





Improving Handwritten Mathematical Expression Recognition via Similar Symbol Distinguishing

Zhe Li , Xinyu Wang , Yuliang Liu , *Member, IEEE*, Lianwen Jin , *Member, IEEE*, Yichao Huang, and Kai Ding

Abstract—Handwritten mathematical expression recognition (HMER) is an essential task in the OCR community, which consists of two sub-tasks, i.e., symbol recognition and structure parsing. Modern literature treats HMER as a LaTeX sequence predicting problem that simultaneously recognizes symbols and parses the structures of MEs. Although deep learning-based HMER methods have been achieving promising results on public benchmarks, it is admitted that the misclassification error between visually similar symbols still prevents these approaches from more generalized scenes. In this paper, we try to solve this issue from three aspects. 1) We enhanced the feature extraction progress by introducing path signature features, which incorporates local writing details and global spatial information. 2) We developed a language model that uses contextual information to correct the symbols misclassified by vision-only-based recognition models. 3) We solved the misalignment problem in existing ensemble method by designing a dynamic time warping (DTW) based algorithm. By combining the above improvements, our method achieved state-of-the-art results on three CROHME benchmarks, outperforming previous methods by a large margin.

Index Terms—Handwritten mathematical expression recognition, path signature, language model, dynamic time warping, ensemble.

I. INTRODUCTION

HANDWRITTEN mathematical expression recognition (HMER) is a fundamental task among the OCR community, which has been attracting tremendous research interest in the past decades [1]. Recently, as the surge of deep learning, more and more HMER algorithms [2], [3], [4], [5], [6]

Manuscript received 24 September 2021; revised 31 January 2022 and 1 August 2022; accepted 14 March 2023. Date of publication 22 March 2023; date of current version 8 January 2024. This work was supported in part by NSFC under Grant 61936003, and in part by Zhuhai Industry Core and Key Technology Research Project under Grant 2220004002350. The Associate Editor coordinating the review of this manuscript and approving it for publication was Dr. Simone Milani. (Corresponding authors: Yuliang Liu; Kai Ding.)

Zhe Li is with the School of Electronics and Information, South China University of Technology, Guangzhou 510640, China (e-mail: zheli0205@foxmail.com).

Xinyu Wang is with the School of Computer Science, The University of Adelaide, Adelaide 5005, Australia (e-mail: xinyu.wang02@adelaide.edu.au).

Yuliang Liu is with the School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Hongshan 430074, China (e-mail: ylliu@hust.edu.cn).

Lianwen Jin is with the School of Electronics and Information, South China University of Technology, Guangzhou 510640, China (e-mail: lianwen.jin@gmail.com).

Yichao Huang and Kai Ding are with the IntSig Information Co., Shanghai 200433, China (e-mail: charlie_huang@intsig.net; danny_ding@intsig.net).

Digital Object Identifier 10.1109/TMM.2023.3260648

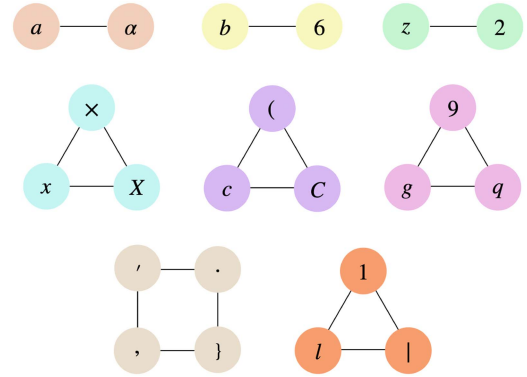


Fig. 1. Homoglyph and confusable symbols can easily lead to misclassification errors in mathematical expression recognition. Nodes of the same color represent a group of ambiguous symbols, e.g., the Latin ‘a’ shares a similar appearance with the Greek ‘ α ’.

benefit from deep neural networks, which have been achieving record-breaking performance on widely-used benchmarks. Benefiting from the promising results obtained by these methods, many HMER systems are developed for real-world applications, e.g., human-computer interaction, office automation, and intelligent education. Despite the well-admitted success, a dilemma still existing in the HMER is that current methods are prone to misclassify visually similar symbols (see Fig. 1).

The mainstream deep learning-based HMER methods can be categorized into two types based on the modeling scheme, i.e., sequence-based modeling and graph-based modeling. The former regards HMER as a sequence prediction task [3], [5], [6], [7], [8], [9], which examines Mathematical Expression (ME) features via Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) based encoders, and then directly decoding the features into corresponding LaTeX sequence. The latter forms the task as a graph or tree prediction problem [2], [4], [10]. Specifically, each symbol in the ME is considered as a node in the graph, while the relations between the symbols are edges. Thus, the Symbol Layout Tree (SLT) can be obtained by training a tree-based LSTM or Graph Neural Network (GNN). Although both methods have shown promising results in controlled environments, they still suffer from severe performance degradation due to several challenges, including the complex two-dimensional structure of the MEs, visually similar symbols, various handwriting styles, and the lack of training data.

The fact that recognition models are prone to misclassify the visually similar symbols, e.g., “x,” “X,” “ \times ,” has been repeatedly

identified in previous works and reports [11], [12], [13], [14]. As shown in Fig. 1, these symbols are confusing in visual, especially when combined with various font and writing styles. To address this issue, we propose a brand-new framework in three aspects, including *Feature Extraction Enhancement*, *Language Model Rectification*, and *Multi-Model Ensemble*.

Feature extraction enhancement: The form of online HMER data is a sequence of two-dimensional coordinate sampling points. In order to obtain the ME features, previous methods either directly encode the point sequences by RNN, or use CNN to extract convolutional features from the reconstructed ME images [5], [6], [7], [8], [9], [15]. However, both methods fail to resolve the contradiction between trajectory information and visual content. Specifically, the RNN-based algorithms directly process the sequential features but neglect the vision information that may contain rich spatial semantics; thus, the performance is easily affected by the writing trajectory disorder. On the contrary, the CNN-based models first reconstruct an ME image by connecting sampling points, then extract image features upon the reconstructed picture. Although these methods can preserve the spatial features of the two-dimensional structure, the trajectory information is abandoned; therefore, they fail catastrophically on visually similar symbols. Recently, some methods [3], [16] attempt to combine the CNN and RNN module to enable the exploration of both trajectory and image features, which boosts the recognition accuracy, albeit at the cost of computational overhead. To this end, we propose a new framework by extracting the path signature features [17], [18], [19] from writing trajectories to capture both local writing details and global spatial information. We observed that the proposed method improves symbol recognition performance and facilitates ME structure parsing without introducing additional model computation and storage. Notably, our method does not rely on writing trajectory order while extracting the path signature feature, which alleviates the issue that causing performance degradation in RNN-based methods. The detail of extracting path signature features is illustrated in Section III-A.

Language model rectification: Previous RNN-based language models typically factor computation along with the symbol positions of the sequence, which inherently precludes parallelization at the training stage; thus, it consumes considerable training time and memory, especially for the longer sequences. Yu et al. [20] try to solve this issue by utilizing the transformer [21] to predict the exact symbol category in parallel. However, it can be ambiguous to predict the exact symbol category since there is usually more than one symbol that conforms to the ME grammar. For example, in the LaTeX sequence, “ $a [mask] b = 36$,” the only clue we have is that the “[mask]” represents a binary operation; however, it is less possible to confirm the exact symbol, as there is more than one option that meets the condition, e.g., ‘ \times ’, ‘ $+$ ’. Besides, as shown in Fig. 1, we observed that visually similar symbols usually play different roles in the MEs, e.g., the left round bracket ‘(’ is used to isolate a segment of data from its surroundings, which is typically deployed with its symmetric counterpart ‘)’; while ‘ c ’ belongs to the English letter, which usually denotes a variable. Thanks to this finding, it becomes possible to train a language model to aid the vision-based

Ground Truth	1	0	0	,	0	0	0
Model A	1	0	,	0	0	0	
Model B	1	0	0	.	0	0	0
Model C	1	0	0	,	0	0	0
Ensemble	1	0	0	0	0	0	0

Fig. 2. Problem of existing time-step-based ensemble methods. Although model A correctly predicted the punctuation symbol ‘,’ the different sequence length caused its confidence cannot be accurately added to Model B and Model C based on the time step, resulting in error ensemble results.

recognition model. That is to say, the language model can rectify the misclassification errors due to visual similarity by analyzing the contextual information, which further improves the HMER precision.

Multi-model ensemble: Model ensemble is a widely-used technique for reducing the variance in deep neural networks. Previous works [6], [8], [22] eliminate the variance by combining predictions from multiple models based on the time step, thereby increasing the accuracy of HMER. However, the main weakness with the existing framework is that the misalignment of different predictions will lead to cumulative errors. As shown in Fig. 2, although model A correctly predicted the punctuation symbol ‘,’ the different sequence length caused its confidence cannot be accurately added to Model B and Model C based on the time step, resulting in error ensemble results. To tackle this issue, we propose to utilize the Dynamic Time Warping (DTW) algorithm [23] to align the sequences predicted from different models. After alignment, the candidate symbol confidence of corresponding positions in different length sequences can be added correctly, further enhancing the ensemble accuracy.

We summarize the main contributions of this paper as follows:

- We proposed a feature enhancement method by extracting path signature features, which preserves both global spatial information and local writing details.
- A neural language model is introduced to rectify misclassification errors between visually similar symbols by analyzing the contextual information.
- We solved the misalignment issue in existing HMER ensemble approaches by designing a DTW-based algorithm.
- Thanks to the above improvements, our method achieves state-of-the-art results on CROHME datasets, outperforming previous methods by a large margin.

