

PosFormer: Recognizing Complex Handwritten Mathematical Expression with Position Forest Transformer

Tongkun Guan^{1*}, Chengyu Lin^{2*}, Wei Shen^{1(✉)}, and Xiaokang Yang¹

¹ MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
{gtk0615,wei.shen}@sjtu.edu.cn

² Paris Elite Institute of Technology, Shanghai Jiao Tong University
lacayqwq@sjtu.edu.cn

<https://github.com/SJTU-DeepVisionLab/PosFormer>

Abstract. Handwritten Mathematical Expression Recognition (HMER) has wide applications in human-machine interaction scenarios, such as digitized education and automated offices. Recently, sequence-based models with encoder-decoder architectures have been commonly adopted to address this task by directly predicting LaTeX sequences of expression images. However, these methods only implicitly learn the syntax rules provided by LaTeX, which may fail to describe the position and hierarchical relationship between symbols due to complex structural relations and diverse handwriting styles. To overcome this challenge, we propose a position forest transformer (PosFormer) for HMER, which jointly optimizes two tasks: expression recognition and position recognition, to explicitly enable position-aware symbol feature representation learning. Specifically, we first design a position forest that models the mathematical expression as a forest structure and parses the relative position relationships between symbols. Without requiring extra annotations, each symbol is assigned a position identifier in the forest to denote its relative spatial position. Second, we propose an implicit attention correction module to accurately capture attention for HMER in the sequence-based decoder architecture. Extensive experiments validate the superiority of PosFormer, which consistently outperforms the state-of-the-art methods 2.03%/1.22%/2.00%, 1.83%, and 4.62% gains on the single-line CROHME 2014/2016/2019, multi-line M²E, and complex MNE datasets, respectively, with no additional latency or computational cost.

Keywords: Handwritten mathematical expression recognition · Position forest transformer · Attention mechanism

1 Introduction

Handwritten mathematical expressions, serving as a nexus between the language and symbols, are common in fields including mathematics, physics, and chem-

*Equal contribution.

✉Corresponding author.

istry. The corresponding task, known as Handwritten Mathematical Expression Recognition (HMER), aims to accurately convert expression images into LaTeX sequences. This task has wide applications in human-machine interaction scenarios like online education, manuscript digitization, and automatic scoring. However, recognizing these expressions poses two distinct challenges: 1) the complexity of inter-symbol relationships [1], which makes the model struggle to generate appropriate structural symbols regulated by LaTeX; and 2) the diversity of handwriting inputs, including variations in scale and style.

Existing methods introduce or develop advanced components for symbol recognition and/or structural analysis to enhance recognition capabilities. They can be summarized into two branches: tree-based methods [20, 35] and sequence-based methods [34, 41, 43]. Specifically, following the syntax rules of LaTeX, tree-based methods model each mathematical expression as a tree structure [42], and then output sequences about the complete triple tuples (parent, child, parent-child relationships) of the syntax tree and decode them into a LaTeX sequence. These methods exhibit lower accuracy and poor generalization, limited by the insufficient diversity of tree structure across expressions. Sequence-based methods, which model HMER as an end-to-end image-to-sequence task [25], have gained increasing attention. They view the mathematical expression as a LaTeX sequence and employ an attention-based encoder-decoder architecture to predict each symbol in an autoregressive manner. However, these methods only implicitly learn the structure relationships between symbols and fall short in processing complex and nested mathematical expressions.

To address this issue, we propose a Position Forest Transformer (PosFormer), which explicitly models structure relationships between symbols in the attention-based encoder-decoder models to enable complex mathematical expression recognition. Specifically, we encode the LaTeX mathematical expression sequence as a position forest structure, where each symbol is assigned a position identifier to denote its relative spatial position in a two-dimensional image. Leveraging this position forest coding, we parse the nested levels and relative positions of each symbol in the forest to assist position-aware symbol-level feature representation learning in complex and nested mathematical expressions. Additionally, we introduce an implicit attention correction module in the attention-based decoder architecture to enhance attention precision. By adaptively incorporating zero attention as a refinement term, we refine attention weights with past alignment information, leading to more fine-grained feature representations.

