misspelled char into the right one through edit distance computing. Specifically, the radical tree layout can be first serialized into a radical sequence by traversing in a depth-first order, as shown in Fig. 5. Then edit distance algorithm [49] is employed to transform the predicted sequence into ideal one with the shortest edit times. The computed edit operations can guide us to correct the errors. For example, as shown in Fig. 5, by deleting the two selected radicals, the predicted sequence can be corrected into right one.

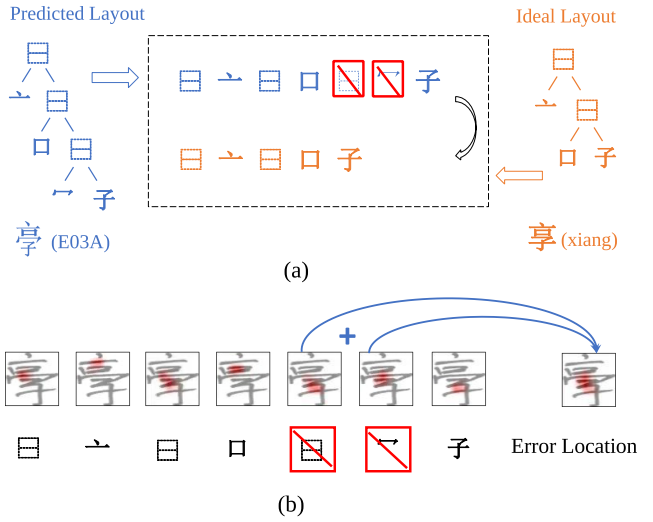

Fig. 5. (a) Illustration of correction operation. Left part in blue denotes predicted layout of misspelled char 'E03 A'. Right part in yellow denotes ideal layout of right char 'xiang'. (b) The illustration of error location.

Step III. Locate the errors. For writers, the errors can be difficult to detect since they have written wrongly. Thus the approximate location of error will be helpful. Thanks to the attention mechaism, our proposed TAN can roughly locate each radical during decoding. Combining the error radicals generated in Step II, we can locate the errors through attention visualization. For example, as shown in Fig. 5, with the correction operation of deleting the two radicals in Step II, we merge the attention map of these two steps as the error location.

IV. Task Setting

There are two sets of character classes: right characters and misspelled characters. Let $(I_1, a_1, y_1, y_1^c, L_1)$ $, \ldots, (I_N, a_N, y_N, y_N^c, L_N)$ be N samples. I_i denotes the i-th image. a_i denotes the i-th assessment label, which is 0 or 1, 0 means misspelled character, 1 means right character. y_i denotes the i-th character-level label of its exact category and L_i corresponds to its radical-level label. y_i^c denotes the i-th character-level correction label.

A. Dataset

Our dataset consists of 5,500 common characters and 570 misspelled characters, which are commonly written in primary schools. The annotations are all manually marked, including the correctness, character-level label and radical-level label. For misspelled characters, we also annotate their corresponding ideal characters. Among the 570 misspelled characters, there are 234 stroke-level errors, 320 radical-level errors and 16 structure-disorder errors according to the division method described in Section I.

Zero-shot learning assumes disjoint training and test classes. For better model performance without breaking the assumption, a separate validation set is prepared for tuning parameters. Following the dataset splits method in [1], we divide right characters into two disjoint parts: training classes and validation classes.

TABLE I
Statistics of Handwriting Dataset in Term of Number of Classes in Training, Validation and Test and Number of Images in Three Phases

Number of Classes			
Summary	Training	Validation	Testing
6070	5000	500	2000+570
Number of Images			
Summary	Training	Validation	Testing
401.4k	250k	100k	40K+11.4K

Each character in training set is written by 50 writers, making up 250,000 training samples. And each character in validation set is written by 200 writers, making up 100,000 validation samples. Since this is a GZSL problem, the testing set consists of right characters and misspelled characters. The 2,000 characters of right testing set are randomly selected from training classes. Each character in testing set is written by 20 writers, consisting of 40,000 right samples and 11,400 misspelled samples. The specific division is shown in Table I. In summary, there are 401.4 k samples made up of 6,070 characters.

B. Evaluation Metrics

Appropriate evaluation metrics can thoroughly and fairly reflect the performance of models. Next we introduce three metrics to evaluate the quality of models on two subtasks of HCCEC. The first one is F_1-score, a measure of prejudgement ability. The second one is accuracy, a fine measure of classification ability. The last one is correction rate, aiming to measure the error correction ability of models.

F_1 -score: In the assessment subtask, a given character needs prejudgement of correctness, right character or misspelled character. The prejudgement result can affect the following operation: misspelled character needs to be corrected but right character needs not. To measure the performance of prejudgement, we calculate the precision and recall of the two classes to get F_1-score respectively.

$$F_1 = \frac{2 * precision * recall}{precision + recall} \tag{24}$$

Accuracy: The metric F_1-score can reflect the prejudgement ability of model, but it is not refined. There are a large number of existing Chinese characters and misspelled characters are even countless. The classification ability of character category is of great significance. Additionally, the prediction of specific category for given character is also indispensable in the follow-up correction subtask. Therefore, we set the metric accuracy of right characters and misspelled characters to measure the classification performance of models.