II. RELATED WORKS

The section below describes related works of traditional and deep learning-based methods of HMER, the Language model used in sequence recognition, and ensemble methods.

A. Handwritten Mathematical Expression Recognition

1) *Traditional Methods:* Traditional methods for HMER have been surveyed by Zanibbi et al. [24] and Mouchere et al. [25]. These methods were usually based on grammar, such

as graph grammar [26], [27], definite clause grammar [28], relational grammar [29], and context-free grammar [30], [31]. According to the solution of HMER, traditional methods can be categorized into sequential methods and global methods. Sequential methods treat HMER as a two-stage fashion, which separately recognize symbols and parse structures. For example, Alvaro et al. [30] applied Hidden Markov Model to recognize symbols while parsing the ME structure via two-dimension stochastic context-free grammar. Such approaches, however, have failed to utilize the contextual information of the whole ME and may cause an accumulation of errors. On the contrary, global methods process the two sub-tasks simultaneously. A later work of Alvaro et al. [32] combined several stochastic sources of information in different perception levels and globally determine the most likely ME. However, the computation cost of this type of method increased exponentially with the number of symbols.

2) *Deep Learning-Based Methods*: Recently, deep learning technology has rapidly developed and been applied in HMER task successfully. There are mainly two types of modeling methods of deep learning-based methods: sequence modeling and graph modeling. Sequence modeling methods treat HMER as a sequence predicting problem, outputting LaTeX sequence for representing ME. The method proposed by Deng et al. [33] is a pioneer work that first introduced the encoder-decoder framework into the offline HMER task. Subsequently, Zhang et al. [9] improved the encoder-decoder framework, achieving excellent results on CROHME datasets [11], [12]. Multi-scale attention was further proposed in [34] for tackling the issue of variant symbol size. In order to alleviate the affect of various handwriting styles in ME, Wu et al. [22], [35] utilized the paired adversarial learning to map the handwriting ME features into print ME features, reducing the difficulty of recognition and improving the performance. Considering the issue of lack of data in the HMER task, Le et al. [15] proposed a series of pattern generation strategies to augment data diversity. For increasing corpus diversity in training sets, Le et al. [5] also utilized a dual loss to train model on print ME data, which is generated by existing LaTeX corpus [5]. Li et al. proposed a scale augmentation method to handle the variant ME size and a drop-attention regularization to improve robustness [6]. Truong et al. proposed a symbol classifier to improve the localization and classification of the high-level features [7]. For online HMER, Zhang et al. [8], [36] proposed a framework equipping GRU or LSTM encoder to extract features from writing trajectories. Wang et al. [3] proposed a new attention strategy, constraining the allocation unit of attention weights on strokes. These works were mainly trained on CROHME datasets, whose data form is a sequence of two-dimensional coordinate sampling points. Offline HMER methods used CNN for feature extraction on images reconstructed by connecting sampling points, which lost the writing trajectory order and direction information. Online HMER methods used RNN for feature extraction on writing trajectory sequence, which neglects the image features and unable to model the two-dimensional spatial structure directly. In addition, Wang et al. [3], [16] proposed multi-modal methods via

combining CNN and RNN features but somewhat fall short in the model complexity.

Graph modeling methods treat HMER as a graph predicting problem, outputting Symbol Layout Tree (SLT) for representing ME. Zhang et al. [37] utilized a tree-based BLSTM, modeling the symbols as nodes and the relationship between symbols as edges from writing trajectories directly. Mahdavi et al. [38] utilized a Line-Of-Sight (LOS) graph for HMER, recognizing symbols by CNN and parsing the SLT by Edmonds' algorithm. Furthermore, Mahdavi et al. [39] utilized the query-driven global graph attention on LOS to parsing SLT. Meanwhile, Zhang et al. [2], [10] proposed a sequential relation decoder to decode expressions into tree structures. Wu et al. [4] took HMER as a graph-to-graph learning problem, utilizing the GNN to explore the hierarchical structure and segment mathematical symbols explicitly. These methods are more interpretative and thus having great potentials for further deployment.

B. Language Model for Sequence Recognition

Regarding improving the HMER performance of sequence modeling methods, it is common to use language models to aid the vision-based recognition model in correcting misclassified symbols. Wu et al. [22] used a 4-gram statistical language model approximated by statistical data from the corpus. By given contexts, Zhang et al. [8] used a GRU-based language model to predict the next symbol in textual data. However, such approaches have been mostly restricted due to the time-consuming pipeline since they are unparalleled when predicting the next symbol. To take advantage of the parallelism and modeling capabilities of transformer, transformer-based language model [21] has been used to rectify the recognition results in the scene text recognition task. For example, Ung et al. [40] proposed a transformer-based math language model, using unidirectional semantics for improving HMER performance. Yu et al. [20] used it to capture global semantic context through multi-way parallel transmission and improved the recognition performance. Meanwhile, Fang et al. [41] proposed an autonomous and bidirectional transformer-based language model to rectify the recognition results iteratively. It is important to note that all of the language models mentioned above directly predict exact symbol categories, e.g., 'c' for 'c'. However, since more than one symbol can conform to the grammar of MEs, predicting a particular category becomes ambiguous. To this end, our language model instead to predict the group to which the symbol belongs, e.g., 'c' for "lowercase letter".

C. Ensemble of Sequence Recognition Model

The model ensemble is a widely used technique to improve the performance and robustness of sequence recognition models. One of the most straightforward ways is to directly accumulate the candidate symbol confidence from multiple models according to the time step, then use the maximum index to gain the final prediction [6], [9], [22]. Based on this method, Zhang et al. [8] and Truong et al. [7] further employ beam search process to select the best candidate. In the text recognition task,

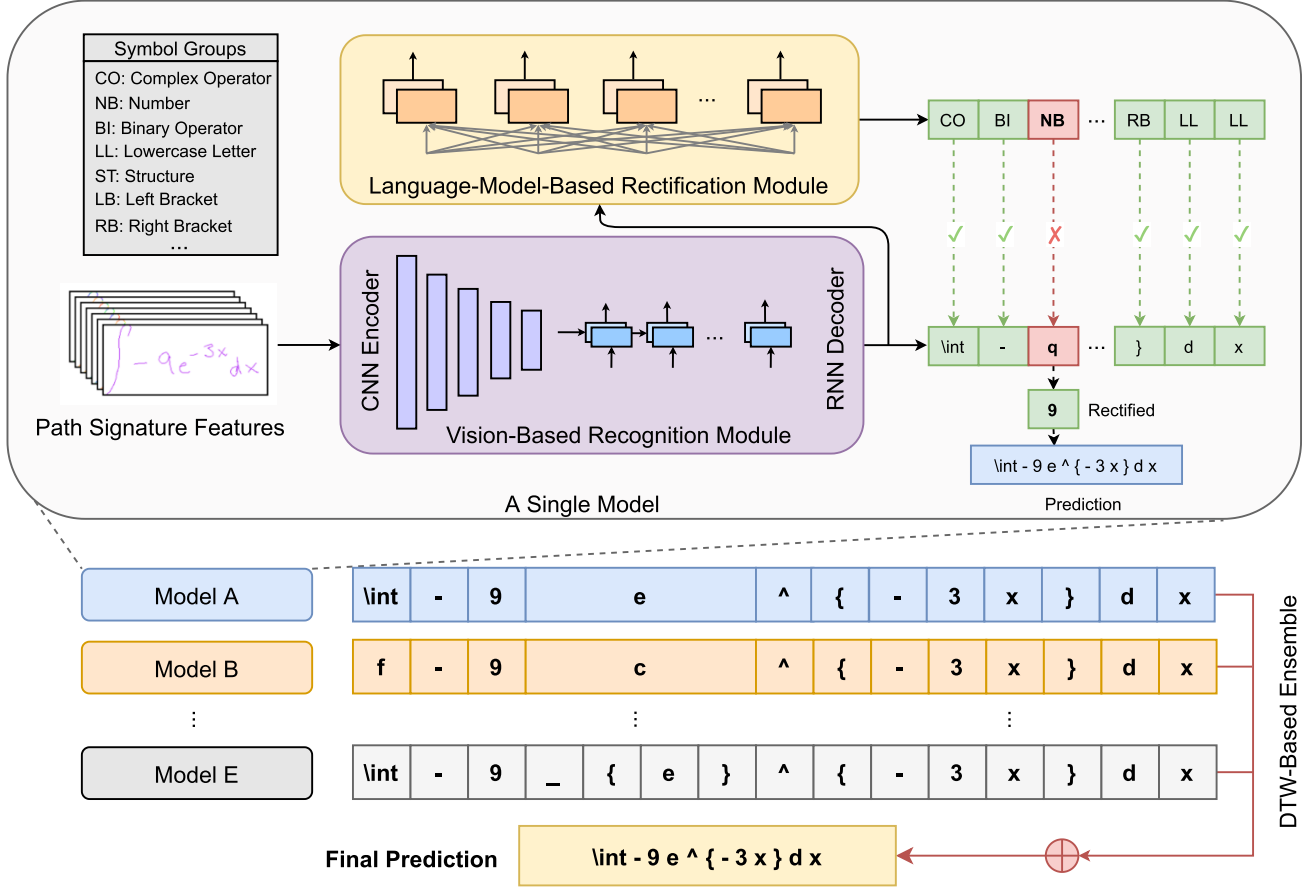


Fig. 3. The entire pipeline of the proposed framework. The path signature features are first extracted from writing trajectories and then fed into an encoder-decoder based visual recognition model. Furthermore, the sequence decoded by the visual recognition model is rectified by a language model. Finally, results from multiple models are ensemble by a DTW-based algorithm for final predictions.