It is noteworthy that our proposed PosFormer method just requires original HMER annotations (LaTeX sequences) without extra labelling work, and incurs no additional latency or computational cost during the inference stage. In general, our contributions are summarized as follows:

- We introduce a new method for HMER, PosFormer, which models the mathematical expression as a position forest structure and explicitly parses the relative position relationships between symbols to significantly improve the end-to-end image-to-sequence recognition capability.

- PosFormer achieves state-of-the-art (SOTA) performance on the publicly available single-line and multi-line benchmarks, with 2.03%/1.22%/2.00% and 1.83% gains on the CROHME 2014 [23]/2016 [24]/2019 [22] and M²E [34] datasets, respectively. When recognizing the complex MNE dataset, PosFormer further exhibits a remarkable average gain of 4.62%.

2 Related Work

Handwritten Mathematical Expression Recognition (HMER) has attracted considerable attention, due to the challenge of handling complicated text images with a variety of handwriting styles, structure complexity, *etc.* Traditional methods [3, 12, 14, 15, 28, 30] show low accuracy and typically involve a two-step pipeline: recognizing individual symbols and subsequently correcting them guided by grammatical rules [4]. Recently, with the development of deep learning [8, 11], two mainstream methods have been developed to improve recognition performance: tree-based methods [20, 31, 33, 35, 39, 40, 42, 45] and sequence-based methods [6, 9, 13, 16–18, 26, 29, 32, 34, 37, 38, 41, 43, 44].

Tree-based Methods. Tree-based methods view the mathematical expression as a tree structure and develop tree-structured decoders to model the hierarchical relationship based on the corresponding syntax rules. Specifically, early tree-based methods are developed to recognize online datasets, which contain writing trajectory information. BLSTM [42] proposes a pioneering work that encodes each mathematical expression into a tree by modeling symbols as nodes and relationships between symbols as edges. Utilizing the uniqueness of strokes, SRD [39] proposes a sequential relationship decoder, which improves the recognition performance of tree structures. Recently, with the development of tree-based methods, some approaches have expanded to more challenging offline HMER tasks. TSDNet [45] employs a Transformer-based multi-task learning to jointly model and predict the node attributes, edge attributes, and node connectivities, capturing the structural correlations among tree nodes. SAN [35] is proposed to enhance syntax awareness by introducing grammatical constraints.

Sequence-based Methods. Considering image features as sequences, these methods employ an autoregressive decoder framework for HMER. Specifically, a pioneering method introduced by Deng *et al.* [6] converts images into LaTeX sequences. They employ an RNN decoder to sequentially recognize symbols, with contextual priors provided by previously recognized symbols. WAP [41] utilizes a CNN for visual feature extraction and a GRU for decoding LaTeX sequences. By incorporating a coverage attention mechanism into the GRU, the model alleviates the over-translation problem. Building on this, DWAP [37] incorporates a multi-scale DenseNet [13] encoder to strengthen feature extraction and facilitate gradient propagation. CAN [18] introduces a weakly supervised counting task as an auxiliary branch to assist the recognition. Additionally, BTTR [44] is the first to apply a transformer structure for HMER, which utilizes bidirectional decoding to mitigate error accumulation. Inspired by the coverage information mechanism in RNNs, CoMER [43] further develops an attention refinement mod-

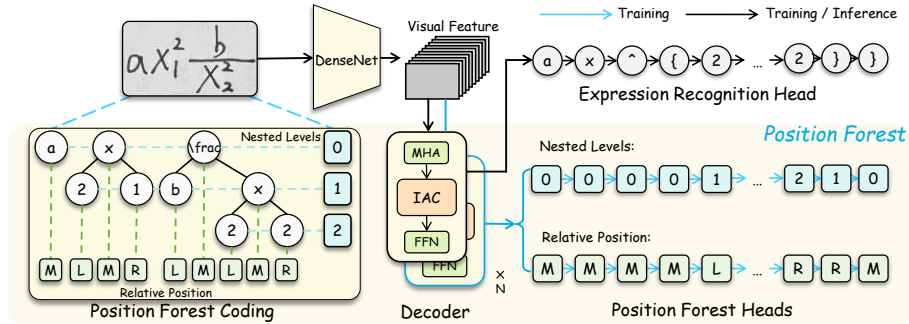


Fig. 1: Overall structure of our proposed PosFormer, which jointly optimizes two tasks: expression recognition and position recognition. The former employs parallel linear prediction for symbol recognition; the latter encodes the LaTeX sequence as a position forest structure and decodes the nested levels and relative positions of each symbol to assist in position-aware symbol-level feature representation learning.