Correction rate: After assessing stage, misspelled characters need correction operation. A measure of correction rate can intuitively reflect the the correction ability of model. To correct the error, models need to both correctly predict the character category and infer the ideal character (which character the user intends to write). Suppose there are totally N samples in test set of misspelled character. N_c denotes the samples which are correctly classified. N_i denotes the samples whose ideal character is correctly predicted. Then the correction rate CR can be

computed as:

$$CR = \frac{N_c \cap N_r}{N} \tag{25}$$

The metric of correction rate puts forward higher requirements on the models: not only need to classify a given character, but also need to infer the ideal character based on the model output.

V. Experiment Setting

A. Training

For training, our proposed model is optimized by three losses: child node classification loss L_{c}, relation guidance loss L_{g} and radical triplet loss L_{t}. The overall loss is shown as follows:

$$L = \lambda_1 L_{\mathrm{c}} + \lambda_2 L_{\mathrm{g}} + \lambda_3 L_{\mathrm{t}} \tag{26}$$

In experiment, we set $\lambda_1 = 1, \lambda_2 = 0.5, \lambda_3 = 0.1$.

To make a fair comparison, we uniformly use DenseNet as encoder in the following models. DenseNet mainly consists of three dense blocks and two transition layers. Each dense block contains 22 bottlenecks. There is a 1×1 convolution layer and a 3×3 convolution layer in the bottleneck, which are all followed by BatchNorm layer and Relu activation layer consecutively. To avoid overfitting, we add dropout layer with the rate of 0.2 in each bottleneck. The growth rate of each bottleneck is set to 24. Then between each two blocks, a transition module is set to reduce the channels of feature map, with the reduction of 0.5. For transition layer, we use 1×1 convolution followed by 2×2 average pooling. Before the first dense block, a 7×7 convolution layer with stride of 2 is performed. In decoder, the node prediction module consists of two-layer unidirectional GRUs and an attention module. Each GRU layer has 256 units. The embedding dimensions for parent node and relation are all set to 256. The attention vector dim is 512. All images are resized to 64×64. The sizes M and M_{re} of radical dictionary and spatial relation dictionary we used are respectively 413 and 22.

During training, we utilize Adadelta optimizer to train the proposed model. The learning rate is set to 1.5 and the weight-decay is 1e-4. The experiments are all conducted on two Nvidia Telsa V100 GPUs.

B. Inference

In inference, we novelly employ a stack to record the decoded tuple $(o_t^{\mathrm{p}}, o_t^{\mathrm{re}})$ and pop them at the right time for simplifying the decoding process. In one step, the decoded node guides the pushing operation. In the next step, accurate parent and relation information are popped for child node decoding. As illustrated in Fig. 6, we show the detailed decoding process of an example character 'yuan'. In the first step, given the parent node '<s>' and relation 'start,' TAN decodes current node 'up-down,' which belongs to 10 predefined structures. Thus two tuples with 'up-down' as parent node are needed to push in stack. In the second step, we pop the tuple at the top of stack to decode current node 'left-right'. In the same way, two tuples with 'left-right' as parent node are pushed in stack. We can repeat the operation utill the stack is empty. During decoding, the radical tree can also be built.

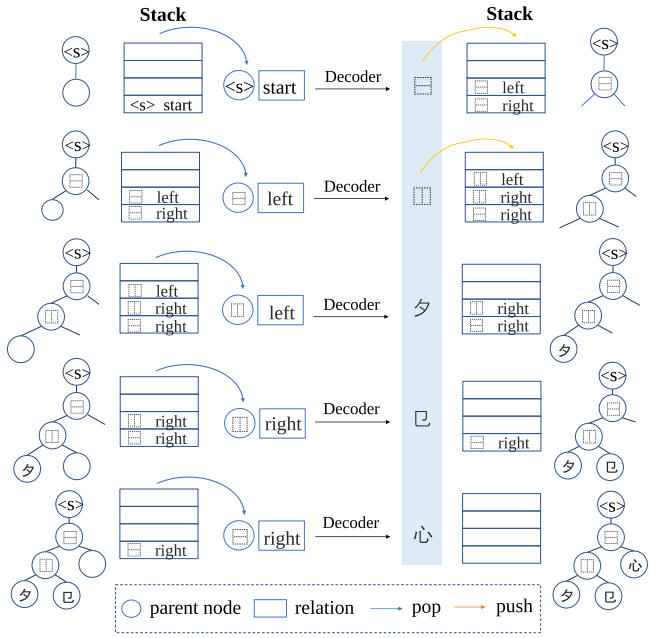

Fig. 6. The inference process of decoding character 'yuan,' along with the change of stack and the process of building radical tree.

Given an input image x and the candidate label Y, with the decoding of tree layout, the probability vector of each step can be generated. Following Eqs (22) and (23), we can compute the distance between predicted probability embedding and all possible label embeddings, classifying the input into the one with the closest distance.

VI. Experiments

Following the data splits as introduced in Section IV, we conduct experiments of handwritten Chinese character error correction on our collected handwritten dataset.