Shi et al. [42] utilized a bidirectional decoder, which consists of two decoders with opposite directions. Specifically, the decoder predicts the sequence from left-to-right and right-to-left order and then merges them by picking the one with the highest confidence. All these methods can achieve higher performance than a single model but still suffer from the alignment problem caused by different lengths of predicting sequence from multiple models.

III. METHOD

The entire pipeline of the proposed framework is shown in Fig. 3. First, we extract the path signature features from writing trajectories and incorporate the features into the visual recognition model. The visual recognition model contains a CNN-based encoder that performs further feature extraction, while the other part is a decoder based on attention mechanisms, which predicts the LaTeX sequence step by step. Subsequently, the language model learns to predict the group to which the symbol belongs by reading contextual information, correcting the misclassification error caused by the visual model. Finally, a DTW-based algorithm is designed to ensemble the results from multiple models for generating the final predictions, which significantly improves the robustness and performance.

A. Path Signature Feature

The path signature is pioneered by [17] in the form of iterated integrals and developed as a fundamental component of rough path theory by [18] and [19]. Mathematically, for positive integers k and time intervals $[s, t]$, the k -th order path signature feature of path S is defined as follows:

$$P(S)_{s,t}^k = \int_{s < u_1 < u_2 < \dots < u_k < t} 1 dS_{u_1} \otimes dS_{u_2} \otimes \dots \otimes dS_{u_k}, \quad (1)$$

where \otimes denotes tensor product. In this work, the trajectory of ME written in two-dimensional plane is regarded as the path S , and the path signature is a 2^k dimensional vector. For $k = 0$, the iterated integral is defined as 1. For $k = 1$, the path signature is a 2^1 dimensional vector as follows:

$$(P_x(S)_{s,t}^1, P_y(S)_{s,t}^1), \quad (2)$$

$$P_x(S)_{s,t}^1 = \int_{s < u_1 < t} 1 dS(x)_{u_1} = S(x)_t - S(x)_s, \quad (3)$$

$$P_y(S)_{s,t}^1 = \int_{s < u_1 < t} 1 dS(y)_{u_1} = S(y)_t - S(y)_s, \quad (4)$$

where $S(x)$ and $S(y)$ are the projection of X-axis and Y-axis of path S in the two-dimensional plane (see Fig. 4), respectively.

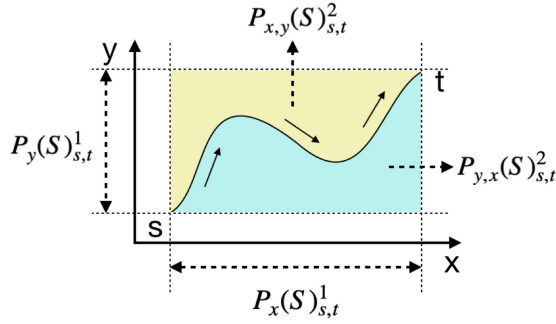


Fig. 4. Physical meaning of path signature. $P_x(S)^1_{s,t}$ and $P_y(S)^1_{s,t}$ represent the projections of X-axis and Y-axis of the displacement between S_s and S_t . $P_{x,y}(S)^2_{s,t}$ and $P_{y,x}(S)^2_{s,t}$ represent the areas in yellow and blue.

Therefore, in the case of $k = 1$, the path signature represents the displacement between S_s and S_t . For $k = 2$, the path signature is a 2^2 dimensional vector as follows:

$$(P_{x,x}(S)^2_{s,t}, P_{x,y}(S)^2_{s,t}, P_{y,x}(S)^2_{s,t}, P_{y,y}(S)^2_{s,t}), \quad (5)$$

$$\begin{aligned} P_{x,x}(S)^2_{s,t} &= \int_{s < u_2 < t} \int_{s < u_1 < u_2} 1 dS(x)_{u_1} dS(x)_{u_2} \\ &= \frac{1}{2} (S(x)_t - S(x)_s)^2 \\ &= \frac{1}{2} (P_x(S)^1_{s,t})^2, \end{aligned} \quad (6)$$

$$P_{x,y}(S)^2_{s,t} = \int_{s < u_2 < t} \int_{s < u_1 < u_2} 1 dS(x)_{u_1} dS(y)_{u_2}, \quad (7)$$

$$P_{y,x}(S)^2_{s,t} = \int_{s < u_2 < t} \int_{s < u_1 < u_2} 1 dS(y)_{u_1} dS(x)_{u_2}, \quad (8)$$

$$\begin{aligned} P_{y,y}(S)^2_{s,t} &= \int_{s < u_2 < t} \int_{s < u_1 < u_2} 1 dS(y)_{u_1} dS(y)_{u_2} \\ &= \frac{1}{2} (S(y)_t - S(y)_s)^2 \\ &= \frac{1}{2} (P_y(S)^1_{s,t})^2. \end{aligned} \quad (9)$$

The values of (6) and (9) are exponentially related to displacement between S_s and S_t . Moreover, the (7) and the (8) represent the areas in yellow and blue in Fig. 4, respectively. Therefore, in the cases of $k = 2$, the path signature is related to the curvature of the path S . The other k -th order path signatures also have other geometric meanings.

In practice, the path signature is applied in various ways to some tasks, including online character recognition [43], [44], Chinese text recognition [45], signature verification [43], sound compression [46], hand movement classification [47], etc. In this work, it is implemented as a set of feature maps, containing both writing trajectory details and two-dimensional spatial information, which is important for symbol recognition and structure parsing in HMER task. The form of online HMER data is a sequence of two-dimensional coordinate sampling points. Given a sequence data as follows:

$$[(x_1, y_1), (x_2, y_2), \dots, (0, -1), \dots, (x_L, y_L), (0, -1)], \quad (10)$$

Algorithm 1: Inserting points in a stroke

Input:

Stroke = $[(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)]$;

Output:

- 1: Define accumulation distance $D_1 = 0$;
 - 2: **for** each $n \in [2, N]$ **do**
 - 3: $D_n = D_{n-1} + \sqrt{(x_n - x_{n-1})^2 + (y_n - y_{n-1})^2}$
 - 4: **end for**
 - 5: Declare $(x'_1, y'_1) = (x_1, y_1)$;
 - 6: Initialize $j = 1$
 - 7: **for** each $i \in [2, D_N]$ **do**
 - 8: **while** $D_{j+1} < i$ **do**
 - 9: $j = j + 1$
 - 10: **end while**
 - 11: $\alpha = (D_{j+1} - i) / (D_{j+1} - D_j)$;
 - 12: $x'_i = \alpha * x_j + (1 - \alpha) * x_{j+1}$;
 - 13: $y'_i = \alpha * y_j + (1 - \alpha) * y_{j+1}$;
 - 14: **end for**
 - 15: **return**
- Normalized stroke = $[(x'_1, y'_1), (x'_2, y'_2), \dots, (x'_{D_N}, y'_{D_N})]$.

(x_1, y_1) represents the first sampling point of the first stroke from two-dimensional plane of input device and L represents the total number of sampling points of this sequence data. Meanwhile, $(0, -1)$ means the end of one stroke, generated by pen-up detection of input device. We restructure the sequence data as follows:

$$[Stroke_1, Stroke_2, \dots, Stroke_M], \quad (11)$$

$$Stroke_m = [(x_1, y_1), (x_2, y_2), \dots, (x_{N_m}, y_{N_m})], \quad (12)$$

where $m = 1, 2, \dots, M$, M is the number of stroke and N_m is the number of sampling points of the m -th stroke. In order to tackle the issue of writing trajectory disordered, we extract path signature features based on individual stroke. Firstly, we insert points between sampling points for normalization, detailing in Algorithm 1. This normalization operation can avoid the influence of data distribution caused by handwriting speed or sampling rate.

Through the way of sliding window, we cropped a sub-stroke S' from normalized stroke and calculated the path signature features recursively as follows:

$$P(S')^k_{i_1, i_{N'}} = \sum_{r=0}^k P(S')^r_{i_1, i_{N'-1}} \otimes P(S')^{k-r}_{i_{N'-1}, i_{N'}}, \quad (13)$$

$$P(S')^k_{i_1, i_2} = \begin{cases} 1 & k = 0 \\ (P(S')^{k-1}_{i_1, i_2} \otimes \Delta_{i_1, i_2}) / k & k > 0 \end{cases}, \quad (14)$$

where N' is the number of points of sub-stroke S' , $N' = 2 * window\ size + 1$, i_1 is the index of the first point of sub-stroke S' , and Δ_{i_1, i_2} is the displacement between the first point and the second point. We define the collection of $\leq k$ order path signature as follows:

$$(P(S')^0_{i_1, i_{N'}}, P(S')^1_{i_1, i_{N'}}, \dots, P(S')^k_{i_1, i_{N'}})$$

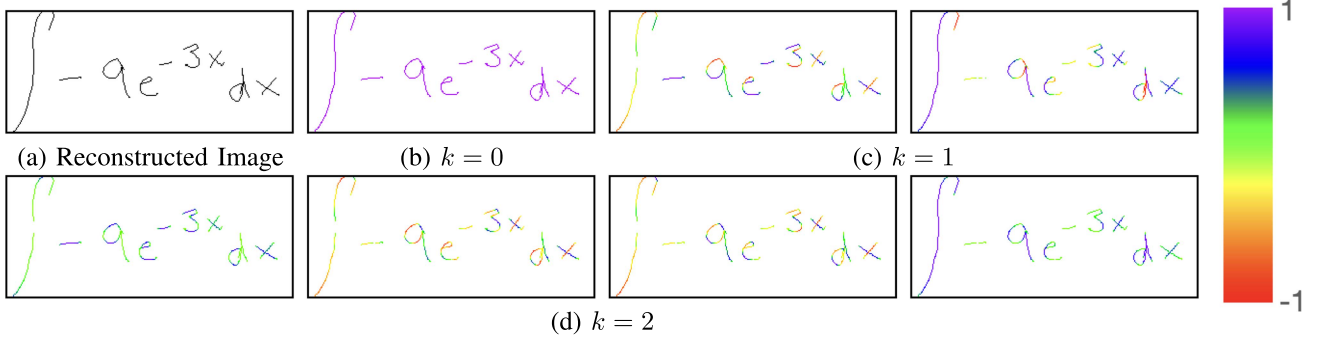


Fig. 5. Path signature feature map of ME. (a) Reconstructed image by connecting sampling points. (b) Feature map when $k = 0$. (c) Feature maps when $k = 1$. (d) Feature maps when $k = 2$. The value is mapped into rainbow color, the max value is represented as purple while the min value is represented as red.