A. Ablation Study

The high-dimensional radical representations $\boldsymbol{g}_t$ can be reduced to 2D with LargeVis algorithm [50]. We choose 5 radicals, each with 500 character samples from at least 5 characters. The 2D-features of 2500 samples can be visualized. From the upper figure of Fig. 7, we have two observations: 1. The distance between samples of the same radical is sometimes greater than the distance between samples of different radicals. 2. The unexpected large inter-class distance is mainly caused by two factors: radical position and writing styles. Specifically, we can see points in orange are scattered in various positions, whose distances are even greater than distances between some blue points and them, which leads to the first obversation. Focusing on the samples of radical "gong," we zoom in orange points in the dashed box. In the zoomed figure with yellow background, points of one color represent features of radical "gong" from one character. It can be seen that generally points with the same color tend to cluster together, which means features of radical from the same character have high similarity. Besides, some purple points are far away from the center of purple points cluster, which means the same radical from different writers also generates diverse feature representations. That confirms the second

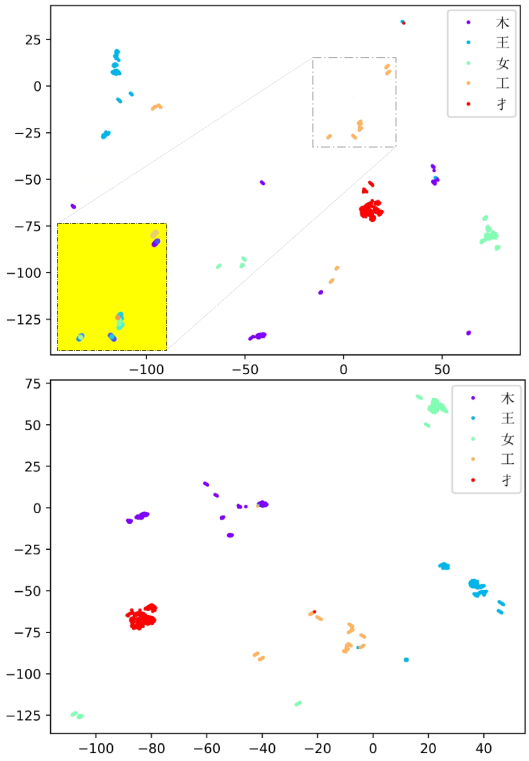

Fig. 7. Upper: Features of 5 radicals from 500 samples without triplet loss. Below: Features of 5 radicals from 500 samples with triplet loss. Points with the same color represent samples of the same radical.

TABLE II
PERFORMANCE COMPARISON OF DIFFERENT COEFFICIENTS OF TRIPLET LOSS WITH METRIC ACCURACY. VS=VALIDATION SET, RTS=RIGHT TESTING SET, MTS=MISSPELLED TESTING SET. λ_3=0. IS THE SETTING OF TAN WITHOUT TRIPLET LOSS. λ_3=0.1 IS OUR DEFAULT SETTING IN TAN

λ_3	0	0.01	**0.1**	0.3	0.5	0.7	0.9	1
VS	75.7	75.9	**76.0**	75.9	74.7	75.0	73.4	72.9
RTS	94.3	94.3	**94.6**	94	94.1	93.6	93.6	94.1
MTS	56.5	57.5	**58.0**	57.6	57.4	58.5	56.1	54.3

observation. Considering that, we propose radical triplet loss to constrain the radical features to achieve the effect of clustering the same class and increasing inter-class distance.

To show the effectiveness of radical triplet loss, we conduct ablation experiments on the validation set and verify the effect on the testing set. As shown in Table II, TAN outperforms its counterpart without triplet loss on right testing set by 0.3% and misspelled testing set by 1.5%. Additionally, the selection of hyperparameters has a great influence on the performance of TAN. At first, as λ_3 grows, the accuracy on three sets all increases. With $\lambda_3 = 0.1$, triplet loss works best on validation set, so that we select 0.1 as the optimal choice. From a comprehensive view, $\lambda_3 = 0.1$ also maximizes the performance gain on two test sets. As λ_3 continuously increases, the performance of model decreases, which is understandable since our main purpose is to classify and the triplet loss just works as an auxiliary role to cluster features. We also compare the performance of different margins as shown in Table III. With $m = 0.1$, the model can achieve the best performance on validation set, which is consistent with testing set.

TABLE III
PERFORMANCE COMPARISON OF DIFFERENT MARGINS IN TRIPLET LOSS WITH METRIC ACCURACY. ALL EXPERIMENTS ARE CONDUCTED WITH $\lambda_3 = 0.1$. $m = 0.1$ IS OUR DEFAULT SETTING IN TAN

	m=0.1	m=0.2	m=0.3
Validation Set	**76.1**	74.3	72.6
Right Testing Set	**94.6**	93.8	94.3
Misspelled Testing Set	**58.0**	58.2	52.6

TABLE IV
COMPARISON OF METRIC F_1-SCORE AMONG DIFFERENT MODELS. 'P' DENOTES PRECISION AND 'R' DENOTES RECALL

	Right Set			Misspelled Set		
	P	R	F_1	P	R	F_1
RCN	0.865	0.997	0.926	0.972	0.452	0.617
HDE	0.876	0.995	0.932	0.967	0.505	0.663
RAN	0.901	0.952	0.926	0.796	0.651	0.716
TAN	**0.897**	**0.997**	**0.944**	**0.98**	**0.60**	**0.744**

Additionally we visualize the radical feature with radical triplet loss. As shown in the bottom figure of Fig. 7, samples of the same radical from different characters are obviously more gathered and the distance between classes is also enlarged. Classification is the main purpose in our task and radical triplet loss only works as an auxiliary classification. Thus from the perspective of maximizing classification accuracy, the coefficient of triplet loss cannot be too large (as discussed in Table II), which causes the imperfect gathering effect.

B. Experiment Results of Assessment Subtask

1) Assessment With Metric F_1-Score: In this Section, we conduct experiments with the dataset introduced in Section IV and evaluate the models with the metric F_1-score. For comparison, we also implement a few recognition models in this subtask.