$$= \left(1, P_x(S')_{i_1, i_{N'}}, P_y(S')_{i_1, i_{N'}}, \dots, P(S')_{i_1, i_{N'}}^k\right), \quad (15)$$

where the dimension is $C = 2^{k+1} - 1$. The path signature values are normalized into $[-1, 1]$ and then assigned to the feature maps $\mathbf{I} \in \mathbb{R}^{H \times W \times C}$ according to the coordinate of the $i_{\lfloor N'/2 \rfloor}$ -th point in S' , where H and W are the height and width of the feature map, respectively. Furthermore, we slide the window with stride 1 for calculating feature values of the next sub-stroke, as shown in Fig. 5. By applying the path signatures, the feature map \mathbf{I} preserves both local details *i.e.*, the writing trajectory information (writing direction, displacement, and curvature of strokes), and global spatial information (shape and position of symbols), which are then fed into CNN encoders for further symbol recognition and structure parsing.

B. Vision-Based Recognition Model

Thanks to the superiority in handling variable-length input and output sequences, the encoder-decoder framework has been widely used in many sequence predicting tasks, including text recognition, speech recognition, machine translation, and etc [48], [49], [50]. Such framework uses the encoder to extract features from input data; then predicts the sequences via the decoder based on attention mechanism. In this work, the encoder is built with a 33-layer ResNet [51] like backbone network while the decoder is constructed on RNN. In order to obtain the recognition results, the path signature feature maps \mathbf{I} of an ME are first fed into the encoder; subsequently, the output features $\mathbf{F} \in \mathbb{R}^{H' \times W' \times C'}$ are fed to decoder to predict sequence \hat{Y} iteratively. At time step t , the probability p_t^{vis} of predicting symbol depends on context \mathbf{c}_t and current hidden state \mathbf{h}_t , as shown in (16):

$$p_t^{vis} = g([\mathbf{c}_t; \mathbf{h}_t]), \quad (16)$$

where $g(\cdot)$ is a linear function and $[\mathbf{c}_t; \mathbf{h}_t]$ means the concatenation of \mathbf{c}_t and \mathbf{h}_t . Context \mathbf{c}_t is computed as a weighted sum of features \mathbf{F} and position embeddings \mathbf{Q} , as illustrated by (17):

$$\mathbf{c}_t = \sum_{i,j} \alpha_t^{i,j} \mathbf{F}_{i,j};$$

$$i = 1, 2, \dots, H';$$

$$j = 1, 2, \dots, W', \quad (17)$$

where $\alpha_t^{i,j}$ is the attention weight of features $\mathbf{F}_{i,j}$ at time step t , H' and W' are the height and width of \mathbf{F} , respectively. The attention weights are computed by linear functions with activation function \tanh as follows:

$$e_t^{i,j} = \mathbf{W}_1 \tanh \left(\mathbf{W}_2 \left[\mathbf{W}_3 \mathbf{h}_t; \mathbf{F}_{i,j}; \mathbf{Q}_{i,j}; \mathbf{W}_4 s_t^{i,j} \right] \right) \quad (18)$$

$$\alpha_t^{i,j} = \frac{\exp(e_t^{i,j})}{\sum_{i,j} \exp(e_t^{i,j})}, \quad (19)$$

where \mathbf{W}_1 , \mathbf{W}_2 , \mathbf{W}_3 , and \mathbf{W}_4 are trainable parameters. To enhance the awareness of two-dimensional spatial position, absolute position embeddings \mathbf{Q} are considered when computing the attention weights. Position embeddings \mathbf{Q} are defined as the concatenation of the embeddings in vertical and horizontal direction, annotated as $\mathbf{E}_v \in \mathbb{R}^{H' \times C'}$ and $\mathbf{E}_h \in \mathbb{R}^{W' \times C'}$:

$$\mathbf{Q}_{i,j} = [\mathbf{E}_v(i); \mathbf{E}_h(j)]. \quad (20)$$

Moreover, coverage features s are also utilized to prevent from over-attention or under-attention [9], [52], calculated as follows:

$$s_t^{i,j} = \mu s_{t-1}^{i,j} + \alpha_{t-1}^{i,j}, \quad (21)$$

where μ is a hyper-parameter.

The hidden state is computed by RNN with long-short term memory cells. At time step t , the RNN input are the word embeddings \mathbf{E}_w of last time step predicting word \hat{Y}_{t-1} concatenated with last time step context \mathbf{c}_{t-1} , as follows:

$$\mathbf{h}_t = \text{RNN}([\mathbf{E}_w(\hat{y}_{t-1}); \mathbf{c}_{t-1}], \mathbf{h}_{t-1}), \quad (22)$$

$$\hat{y}_{t-1} = \arg \max_K p_{t-1}^{vis}, \quad (23)$$

$$\mathbf{c}_{t-1} = \sum_{i,j} \alpha_{t-1}^{i,j} (\mathbf{F}_{i,j} + \mathbf{Q}_{i,j}); \quad (24)$$

where K is the size of the symbol set. This decoder with attention mechanism predicts sequence iteratively until predicted the special mark “end-of-sequence”. By truncating the words of total time step T , we get the predicting sequence \hat{Y} :

$$\hat{Y} = [\hat{y}_1, \hat{y}_2, \dots, \hat{y}_{t-1}, \hat{y}_t, \hat{y}_{t+1}, \dots, \hat{y}_T]. \quad (25)$$

TABLE I
11 GROUPS OF ME SYMBOLS

Groups	Symbols
binary operator	$+ - \times \div / \pm \leq \geq < > \neq =$
structure	$\backslash \text{sqrt} \backslash \text{frac} \wedge _$
left brackets	$([\{$
right brackets	$)] \}$
punctuation	$, ! \dots$
number	$1 2 3 4 5 6 7 8 9 0 . \infty \pi$
greek alphabet	$\sigma \gamma \theta \Delta \alpha \lambda \phi \beta \mu$
complex operator	$\int \sum \log \lim \sin \cos \tan$
uppercase letter	A B C E F G H I L M N P R S T V X Y
lowercase letter	a b c d e f g h i j k l m n o p q r s t u v w x y z
others	$ \forall \exists ! \in \rightarrow$

The loss function calculated between predicted sequence \hat{Y} and ground-truth sequence Y for training the recognition model is illustrated as follows:

$$\begin{aligned} Loss_{vis} &= CrossEntropyLoss(\hat{Y}, Y) \\ &= - \sum_{t=1}^T \log \left(\frac{\exp(p_t^{vis}(y_t))}{\sum_{k=1}^K \exp(p_t^{vis}(k))} \right). \end{aligned} \quad (26)$$

During training, the recognition model is optimized to recognize symbols and parse structure by predicting LaTeX sequence.

C. Language Model-Based Rectification Module

As shown in Fig. 1, visually similar symbols are prone to be recognized incorrectly. A common way to solve this issue is to use language models to rectify misclassified symbols by understanding contextual and transition information [8], [22]. Unfortunately, due to the nature of sequential features, most of the previous language models inherently preclude parallelization within training samples; thereby these methods are quite time-consuming. Although transformer-based language models [20], [21] solved the parallel training issue, they still encounter ambiguity problems for HMER. For example, multiple operations conform to the expression “a [mask] b = 36,” i.e., “[mask]” can represent any of the binary operators. To this end, we did not use the language model to predict a specific category for the symbols directly but divided them into 11 groups according to their attributes (see Table I). Then, the visual recognition results are rectified by predicting the group to which the symbol belongs (referred to as “symbol group” in the following).

Firstly, we input the predicted sequence \hat{Y} from the recognition model into the masked language model proposed in BERT [53]. In masked language model, we replace the \hat{y}_t with symbol [mask] and input this masked sequence \hat{Y}'_t into the neural network language model LM . The language model LM is constructed based on the transformer encoder, which consists of a stack of identical layers. Each layer has two sub-layers, a multi-head self-attention network and a fully connected feed-forward network [21]. The language model LM predicts symbol group \hat{r}_t of ground-truth y_t instead of specific category, as follows:

$$\hat{r}_t = \arg \max_{K^*} p_t^{lang}, \quad (27)$$

$$p_t^{lang} = f(o_t), \quad (28)$$

$$o_t = TransformerEncoder(E_w^*(\hat{Y}'_t)), \quad (29)$$

$$\hat{Y}'_t = [\hat{y}_1, \hat{y}_2, \dots, \hat{y}_{t-1}, [mask], \hat{y}_{t+1}, \dots, \hat{y}_T]. \quad (30)$$

where K^* is the size of symbol group set, $f(\cdot)$ is a linear function, E_w^* is the word embedding, and o_t is the t -th output of transformer encoder, corresponding to the symbol [mask]. The symbol group \hat{r}_t of different time step t can be calculated in parallel and forms predicting sequence \hat{R} :

$$\hat{R} = [\hat{r}_1, \hat{r}_2, \dots, \hat{r}_T]. \quad (31)$$

The loss function calculated between predicted sequence \hat{R} and ground-truth sequence R for training the language model is illustrated as follows:

$$\begin{aligned} Loss_{lang} &= CrossEntropyLoss(\hat{R}, R) \\ &= - \sum_{t=1}^T \log \left(\frac{\exp(p_t^{lang}(r_t))}{\sum_{k=1}^{K^*} \exp(p_t^{lang}(k))} \right). \end{aligned} \quad (32)$$

To achieve high efficiency, the language model and the vision-based recognition model are trained at the same time.

Subsequently, we assign the probability value of symbol group $p_t^{lang}(\hat{r}_t)$ to the corresponding category illustrated in Table I. The probability of exact category from the language model is denoted as \tilde{p}_t^{lang} . The predicting sequence after rectification is shown as follows:

$$\hat{Z} = [\hat{z}_1, \hat{z}_2, \dots, \hat{z}_T], \quad (33)$$

$$\hat{z}_t = \arg \max_K p_t^{total}, \quad (34)$$

$$p_t^{total} = \Phi(p_t^{vis}) + \lambda \Phi(\tilde{p}_t^{lang}), \quad (35)$$

where λ is the hyper-parameter and Φ is the soft-max function, outputting the confidence of predicting symbol.

D. DTW-Based Model Ensemble

The errors in sequence predicting task can be summarized into 3 cases: substitution error, deletion error, and insertion error. The language model can only rectify the substitution error occurred in vision-based recognition model. The most straightforward way to ensemble models is that training multiple identical networks with different initialization, then merging the results corrected by language models according to the time-step [6], [8], [9], [22]. Nevertheless, as shown in Fig. 2, such methods cannot handle the case of different sequence length caused by deletion error or insertion error.

To tackle this issue, we propose a new ensemble strategy based on the DTW algorithm. Concisely, the DTW algorithm measures similarity between multiple temporal sequences that vary in length; then, it performs an optimal match between the given sequences, further outputting explicit alignment paths. In detail, N networks consist of identical structures, including vision-based recognition and language-model-based rectification modules are trained with different initialization. Thus, for

each input sample, N sequence results $\Omega_1, \Omega_2, \dots, \Omega_N$ can be obtained. Supposing the time step length of each output Ω_i is T_i ; then, we count the occurrence of T_i and denote the one with the highest frequency as \tilde{T} . If \tilde{T} is not unique, e.g., all output sequence has a different length, then the prediction from N models that has the highest confidence would be selected as the reference. Otherwise, only the prediction with the same length as \tilde{T} and the highest average confidence will be selected as reference. Once the reference sequence is determined, the alignment paths $\mathbf{P} \in \mathbb{R}^{\tilde{T} \times 2}$ between it and other predictions will be calculated by the DTW algorithm. Specifically, the alignment path \mathbf{P} output from DTW is illustrated as follows:

$$[(s_1, t_1), (s_2, t_2), \dots, (s_{\tilde{T}}, t_{\tilde{T}})], \quad (36)$$

where (s, t) means the t -th output of query sequence is aligned to the s -th output of reference sequence. Based on the alignment paths \mathbf{P} , the confidence of the outputs obtained by different models can be accurately accumulated and thus contribute to the final prediction. Since the models have the same network structure but are initialized differently, we set the same weights for each model when accumulating confidence. The detail of our DTW-based model ensemble strategy is illustrated in Algorithm 2.

IV. EXPERIMENTS

In this section, we give the implementation details of our method. To further explore the effectiveness of the proposed methods, experiments on the widely-used benchmark CROHME are conducted, which show that our model outperforms previous state-of-the-art methods by a large margin.

A. Implementation Details

Our method was implemented in PyTorch and optimized on 6 Nvidia GTX 1080Ti GPUs distributionally. The batch size was set to 6 on every GPU for parallel computing. Since the ME samples in the training set were collected from different devices, we used different scale factors to normalize each ME sample into a 256×768 matrix and applied random scale augmentation in the range of $[0.5, 1.5]$ while training. When calculating path signature features, the window size and order were practically set as 4 and 2, respectively. After that, the path signature features were assigned on a 256×768 canvas. For the feature extraction enhancement module, we used a customized ResNet-34 [51] as the CNN backbone network for the encoder part. Empirically, the kernel size and stride of *conv1* layer are modified to 3×3 and 1×1 , respectively, along with a *max-pooling* layer that has 2×3 stride. Turning to the decoder, it was built with an RNN that consists of 2-layer long-short term memory blocks, while the dimension of h_t was set to 256. Meanwhile, the dimensions of position embeddings E_v and E_h were practically set to 32. The dimension of word embedding E_w was set to 64. In addition, the language-model-based rectification module was developed upon the Transformer [21]. Specifically, a 6-layer transformer encoder with eight heads was adopted, the dimensions of position encoder, word embedding, self-attention network, and feed-forward network were all set to 512. Notably, this module was

Algorithm 2: DTW-based Model Ensemble

Input:

The sequences of predicting symbol probability from N different models $\Omega_1, \Omega_2, \dots, \Omega_N$;

$\Omega_n = [p_1^{total}, p_2^{total}, \dots, p_{T_n}^{total}]$; $\Omega_n \in \mathbb{R}^{T_n \times K}$; The total time steps (lengths) of N predicting sequences T_1, T_2, \dots, T_N ;

Average confidence for each time step of n predicting sequences

C_1, C_2, \dots, C_N ; $C_n = \sum_{t=1}^{T_n} (\max_K p_t^{total}) / T_n$;

Output:

- 1: Count the most frequent time step \tilde{T} according to T_1, T_2, \dots, T_N ;
 - 2: Calculate the number $num(\tilde{T})$ of N predicting sequences whose total time step is equal to \tilde{T} ;
 - 3: Declare the reference sequence $\tilde{\Omega}$;
 - 4: Initialize the max average confidence $C_{max} = 0$;
 - 5: **for** each $n \in [1, N]$ **do**
 - 6: **if** $num(\tilde{T}) > 1$ **then**
 - 7: **if** $T_n == \tilde{T}$ and $C_n > C_{max}$ **then**
 - 8: $C_{max} \leftarrow C_n$,
 - 9: $\tilde{\Omega} \leftarrow \Omega_n$;
 - 10: **end if**
 - 11: **else**
 - 12: **if** $C_n > C_{max}$ **then**
 - 13: $C_{max} \leftarrow C_n$,
 - 14: $\tilde{\Omega} \leftarrow \Omega_n$;
 - 15: **end if**
 - 16: **end if**
 - 17: **end for**
 - 18: Initialize the final ensemble sequence $\bar{\Omega} = \tilde{\Omega}$;
 - 19: Initialize the temp sequence $\Omega' = \tilde{\Omega}$;
 - 20: Denote the p_t^{total} of Ω_n as $p_t^{\Omega_n}$, omitting the sup-script *total*;
 - 21: **for** each $n \in [1, N]$ **do**
 - 22: **if** Ω_n is not $\tilde{\Omega}$ **then**
 - 23: Calculate the alignment path \mathbf{P} between as:
 $\mathbf{P} = DTW(\Omega', \Omega_n)$;
 - 24: **for** each $s, t \in \mathbf{P}$ **do**
 - 25: $p_s^{\bar{\Omega}} = p_s^{\tilde{\Omega}} + p_t^{\Omega_n}$,
 - 26: $p_s^{\Omega'} = (p_s^{\tilde{\Omega}} + p_t^{\Omega_n}) / 2$;
 - 27: **end for**
 - 28: **end if**
 - 29: **end for**
 - 30: **for** each $t \in [1, \tilde{T}]$ **do**
 - 31: $\hat{\xi}_t = \arg \max_K p_t^{\bar{\Omega}}$;
 - 32: **end for**
 - 33: **return** $\hat{\Xi} = [\hat{\xi}_1, \hat{\xi}_2, \dots, \hat{\xi}_{\tilde{T}}]$.
-

jointly trained with the vision-based recognition module without using any additional corpus. Moreover, the hyper-parameter μ in (21) was set to 0.9, while λ in (35) was set to 0.5.

The whole model was trained on CROHME training set (8834 ME samples) for 80,000 iterations with SGD optimizer.

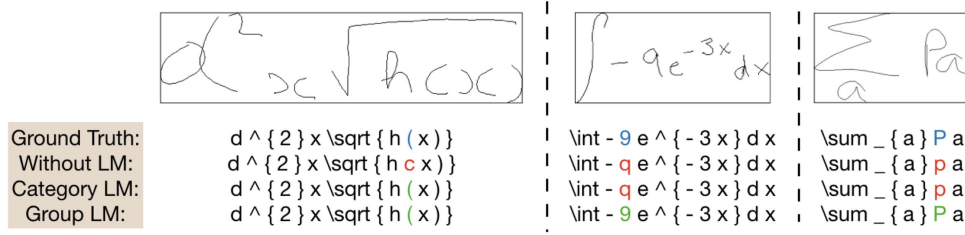


Fig. 6. Examples of language model rectification. The proposed “Group LM” is able to correct the visually similar symbols by reading the contextual information. Especially in some cases, it can rectify the errors that cannot be noticed by the regular “Category LM” since it would be ambiguous to predict particular categories for each symbol in the HMER task.