Metric-learning based models, such as RCN [29] and HDE [3], represent a character into a radical-based embedding. When adapting to HCCEC task, we can calculate the distance between the output embedding with all candidate character embeddings to generate the predicted character. If the predicted character belongs to the right character set, we judge it as right character. Otherwise, it is predicted as misspelled character.

Encoder-decoder based models (referring to RAN here) decode a character into a radical sequence, and look up the character-radical lexicon to determine the predicted character in recognition task. When adapting to HCCEC task, if the predicted radical sequence belongs to the candidate right character set, we judge it as the right character. Otherwise, it is predicted as a misspelled character.

Following their respective assessing methods, we compare the following four models: RCN, HDE, RAN and our proposed TAN. Please note that all the models have the same encoder and the image is resized to the same size for fair comparison. As shown in Table IV, TAN achieves the best performance on both right testing set and misspelled testing set in terms of metric F_1-score. Specifically, the precision on right testing set and the recall on misspelled testing set of RAN are slightly higher than TAN. That is because RAN decodes an unrestricted radical sequence and looks up the sequence in fixed character-radical

TABLE V
COMPARISON OF ACCURACY (%) AMONG DIFFERENT MODELS ON THREE ERROR TYPES

	Stroke	Radical	Structure	Misspelled Set	Right Set
RCN	29.4%	54.2%	0%	42.37%	92.4%
HDE	32.7%	59%	54.9%	47.20%	92.13%
RAN	31.5%	56%	33.8%	45.64%	92.3%
TAN	**44.2%**	**68.0%**	**62.2%**	**58.0%**	**94.6%**

TABLE VI
COMPARISON OF CORRECTION RATE (%) AMONG DIFFERENT MODELS ON THREE ERROR TYPES

	Stroke	Radical	Structure	CR
RCN	17.8%	26.3%	0%	22.1%
HDE	27.6%	36.1%	46.3%	32.9%
TAN	**33.3%**	**42.0%**	**51.8%**	**38.7%**

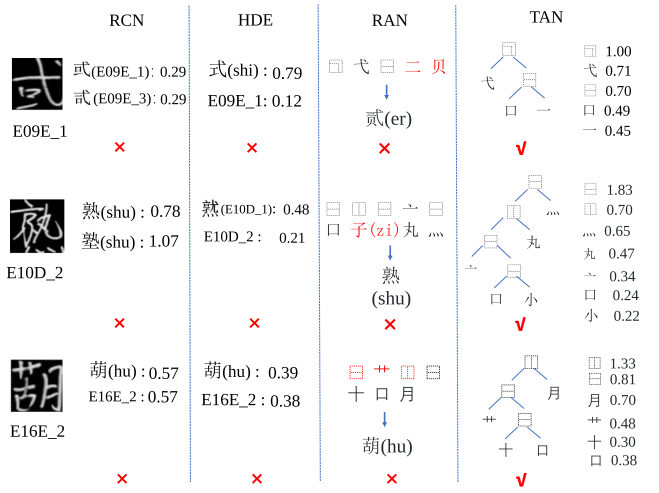

Fig. 8. The prediction results of four models on three exampled misspelled characters. The first character 'E09E_1' misses a stroke 'pie' and belongs to stroke-level error. The second character 'E10D_2' misuses a radical and belongs to radical-level error. The third character 'E16E_2' belongs to structure-disorder error.

lexicon to judge the correctness. It tends to predict into misspelled char. Accordingly, the precision on misspelled testing set and the recall on right testing set of RAN are significantly lower than other models. On the whole, the F_1-score on two testing sets is still lower than TAN.

2) Assessment With Metric Accuracy: Other than judging the correctness, the classification accuracy can better reflect the ability of models, as it requires accurate judgement of character category. Since the test set consists of two parts: right set and misspelled set, we separately calculate accuracy on the two sets. As shown in Table V, On right set, other three models achieve comparable performance, while TAN significantly exceeds them by about 2%. On misspelled set, TAN outperforms other models with great improvement of 10% $\sim$ 16%, which shows its strong generalization capacity.

Additionally, we further analyze the performance of models on three types of errors. As shown in Table V, comparing with the best-performing recognition model HDE, our proposed TAN achieves better performance with significant improvements: 11.5% on stroke-level error, 9% on radical-level error and 7.3% on structure disorder respectively. Among the three errors, the accuracy on stroke-level error is the lowest since stroke is the smallest unit and the most difficult to be detected. The performance on radical-level error of TAN can reach a high performance of 68%, which is closest to ordinary unseen characters.

To exploit how TAN performs best, we show some examples of misspelled characters and the corresponding predicted results of four models in Fig. 8. HDE directly encodes the input image into feature embedding containing radical information, whose detailed modeling capability is relatively weak. We list its top-2 predicted characters and the corresponding probability. In the first example, HDE wrongly classifies it into similar character 'shi' with a high probability of 0.79. In the second example, HDE wrongly finds another misspelled char 'E10D_1' as the most likely prediction. In the third example, character 'hu' and character 'E16E_2' are confusing for HDE even though their structures are slightly different. The string decoder of RAN relies heavily on contextual information so that the decoding results of first several steps easily mislead the following decoding. We list the decoding sequence of RAN in Fig. 8. RAN misrecognizes the first image into a similar, seen character 'er'. In the second sample, influenced by preceding decoding, the seventh radical is wrongly decoded into radical 'zi' as in seen character 'shu'. TAN decodes the input into a radical tree layout and the output probability of each radical is shown next to the tree. With more refined radical modeling and less reliance on contextual information, TAN shows better model generalization on unseen but similar characters.