TABLE II
COMPARISON OF EXPRate(%) BETWEEN DIFFERENT k -ORDERS

k	Channel	CROHME2014	CROHME2016	CROHME2019
0	1	55.78	55.88	59.88
1	3	57.71	58.24	62.14
2	7	58.92	59.46	63.22
3	15	59.63	59.02	62.80

The bold entities represent the highest experiment results.

The learning rate was initialized to 0.1 and decreased by 10 times every 20,000 iterations. To evaluate the effectiveness of the proposed framework, we report the experimental results based on the metric of expression recognition rate (ExpRate) on CROHME 2014 [11] test set (986 ME samples), CROHME 2016 [12] test set (1147 ME samples), and CROHME 2019 [13] test set (1199 ME samples).

B. Ablation Studies

1) *Path Signature Features*: As shown in Table II, we explored the impact of k -order (see (15)) on the discrimination of path signature features.

In the case of $k = 0$, the input is actually a reconstructed ME image without feature enhancement, which sets a baseline performance 55.78%, 55.88%, and 59.80% ExpRate on CROHME2014, CROHME2016, and CROHME 2019 test splits, respectively. When using the enhanced path signature features, our methods achieved a significant performance improvement under all three different settings. For example, when $k = 1$, the ExpRate increased by 1.93%, 2.36%, and 2.26% on three test sets, respectively. The improvement illustrated that the path signature features are helpful for symbol recognition since the features contain the detail of writing trajectory, such as displacement, curvature. It should be noted that with the increase of k , the model cannot always gain performance improvements. Specifically, when $k = 3$, the accuracies of the model on CROHME2016 and CROHME 2019 are even lower than the one trained with $k = 2$. This phenomenon may suggest that redundant information was introduced since the feature channel explosively increased with k , resulting in both performance degradation and computation overhead. Therefore, in order to achieve a reasonable trade-off between performance and complexity, the order k is set to 2 in the following experiments.

2) *Language-Based Rectification*: Table III compares the performance after rectification by different language models.

TABLE III
COMPARISON OF EXPRate(%) BETWEEN DIFFERENT LANGUAGE MODELS

Language model	CROHME2014	CROHME2016	CROHME2019
Without LM	58.92	59.46	63.22
Category LM	59.94	59.63	63.80
Group LM	60.34	59.98	64.22

The bold entities represent the highest experiment results.

TABLE IV
COMPARISON OF EXPRate(%) BETWEEN DIFFERENT ENSEMBLE METHODS

Ensemble methods	CROHME2014	CROHME2016	CROHME2019
Single model	60.34	59.98	64.22
Time-step-based ensemble	62.89	63.12	65.97
DTW-based ensemble	64.81	63.82	66.89

The bold entities represent the highest experiment results.

“Without LM” represents the vision-only-based baseline result with path signature order $k = 2$. “Category LM” represents the regular language model trained to predict the exact category of each symbol, while “Group LM” is the proposed language model that was trained to predict the symbol group.

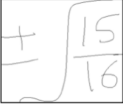
With the aid of the language model, both “Category LM” and “Group LM” enhanced the baseline accuracy on all of the three CROHME testing splits. However, it is noteworthy to mention that our proposed “Group LM” further boosts the performance on the basis of the “Category LM,” which achieved 1.42%, 0.52%, and 1.00% improvements compared to the baseline. The reason why the proposed “Group LM” performs better than the regular one is that predicting symbol groups rather than specific categories can avoid ambiguity problems that may lead to performance degradation. We show some qualitative results in Fig. 6, as illustrated in the figure, the LM is able to rectify the classification error of visually similar symbols. Especially in some cases, “Group LM” can correct the errors that cannot be noticed by “Category LM” since it is easier to get access to the symbol groups but not the certain categories.

3) *Model Ensemble*: Table IV shows the comparison of results between single model and different ensemble methods.

The first row shows the results obtained by a single model fulfilled with path signature features and “Group LM” rectification, which sets the baseline accuracy on the benchmark. The rest are the results achieved by ensembling 5 identical models with different initialization, using various ensemble methods. “Time-step-based ensemble” is the regular method that directly

TABLE V
ABLATION STUDY OF THE PROPOSED MODULES, *i.e.*, PATH SIGNATURE FEATURE (PSF), LANGUAGE-MODEL-BASED RECTIFICATION (LMR), AND DTW-BASED ENSEMBLE (DE), ON EIGHT PAIRS OF AMBIGUOUS SYMBOLS ON THREE CROHME TEST SETS

PSF	LMR	DE	' α '-' a '	' b '-' 6 '	' z '-' 2 '	' 9 '-' g '-' q '	' \times '-' x '-' X '	'('-' c '-' C '	' \cdot '-' \cdot '-' \cdot '	' 1 '-' l '-' l '
-	-	-	26	20	46	86	97	84	52	34
✓	-	-	15	13	46	82	98	82	44	33
✓	✓	-	13	9	44	78	96	80	43	31
✓	✓	✓	11	9	36	67	88	77	38	30
Error Drop (%)			57.69	55.00	21.74	22.09	9.28	8.33	26.92	11.76



Ground truth: $\pm \sqrt{\frac{15}{16}}$

Base ensemble: $\pm \{\frac{5}{6}\}$

DTW ensemble: $\pm \sqrt{\frac{15}{16}}$

Model A	$\pm \sqrt{\frac{1}{5}}$	$\{\frac{1}{5}\}$	$\{\frac{1}{6}\}$	$\{\frac{1}{6}\}$
Model B	$\pm \int \frac{1}{5}$	$\{\frac{1}{5}\}$	$\{\frac{1}{6}\}$	
Model C	$\pm \sqrt{\frac{1}{5}}$	$\{\frac{1}{5}\}$	$\{\frac{1}{6}\}$	
Model D	$\pm \sqrt{\frac{1}{5}}$	$\{\frac{5}{6}\}$	$\{\frac{1}{6}\}$	
Model E	$\pm \sqrt{\frac{1}{5}}$	$\{\frac{1}{5}\}$	$\{\frac{1}{6}\}$	
Base ensemble	$\pm \sqrt{\frac{1}{5}}$	$\{\frac{5}{6}\}$	$\{\frac{6}{6}\}$	

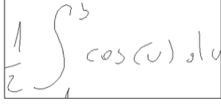
Model A	$\pm \sqrt{\frac{1}{5}}$	$\{\frac{1}{5}\}$	$\{\frac{1}{6}\}$	$\{\frac{1}{6}\}$
Model B	$\pm \int \frac{1}{5}$	$\{\frac{1}{5}\}$	$\{\frac{1}{6}\}$	
Reference	$\pm \sqrt{\frac{1}{5}}$	$\{\frac{1}{5}\}$	$\{\frac{1}{6}\}$	
Model D	$\pm \sqrt{\frac{1}{5}}$	$\{\frac{5}{6}\}$	$\{\frac{1}{6}\}$	
Model E	$\pm \sqrt{\frac{1}{5}}$	$\{\frac{1}{5}\}$	$\{\frac{1}{6}\}$	
DTW ensemble	$\pm \sqrt{\frac{1}{5}}$	$\{\frac{1}{5}\}$	$\{\frac{1}{6}\}$	$\{\frac{1}{6}\}$

Fig. 7. Comparison between existing “Time-step-based ensemble” and the proposed “DTW-based ensemble”. The former directly accumulates the candidate confidence from multiple models based on the time step; however, it produced incorrect results due to insertion and deletion errors. The proposed method first aligns the sequence results from multiple models by the DTW algorithm; thus, it can accurately sum up the confidence without being affected by the different sequence lengths.

sums up the candidate confidence from multiple models according to the time step, then using the arg-max function to gain the final prediction. Although such methods can lift the overall performance, they are susceptible to misalignment problems caused by insertion and deletion errors, which restricts further improvement. Hence, we presented the “DTW-based ensemble” algorithm to align the predictions; thereby, the confidence from sequences of different lengths can be added accurately. Benefiting from this, the proposed method significantly improved the ExpRate by 4.47%, 3.84%, and 2.67% on CROHME test splits. Fig. 7 shows an example between the regular time-based and our proposed DTW-based ensemble method. After aligning the predictions by the DTW algorithm, symbols from each time step can match at least one element from the reference sequence. Moreover, it can be applied to other sequence predicting problems.

C. Analysis of Similar Symbol Distinguishing

To further explore the effectiveness of the proposed methods on recognizing the visually similar symbols, we conducted an ablation study on eight pairs of ambiguous symbols on three CROHME test sets. Table V shows that all of the three proposed modules, *i.e.*, Path Signature Feature (PSF), Language-Model-based Rectification (LMR), and DTW-based Ensemble (DE)



Ground Truth: $\frac{1}{2} \int_0^5 \cos(u) du$

Without LM: $\frac{1}{2} \int_0^5 \cos(u) dv$

Category LM: $\frac{1}{2} \int_0^5 \cos(u) du$

Group LM: $\frac{1}{2} \int_0^5 \cos(u) dv$

Fig. 8. Failure cases of language model-based rectification module. “Group LM” cannot rectify the similar symbols belong to the same group.

can effectively reduce the misclassification errors among similar symbols. For example, the Greek alphabet alpha ‘ α ’ shares almost the same shape as the Latin letter ‘ a ’; however, the handwriting strokes are usually different. Therefore, benefiting from the PSF module that preserves both local writing details and global spatial information, our methods can easily distinguish such types of homographs, significantly reducing the error cases. In addition, for the symbols that can hardly be distinguished by shapes, the proposed LMR module further reduces the misclassification error by classifying the symbol groups. For example, the character ‘ c ’ usually represents a variable while the bracket ‘ $()$ ’ denotes grouping. Moreover, the DE module solves the misalignment issues in the model ensembling, further improving the final performance. Thanks to the three modules introduced in this paper, the proposed recognition method significantly reduces the error rate in visually similar symbols, demonstrating the effectiveness of the algorithms.