C. Experiment Results of Correction Subtask

In this section, we show the correction performance of our system and list a few examples in detail from the perspective of error correction and error location.

1) Correction Rate: Following the three steps introduced in Section III-C, we will introduce the specific correction operation of each model in detail, and compare their correction rate on three error types.

Metric-learning based models, referring to RCN and HDE here, output an embedding vector to represent the character. For the sample judged as misspelled character, we can calculate the distance between the output embedding and the embeddings of all candidate right characters, and select the top-5 ones with the closest distances as the candidate ideal characters. Since the label embeddings are designed in complex rules, the specific error can not be inferred through the output embedding vector, not to mention locating them. RAN, basing on encoder-decoder framework, decodes a character into Ideographic Description Sequences (IDS). For misspelled characters, edit distance between predicted IDS and all candidate IDS of right characters can be calculated. However, the number of the right characters with the closest edit distance can be more than 5 so that the comparison with other methods is unfair. Therefore, the results of RAN are not listed in Table VI.

Since the correction subtask is a follow-up task to asessment, the correction rate is highly related with assessment results. Following the corresponding correction methods, we compare the correction rate of our proposed TAN with other models. As shown in Table VI, the correction rate of all models is reduced

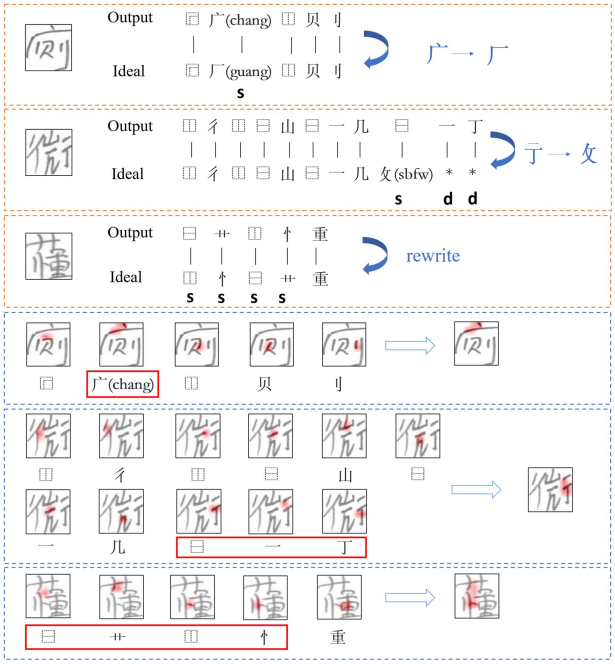

Fig. 9. Examples of error correction and error location.

compared with the accuracy in Table V, since correction operation requires correctly find the ideal characters additionally. But in term of overall correction rate, TAN still performs best, outperforming HDE by 5.8% and RCN by 16.6%. Among three error types, structure disorder is the easiest to correct for TAN while stroke-level error is the hardest with correction rate of 33.3%.

2) Correction Results: With the decoded tree layout, correction operation can be done following the three steps introduced in Section III-C. Here we show a few examples of correction results from two aspects: error correction and error location in Fig. 9.

The first example belongs to error of stroke level and it is actually caused by the confusion of two similar radicals "guang" and "chang". To correct, the user just needs to substitute "guang" with "chang," and the location of the mistaken raical is the error location. The second example belongs to radical-level error. To correct, the user needs to substitue the last three radicals with radical "sbfw". The summary of the three locations composes the error location. When encountering with structure-disorder error as shown in the third example, which is easy to identify since their radicals are all right but in wrong order, the users only need to follow the right order of ideal char to rewrite. And the error location covers almost the whole image so that the location misses its guidance function.

D. Quantitative Analysis

To deeply investigate how tree decoder shows stronger model generalization ability in misspelled characters, we conduct quantitative analysis calculation. During decoding, both string decoder and tree decoder first take the previous hidden state and previous output as input to produce the query vector:

$$s_t = \mathrm{RNN}(s_{t-1}, y_{\mathrm{f}}) \tag{27}$$

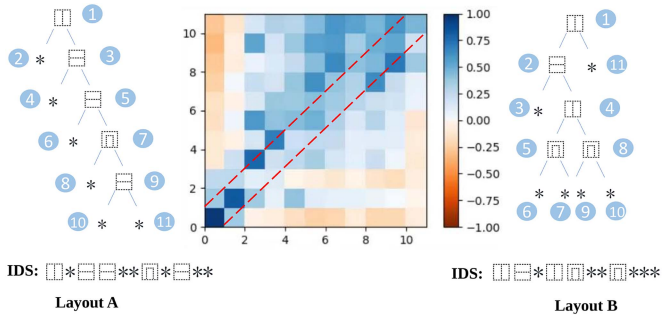

Fig. 10. Visualization of the averaged similarity matrix of characters with layout A and layout B. The x-axis and the y-axis indicate the position index in IDS, while the color indicates the averaged similarity. The number next to the radical represents the index of current radical.

where y_{f} can be y_{t-1} or $(y_{\mathrm{p}}, y_{\mathrm{re}})$ corresponding to string decoder and tree decoder. Then the query vector is utilized to calculate the attention map. s_t obviously contains contextual information, due to the memory mechanism of RNN. We further discover that it contains two-dimensional position information to assist current attention map computing.

According to [51], we first conduct the linear regression between queries (s_t) from model RAN and its positions (t) via fitting $t = W_r s_t + b_r$ on characters with the same tree layout and different tree layouts respectively (80% for training, 20% for test). We set indicator R^2 to reflect the proportion of the variance of the dependent variable that is predictable. For characters with the same tree layout, we get $R^2 = 0.95$. But for characters with different tree layouts of the same length, we get $R^2 = 0.70$. That suggests the position information of radical is related to its location in the tree layout. Although RAN recognizes a Chinese character as a linear sequence during modeling, the position information captured by RAN is no longer one-dimensional positions.

To further verify the position information contained by query vector is consistent with the tree layout, we compute the averaged cosine similarity S between the query feature vectors of the i-th and j-th time steps on characters with layout A and B:

$$S_{\mathrm{AB}}(i,j) = \cos\left(\frac{\sum_{m=1}^{|I_{\mathrm{A}}|} h_i^{m,\mathrm{A}}}{|I_{\mathrm{A}}|}, \frac{\sum_{n=1}^{|I_{\mathrm{B}}|} h_j^{n,\mathrm{B}}}{|I_{\mathrm{B}}|}\right) \tag{28}$$

where $I_{\mathrm{A}}, I_{\mathrm{B}}$ denotes all characters with layout A and layout B respectively. $h_i^{m,\mathrm{A}}$ denotes the query vector at i-th step of the m-th sample in I_{A}.

As shown in Fig. 10, focusing on the diagonal, the radicals of first two steps lie on the same position of layout A and B so that the cosine similarities are high. From step three to five, the similarity remains low since these radicals are in different positions to the two layouts. From that, we can conclude that RAN has captured the spatial information on the radical tree since structures and radicals have different physical meanings. But at the latter steps (>5), even though at different positions, the cosine similarity still remains at the level of 0.3 to 0.5, which is not low. That reflects the ability of RAN to capture spatial location decreases as the length gets longer.

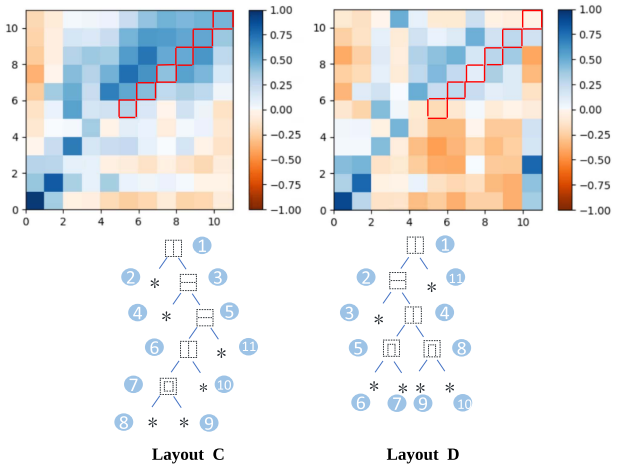

Fig. 11. Comparison of the averaged similarity matrix of string decoder and tree decoder. Left figure shows similarity matrix of string decoder. Right figure shows similarity matrix of tree decoder.

To specifically compare the position information learned by string decoder and tree decoder, we calculate the cosine similarity of characters with layout C and D. As shown in Fig. 11, in the first several steps (<5), both tree decoder and string decoder capture the position information that matches the position of the radical. But at the latter step (>5), the cosine similarity of string decoder starts to behave confusedly, while tree decoder still captures relatively accurate spatial position information. More accurate visual information and less dependance on context information results in good performance on misspelled character for tree decoder.

VII. Discussion

In this section, we discuss the challenges of HCCEC task and summarize the limitations of existing methods.

The challenges are mainly reflected in three aspects. 1) Since the assessment task belongs to GZSL problem and the training set only contains samples of right chars, there is naturally bias problem that performance on unseen classes tend to be poor. 2) The misspelled chars can be very similar with corresponding right chars. So the difficulty is higher than the recognition of regular unseen characters. Moreover, the task is based on handwritten scenario, so the writing styles can greatly influence the model's judgment of subtle difference between right and misspelled chars. 3) Without contextual information, correction subtask is difficulty since it is not reliable to find the ideal char only with the radical form.

Facing with the above challenges, the existing methods have two major limitations. First, the unseen class bias problem still remains and there is considerable room for improvement. Second, the performance of existing methods on stroke-level error is lower than that on other two errors. We speculate that radical-based modeling that all existing methods based is not precise enough, which may lead to the above phenomenon.

VIII. Conclusion and Future Work

In this paper, we design a diagnosis system with the ability of correcting errors for HCCEC task. A novel tree-structure analysis network is proposed to decompose a character into a radical-tree layout. Experiments on our collected dataset show our proposed method outperforms other recognition models on three metrics. Additionally, through quantitative analysis, we prove TAN can capture more accurate spatial information, showing better generalization ability.

In the future, we plan to introduce the idea of feature-generating paradigm in our model to further alleviate the bias problem. Additionally, we also plan to construct a new dataset of phrases or sentences to explore the effect of contextual information on HCCEC task.