D. Comparison With State-of-The-Art Methods

Table VI compares the HMER results on the CROHME dataset. Our method achieves a superior ExpRate of 60.34%, 59.98%, and 64.22% with a single model on all three CROHME test splits, which already outperformed many ensembled methods [2], [10], [22], [57] by a large margin. When the DTM-based ensembling strategy is adopted, the proposed framework sets a new state-of-the-art with ExpRate hitting 64.81%, 63.82%, and 66.89%. Compared to the previous state-of-the-art methods TAP [8], SADA [6], and MHASD [58], our approaches significantly lift the performance by 3.65% on CROHME2014 test split, 5.76% on CROHME2016 test split, and 5.51% on the latest CROHME2019 test split, respectively.

E. Discussion

1) *Limitation of Language Model Rectification:* Fig. 8 shows the typical cases that the language model failed to rectify the

TABLE VI
COMPARISON BETWEEN PREVIOUS WORKS AND OURS ON CROHME
TEST SET (EXPRATE, %)

Methods	Year	CROHME2014	CROHME2016	CROHME2019
WYGIWYS*[33]	2017	36.41	-	-
End-to-end[54]	2017	35.19	-	-
WAP*[9]	2017	44.40	44.55	-
TAP[8]	2018	55.37	50.22	-
TAP*[8]	2018	61.16	57.02	-
PAL[35]	2018	39.66	-	-
PAL*[35]	2018	47.06	-	-
DenseMSA*[34]	2018	52.80	50.10	-
PGS[15]	2019	48.78	45.60	-
E-MAN[16]	2019	54.05	50.56	-
PAL-v2[22]	2020	48.88	49.61	-
PAL-v2*[22]	2020	54.87	57.89	-
SADA[6]	2020	56.59	54.58	-
SADA*[6]	2020	60.45	58.06	-
QD-GGA[39]	2020	32.04	32.84	-
Graphics[27]	2020	-	40.80	-
TreeDecoder[2]	2020	49.10	48.50	51.40
TreeDecoder*[2]	2020	54.00	52.10	54.60
SRD[10]	2020	50.60	46.60	45.90
SRD*[10]	2020	55.30	50.40	50.60
DualLoss[5]	2020	51.88	51.53	-
Graph-to-graph[4]	2021	54.46	52.05	-
MMSCAN[3]	2021	57.20	53.97	56.21
GlobalContext[55]	2021	-	53.44	52.38
BiTransformer[56]	2021	53.96	52.31	52.96
MRD[57]	2021	55.80	52.50	53.60
MRD*[57]	2021	57.91	54.49	56.88
MHASD[58]	2021	-	57.72	61.38
Ours	2022	60.34	59.98	64.22
Ours*	2022	64.81	63.82	66.89

* Using ensemble methods.

The bold entities represent the highest experiment results.

predictions. Specifically, it illustrates that the proposed “Group LM” cannot rectify the similar symbols belong to the same group, e.g., ‘u’ and ‘v’. However, it is important to point out that this failure case occurs infrequently, as the quantitative results still proved the effectiveness of the “Group LM” module.

2) *Feature Extraction on Implicit Symbols*: Another limitation is the prediction of implicit symbols, such as ‘^’, ‘_’, ‘{’, and ‘}’, which is one of the common issues that restricts the performance of sequence-predicting-based HMER methods. These symbols do not appear visually in MEs; however, they are very important for expressing the two-dimensional structure of MEs. Since the proposed feature enhancement module extracts path signature features from writing trajectories directly, it is unable to obtain the features for implicit symbols. Therefore, the implicit symbols are predicted based on the features extracted from explicit ones, limiting the performance of the proposed method on the MEs with complex structures.

V. CONCLUSION

In this paper, we improved handwritten mathematical expression recognition via similar symbol distinguishing. Three new modules, involving Feature Extraction Enhancement, Language Model Rectification, and Multi-Model Ensemble, are proposed. Specifically, the feature enhancement module extracts path signature features from ME writing trajectories, preserving both local writing details and global spatial information. Meanwhile,

different from existing language models that directly classify each symbol into a specific category, we train a language model to learn the symbol group according to the contextual information. Thanks to this strategy, the ambiguity issue can be avoided, and thus the visually similar symbols misclassified by the vision-based recognition model can be corrected. Furthermore, to solve the misalignment problem encountered in existing ensemble methods, we introduced a DTW-based algorithm to accurately accumulate the candidate confidence from the predicted sequences of different lengths. Experimental results on three widely-used CROHME datasets show that the proposed framework significantly improved the HMER accuracy, outperforming previous state-of-the-art methods by a large margin.

REFERENCES

- [1] R. H. Anderson, “Syntax-directed recognition of hand-printed two-dimensional mathematics,” in *Proc. Symp. Interactive Syst. Exp. Appl. Math.: Assoc. Comput. Machinery Inc. Symp.*, 1967, pp. 436–459.
- [2] J. Zhang et al., “A tree-structured decoder for image-to-markup generation,” in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 11076–11085.
- [3] J. Wang, J. Du, J. Zhang, B. Wang, and B. Ren, “Stroke constrained attention network for online handwritten mathematical expression recognition,” *Pattern Recognit.*, vol. 119, 2021, Art. no. 108047.
- [4] J.-W. Wu, F. Yin, Y.-M. Zhang, X.-Y. Zhang, and C.-L. Liu, “Graph-to-graph: Towards accurate and interpretable online handwritten mathematical expression recognition,” in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 2925–2933.
- [5] A. D. Le, “Recognizing handwritten mathematical expressions via paired dual loss attention network and printed mathematical expressions,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 566–567.
- [6] Z. Li, L. Jin, S. Lai, and Y. Zhu, “Improving attention-based handwritten mathematical expression recognition with scale augmentation and drop attention,” in *Proc. IEEE 17th Int. Conf. Front. Handwriting Recognit.*, 2020, pp. 175–180.
- [7] T.-N. Truong, C. T. Nguyen, K. M. Phan, and M. Nakagawa, “Improvement of end-to-end offline handwritten mathematical expression recognition by weakly supervised learning,” in *Proc. IEEE 17th Int. Conf. Front. Handwriting Recognit.*, 2020, pp. 181–186.
- [8] 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, pp. 221–233, 2019.
- [9] J. Zhanget al., “Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition,” *Pattern Recognit.*, vol. 71, pp. 196–206, 2017.
- [10] 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.
- [11] H. Mouchere, C. Viard-Gaudin, R. Zanibbi, and U. Garain, “ICFHR 2014 competition on recognition of on-line handwritten mathematical expressions (CROHME 2014),” in *Proc. IEEE 14th Int. Conf. Front. Handwriting Recognit.*, 2014, pp. 791–796.
- [12] H. Mouchere, C. Viard-Gaudin, R. Zanibbi, and U. Garain, “ICFHR2016 CROHME: Competition on recognition of online handwritten mathematical expressions,” in *Proc. IEEE 15th Int. Conf. Front. Handwriting Recognit.*, 2016, pp. 607–612.
- [13] M. Mahdavi, R. Zanibbi, H. Mouchere, C. Viard-Gaudin, and U. Garain, “ICDAR 2019 CROHME TFD: Competition on recognition of handwritten mathematical expressions and typeset formula detection,” in *Proc. IEEE Int. Conf. Document Anal. Recognit.*, 2019, pp. 1533–1538.
- [14] D.-H. Wang et al., “ICFHR 2020 competition on offline recognition and spotting of handwritten mathematical expressions-offrashme,” in *Proc. IEEE 17th Int. Conf. Front. Handwriting Recognit.*, 2020, pp. 211–215.
- [15] A. D. Le, B. Indurkha, and M. Nakagawa, “Pattern generation strategies for improving recognition of handwritten mathematical expressions,” *Pattern Recognit. Lett.*, vol. 128, pp. 255–262, 2019.
- [16] J. Wang, J. Du, J. Zhang, and Z.-R. Wang, “Multi-modal attention network for handwritten mathematical expression recognition,” in *Proc. IEEE Int. Conf. Document Anal. Recognit.*, 2019, pp. 1181–1186.