References

[1] Y. Xian, B. Schiele, and Z. Akata, "Zero-shot learning-the good, the bad and the ugly," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4582–4591.
[2] W. Wang, J. Zhang, J. Du, Z.-R. Wang, and Y. Zhu, "DenseRAN for offline handwritten Chinese character recognition," in *Proc. 16th Int. Conf. Front. Handwriting Recognit.*, 2018, pp. 104–109.
[3] Z. Cao, J. Lu, S. Cui, and C. Zhang, "Zero-shot handwritten Chinese character recognition with hierarchical decomposition embedding," *Pattern Recognit.*, vol. 107, 2020, Art. no. 107488.
[4] W.-L. Chao, S. Changpinyo, B. Gong, and F. Sha, "An empirical study and analysis of generalized zero-shot learning for object recognition in the wild," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 52–68.
[5] R. Dai, C. Liu, and B. Xiao, "Chinese character recognition: History, status and prospects," *Front. Comput. Sci. China*, vol. 1, no. 2, pp. 126–136, 2007.
[6] T. Iijima, H. Genchi, and K.-I. Mori, "A theory of character recognition by pattern matching method," in *Learning Systems and Intelligent Robots*. Berlin, Germany: Springer, 1974, pp. 437–450.
[7] J. Tsukumo and H. Tanaka, "Classification of handprinted Chinese characters using nonlinear normalization and correlation methods," in *Proc. 9th Int. Conf. Pattern Recognit.*, 1988, pp. 168–169.
[8] H. Yamada, K. Yamamoto, and T. Saito, "A nonlinear normalization method for handprinted Kanji character recognition-line density equalization," *Pattern Recognit.*, vol. 23, no. 9, pp. 1023–1029, 1990.
[9] C.-L. Liu and K. Marukawa, "Pseudo two-dimensional shape normalization methods for handwritten Chinese character recognition," *Pattern Recognit.*, vol. 38, no. 12, pp. 2242–2255, 2005.
[10] C.-L. Liu, K. Nakashima, H. Sako, and H. Fujisawa, "Handwritten digit recognition: Benchmarking of state-of-the-art techniques," *Pattern Recognit.*, vol. 36, no. 10, pp. 2271–2285, 2003.
[11] F. Kimura, K. Takashina, S. Tsuruoka, and Y. Miyake, "Modified quadratic discriminant functions and the application to Chinese character recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 1, pp. 149–153, Jan. 1987.
[12] C.-L. Liu, H. Sako, and H. Fujisawa, "Discriminative learning quadratic discriminant function for handwriting recognition," *IEEE Trans. Neural Netw.*, vol. 15, no. 2, pp. 430–444, Mar. 2004.
[13] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3642–3649.
[14] D. Cireşan and U. Meier, "Multi-column deep neural networks for offline handwritten Chinese character classification," in *Proc. Int. Joint Conf. Neural Netw.*, 2015, pp. 1–6.
[15] Z. Zhong, L. Jin, and Z. Xie, "High performance offline handwritten Chinese character recognition using GoogLeNet and directional feature maps," in *Proc. 13th Int. Conf. Document Anal. Recognit.*, 2015, pp. 846–850.
[16] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.
[17] X.-Y. Zhang, Y. Bengio, and C.-L. Liu, "Online and offline handwritten Chinese character recognition: A comprehensive study and new benchmark," *Pattern Recognit.*, vol. 61, pp. 348–360, 2017.
[18] Z. Li, N. Teng, M. Jin, and H. Lu, "Building efficient CNN architecture for offline handwritten chinese character recognition," *Int. J. Document Anal. Recognit.*, vol. 21, no. 4, pp. 233–240, 2018.
[19] X. Xiao *et al.*, "Building fast and compact convolutional neural networks for offline handwritten chinese character recognition," *Pattern Recognit.*, vol. 72, pp. 72–81, 2017.