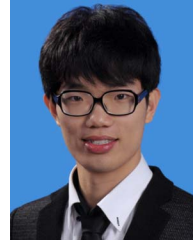
- [17] K.-T. Chen, "Integration of paths—a faithful representation of paths by noncommutative formal power series," *Trans. Amer. Math. Soc.*, vol. 89, no. 2, pp. 395–407, 1958.
- [18] T. Lyonset al., *System Control and Rough Paths*. Oxford, U.K.: Oxford Univ. Press, 2002.
- [19] B. Hambly and T. Lyons, "Uniqueness for the signature of a path of bounded variation and the reduced path group," *Ann. Math.*, vol. 171, no. 1, pp. 109–167, 2010.
- [20] D. Yuet al., "Towards accurate scene text recognition with semantic reasoning networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 12113–12122.
- [21] A. Vaswaniet al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
- [22] J.-W. Wu, F. Yin, Y.-M. Zhang, X.-Y. Zhang, and C.-L. Liu, "Handwritten mathematical expression recognition via paired adversarial learning," *Int. J. Comput. Vis.*, vol. 128, pp. 1–16, 2020.
- [23] P. Senin, "Dynamic time warping algorithm review," Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA, vol. 855, no. 40, pp. 1–23, 2008.
- [24] R. Zanibbi and D. Blostein, "Recognition and retrieval of mathematical expressions," *Int. J. Document Anal. Recognit.*, vol. 15, no. 4, pp. 331–357, 2012.
- [25] H. Mouchere, R. Zanibbi, U. Garain, and C. Viard-Gaudin, "Advancing the state of the art for handwritten math recognition: The CROHME competitions, 2011–2014," *Int. J. Document Anal. Recognit.*, vol. 19, no. 2, pp. 173–189, 2016.
- [26] S. Lavirotte and L. Pottier, "Optical formula recognition," in *Proc. IEEE 4th Int. Conf. Document Anal. Recognit.*, 1997, pp. 357–361.
- [27] F. Julca-Aguilar, H. Mouchère, C. Viard-Gaudin, and N. S. Hirata, "A general framework for the recognition of online handwritten graphics," *Int. J. Document Anal. Recognit.*, vol. 23, no. 2, pp. 143–160, 2020.
- [28] K.-F. Chan and D.-Y. Yeung, "Error detection, error correction and performance evaluation in on-line mathematical expression recognition," *Pattern Recognit.*, vol. 34, no. 8, pp. 1671–1684, 2001.
- [29] S. MacLean and G. Labahn, "A new approach for recognizing handwritten mathematics using relational grammars and fuzzy sets," *Int. J. Document Anal. Recognit.*, vol. 16, no. 2, pp. 139–163, 2013.
- [30] F. Alvaro, J.-A. Sánchez, and J.-M. Benedí, "Recognition of on-line handwritten mathematical expressions using 2D stochastic context-free grammars and hidden Markov models," *Pattern Recognit. Lett.*, vol. 35, pp. 58–67, 2014.
- [31] E. Noya, J. A. Sánchez, and J. M. Benedí, "Generation of hypergraphs from the N-best parsing of 2D-probabilistic context-free grammars for mathematical expression recognition," in *Proc. IEEE 25th Int. Conf. Pattern Recognit.*, 2021, pp. 5696–5703.
- [32] F. Álvaro, J.-A. Sánchez, and J.-M. Benedí, "An integrated grammar-based approach for mathematical expression recognition," *Pattern Recognit.*, vol. 51, pp. 135–147, 2016.
- [33] Y. Deng, A. Kanervisto, J. Ling, and A. M. Rush, "Image-to-markup generation with coarse-to-fine attention," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 980–989.
- [34] J. Zhang, J. Du, and L. Dai, "Multi-scale attention with dense encoder for handwritten mathematical expression recognition," in *Proc. IEEE 24th Int. Conf. Pattern Recognit.*, 2018, pp. 2245–2250.
- [35] J.-W. Wu, F. Yin, Y.-M. Zhang, X.-Y. Zhang, and C.-L. Liu, "Image-to-markup generation via paired adversarial learning," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2018, pp. 18–34.
- [36] J. Zhang, J. Du, and L. Dai, "A GRU-based encoder-decoder approach with attention for online handwritten mathematical expression recognition," in *Proc. IEEE 14th IAPR Int. Conf. Document Anal. Recognit.*, 2017, pp. 902–907.
- [37] T. Zhang, H. Mouchère, and C. Viard-Gaudin, "A tree-BLSTM-based recognition system for online handwritten mathematical expressions," *Neural Comput. Appl.*, vol. 32, no. 9, pp. 4689–4708, 2020.
- [38] M. Mahdavi, M. Condon, K. Davila, and R. Zanibbi, "LPGA: Line-of-sight parsing with graph-based attention for math formula recognition," in *Proc. IEEE Int. Conf. Document Anal. Recognit.*, 2019, pp. 647–654.
- [39] M. Mahdavi and R. Zanibbi, "Visual parsing with query-driven global graph attention (QD-GGA): Preliminary results for handwritten math formula recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 570–571.
- [40] H. Q. Ung, C. T. Nguyen, H. T. Nguyen, T.-N. Truong, and M. Nakagawa, "A transformer-based math language model for handwritten math expression recognition," in *Proc. Int. Conf. Document Anal. Recognit. Workshops*, 2021, pp. 403–415.
- [41] S. Fang, H. Xie, Y. Wang, Z. Mao, and Y. Zhang, "Read like humans: Autonomous, bidirectional and iterative language modeling for scene text recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 7098–7107.
- [42] B. Shi et al., "ASTER: An attentional scene text recognizer with flexible rectification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 9, pp. 2035–2048, Sep. 2019.
- [43] S. Lai, L. Jin, and W. Yang, "Online signature verification using recurrent neural network and length-normalized path signature descriptor," in *Proc. IEEE 14th IAPR Int. Conf. Document Anal. Recognit.*, 2017, pp. 400–405.
- [44] Z. Xie, Z. Sun, L. Jin, H. Ni, and T. Lyons, "Learning spatial-semantic context with fully convolutional recurrent network for online handwritten chinese text recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 8, pp. 1903–1917, Aug. 2018.
- [45] W. Yang, L. Jin, D. Tao, Z. Xie, and Z. Feng, "DropSample: A new training method to enhance deep convolutional neural networks for large-scale unconstrained handwritten chinese character recognition," *Pattern Recognit.*, vol. 58, pp. 190–203, 2016.
- [46] T. J. Lyons and N. Sidorova, "Sound compression: A rough path approach," in *Proc. 4th Int. Symp. Inf. Commun. Technol.*, 2005, pp. 223–228.
- [47] F. J. Király and H. Oberhauser, "Kernels for sequentially ordered data," *J. Mach. Learn. Res.*, vol. 20, no. 31, pp. 1–45, 2019.
- [48] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 577–585.
- [49] C. Luo, L. Jin, and Z. Sun, "MORAN: A multi-object rectified attention network for scene text recognition," *Pattern Recognit.*, vol. 90, pp. 109–118, 2019.
- [50] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. Int. Conf. Learn. Representations*, 2015.
- [51] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
- [52] Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li, "Modeling coverage for neural machine translation," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 76–85.
- [53] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.
- [54] A. D. Le and M. Nakagawa, "Training an end-to-end system for handwritten mathematical expression recognition by generated patterns," in *Proc. IEEE 14th IAPR Int. Conf. Document Anal. Recognit.*, 2017, pp. 1056–1061.
- [55] C. T. Nguyen, T.-N. Truong, H. T. Nguyen, and M. Nakagawa, "Global context for improving recognition of online handwritten mathematical expressions," in *Proc. 16th IAPR Int. Conf. Document Anal. Recognit.*, 2021, pp. 617–631.
- [56] W. Zhao et al., "Handwritten mathematical expression recognition with bidirectionally trained transformer," in *Proc. 16th IAPR Int. Conf. Document Anal. Recognit.*, 2021, pp. 570–584.
- [57] J. Wang et al., "MRD: A memory relation decoder for online handwritten mathematical expression recognition," in *Proc. Int. Conf. Document Anal. Recognit.*, 2021, pp. 39–54.
- [58] H. Ding, K. Chen, and Q. Huo, "An encoder-decoder approach to handwritten mathematical expression recognition with multi-head attention and stacked decoder," in *Proc. 16th IAPR Int. Conf. Document Anal. Recognit.*, 2021, pp. 602–616.



Zhe Li received the B.S. degree in electronics and information engineering in 2018 from the South China University of Technology, Guangzhou, China, where he is currently working toward the Ph.D. degree in information and communication engineering. His research interests include optical character recognition, handwriting recognition, and document analysis and recognition.



Xinyu Wang received the M.Phil. and Ph.D. degrees from The University of Adelaide, Adelaide SA, USA. He is currently a Postdoctoral Research Fellow with the Australian Institute for Machine Learning, The University of Adelaide. His research interests include multi-modal tasks that combine vision and language, such as visual question answering and image captioning.



Yichao Huang received the B.S. degree from the South China University of Technology, Guangzhou, China, in 2017. He is currently an Engineer with INTSIG AI Department. His research interests include optical character recognition and document intelligence.



Yuliang Liu (Member, IEEE) received the Ph.D. degree in information and communication system from the South China University of Technology, Guangzhou, China, in 2020. He is currently a Professor with the Huazhong University of Science and Technology, Wuhan, China. His research interests include artificial intelligence and computer vision research.



Kai Ding received the Ph.D. degree in information and communication system from the South China University of Technology, Guangzhou, China, in 2011. He is currently a R&D Director with IntSig Information Company, Ltd. His research interests include optical character recognition, intelligent document processing, natural language processing, and artificial intelligence applications.



Lianwen Jin (Member, IEEE) received the B.S. degree from the University of Science and Technology of China, Hefei, China, in 1991, and the Ph.D. degree from the South China University of Technology, Guangzhou, China, in 1996. He is currently a Professor with the School of Electronic and Information Engineering, South China University of Technology. He is the author of more than 100 scientific papers. His research interests include computer vision, optical character recognition, handwriting analysis and recognition, machine learning, deep learning, and intelligent systems.

Dr. Jin was the recipient of the award of New Century Excellent Talent Program of MOE in 2006, and Guangdong Pearl River Distinguished Professor Award in 2011.