[20] A.-B. Wang and K.-C. Fan, "Optical recognition of handwritten Chinese characters by hierarchical radical matching method," *Pattern Recognit.*, vol. 34, no. 1, pp. 15–35, 2001.
[21] T.-Q. Wang, F. Yin, and C.-L. Liu, "Radical-based Chinese character recognition via multi-labeled learning of deep residual networks," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit.*, 2017, pp. 579–584.
[22] K. Cho *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014.
[23] J. Zhang, J. Du, and L. Dai, "Track, attend, and parse (TAP): An end-to-end framework for online handwritten mathematical expression recognition," *IEEE Trans. Multimedia*, vol. 21, no. 1, pp. 221–233, Jan. 2019.
[24] J. Zhang, J. Du, Y. Yang, Y.-Z. Song, and L. Dai, "SRD: A tree structure based decoder for online handwritten mathematical expression recognition," *IEEE Trans. Multimedia*, vol. 23, pp. 2471–2480, 2021.
[25] L. Li, S. Tang, Y. Zhang, L. Deng, and Q. Tian, "GLA: Global-local attention for image description," *IEEE Trans. Multimedia*, vol. 20, no. 3, pp. 726–737, Mar. 2018.
[26] L. Gao, Z. Guo, H. Zhang, X. Xu, and H. T. Shen, "Video captioning with attention-based LSTM and semantic consistency," *IEEE Trans. Multimedia*, vol. 19, no. 9, pp. 2045–2055, Sep. 2017.
[27] J. Zhang, Y. Zhu, J. Du, and L. Dai, "Radical analysis network for zero-shot learning in printed Chinese character recognition," in *Proc. IEEE Int. Conf. Multimedia Expo.*, 2018, pp. 1–6.
[28] T. Wang, Z. Xie, Z. Li, L. Jin, and X. Chen, "Radical aggregation network for few-shot offline handwritten Chinese character recognition," *Pattern Recognit. Lett.*, vol. 125, pp. 821–827, 2019.
[29] Y. Li, Y. Zhu, J. Du, C. Wu, and J. Zhang, "Radical counter network for robust Chinese character recognition," in *Proc. 25th Int. Conf. Pattern Recognit.*, 2021, pp. 4191–4197.
[30] K. Kukich, "Techniques for automatically correcting words in text," *ACM Comput. Surv.*, vol. 24, no. 4, pp. 377–439, 1992.
[31] L.-H. Lee *et al.*, "A sentence judgment system for grammatical error detection," in *Proc. COLING 25th Int. Conf. Comput. Linguistics: Syst. Demonstrations*, 2014, pp. 67–70.
[32] T. H. Chang, Y.-T. Sung, J. F. Hong, and J. I. Chang, "KNGED: A tool for grammatical error diagnosis of Chinese sentences," in *Proc. 22nd Int. Conf. Comput. Educ.*, 2014, pp. 48–55.
[33] L.-H. Lee, L.-C. Yu, and L.-P. Chang, "Overview of the NLP-TEA 2015 shared task for Chinese grammatical error diagnosis," in *Proc. 2nd Workshop Natural Lang. Process. Techn. Educ. Appl.*, 2015, pp. 1–6.
[34] S. Huang and H. Wang, "Bi-LSTM neural networks for Chinese grammatical error diagnosis," in *Proc. 3rd Workshop Natural Lang. Process. Techn. Educ. Appl.*, 2016, pp. 148–154.
[35] C. Li *et al.*, "A hybrid system for Chinese grammatical error diagnosis and correction," in *Proc. 5th Workshop Natural Lang. Process. Techn. Educ. Appl.*, 2018, pp. 60–69.
[36] H. Ren, L. Yang, and E. Xun, "A sequence to sequence learning for Chinese grammatical error correction," in *Proc. CCF Int. Conf. Natural Lang. Process. Chin. Comput.*, 2018, pp. 401–410.
[37] D. Liang *et al.*, "BERT enhanced neural machine translation and sequence tagging model for Chinese grammatical error diagnosis," in *Proc. 6th Workshop Natural Lang. Process. Techn. Educ. Appl.*, 2020, pp. 57–66.
[38] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, "Describing objects by their attributes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 1778–1785.
[39] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. ICLR Workshops Track*, 2013.
[40] H. Zhang *et al.*, "Online collaborative learning for open-vocabulary visual classifiers," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2809–2817.
[41] F. Zhang and G. Shi, "Co-representation network for generalized zero-shot learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7434–7443.
[42] Y. Xian, T. Lorenz, B. Schiele, and Z. Akata, "Feature generating networks for zero-shot learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 5542–5551.
[43] J. Li *et al.*, "Leveraging the invariant side of generative zero-shot learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7402–7411.
[44] J. Zhang, J. Du, and L. Dai, "Radical analysis network for learning hierarchies of Chinese characters," *Pattern Recognit.*, vol. 103, 2020, Art. no. 107305.
[45] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.
[46] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *Syntax, Semantics Struct. Statistical Transl.*, p. 103, 2014.
[47] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, 2015, pp. 815–823.
[48] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," 2017, *arXiv:1703.07737*.
[49] V. I. Levenshtein *et al.*, "Binary codes capable of correcting deletions, insertions, and reversals," *Sov. Phys. Doklady*, vol. 10, no. 8, 1966, pp. 707–710.
[50] J. Tang, J. Liu, M. Zhang, and Q. Mei, "Visualizing large-scale and high-dimensional data," in *Proc. 25th Int. Conf. World Wide Web*, 2016, pp. 287–297.
[51] X. Yue, Z. Kuang, C. Lin, H. Sun, and W. Zhang, "Robustscanner: Dynamically enhancing positional clues for robust text recognition," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 135–151.

Yunqing Li received the B.Eng. degree from the School of Electronic Engineering, Xidian University, Xi'an, China, in 2019. She is currently working toward the master's degree with the University of Science and Technology of China, Hefei, China. Her research interests include Chinese character recognition and mathematical expression recognition.

Jun Du (Senior Member, IEEE) received the B.Eng. and Ph.D. degrees from the Department of Electronic Engineering and Information Science, University of Science and Technology of China (USTC), Hefei, China, in 2004 and 2009, respectively. From 2004 to 2009, he was with iFlytek Speech Lab of USTC. During the above years, he was an Intern for two nine month periods with Microsoft Research Asia (MSRA), Beijing, China. In 2007, he was a Research Assistant for six months with the Department of Computer Science, The University of Hong Kong, Hong Kong. From July 2009 to June 2010, he was with iFLYTEK Research on speech recognition. From July 2010 to January 2013, he joined MSRA as an Associate Researcher, working on handwriting recognition, OCR, and speech recognition. Since February 2013, he has been with the National Engineering Laboratory for Speech and Language Information Processing, USTC.

Jianshu Zhang received the B.Eng. degree in 2015 from the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei, China, where he is currently working toward the Ph.D. degree. In 2018, he was a Visiting Student for six months with the Queen Mary University of London, London, U.K. His research interests include deep learning, handwriting mathematical expression recognition, Chinese document analysis, and speech analysis.

Changjie Wu received the B.Eng. degree from the School of Energy and Mechanical Engineering, Nanjing Normal University, Nanjing, China, in 2018, and the M.A.Sc. degree from the School of Cyber Science and Technology, University of Science and Technology of China, Hefei, China, in 2021. His research interests include neural networks, natural scene text recognition, and mathematical expression recognition.