ule to solve the coverage problem. LAST [34] further introduces a line-aware semi-autoregressive transformer to focus on multi-line recognizing tasks.

Differing from these methods, we model the mathematical expression as a position forest structure and parse the nested levels and relative positions between symbols to explicitly enable position-aware symbol-level feature representation learning in sequence-based encoder-decoder models. Accordingly, our PosFormer with position awareness can effectively generate “^”, “_”, “{” and “}” to obtain accurate LaTeX sequences, especially when recognizing complex expressions.

3 Methodology

3.1 Overview

Problem definition: Given a handwritten mathematical expression image $\mathbf{X} \in \mathbb{R}^{H \times W}$, we aim to interpret the mathematical expression and get the corresponding sequence $\mathcal{Y}_c = \{y_c^{(t)} | y_c^{(t)} \in \{“a”, “b”, \dots, “_”\}^T\}_{t=1}^T$. H, W, and T are the image height, image width, and sequence length, respectively. For simplification, these notations are the same on all images within the scope of this paper.

Network pipeline: We employ the sequence-based encoder-decoder method, CoMER [43], as our baseline model. As shown in Figure 1, the Position Forest Transformer (PosFormer) consists of a DenseNet backbone, our position forest, and an expression recognition head. 1) **Training:** Initially, the backbone extracts 2D visual features from the input image. These features are subsequently fed into the attention-based transformer decoder to obtain discriminative symbol features. A parallel linear head is then deployed to recognize the LaTeX expression. Together with expression recognition, a position forest is introduced for joint optimization to facilitate the learning of position-aware symbol-level feature representations. Specifically, this process begins by encoding the sequence

Superscript-Subscript	Fraction	Radical	Special Operator
	$\frac{a}{b}$	$\sqrt[n]{a}$ \sqrt{a}	\prod \lim \Leftrightarrow ...

Fig. 2: Four substructure types of mathematical expressions, including superscript-subscript, fraction, radical, and special operator structures.

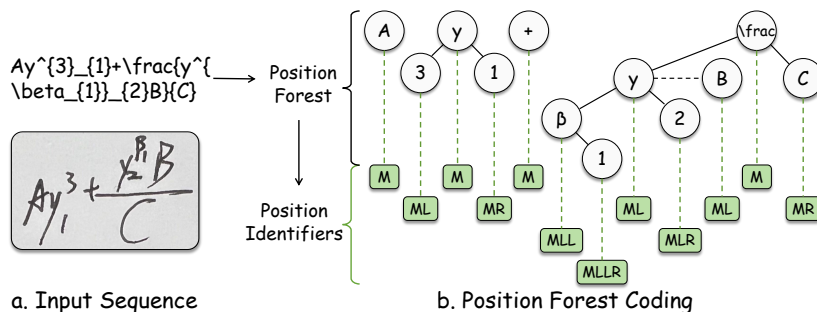


Fig. 3: Illustration of the position forest coding process, which can be simply described as sequence \rightarrow substructure \rightarrow tree \rightarrow position forest. Specifically, we encode each symbol as a position identifier to denote its relative spatial position (e.g., “MLLR”).

$\mathcal{Y}_c = \{y_c^{(t)}\}_{t=1}^T$ of mathematical expression into an identifier set $\mathcal{I} = \{\mathbf{I}_t\}_{t=1}^T$, as outlined in Algorithm 1. Each identifier denotes a string that represents its position information. Leveraging this encoding, two position forest heads are then employed to parse their nested levels and relative positions in the forest, details of which will be introduced later. 2) **Inference:** The input image is sequentially passed through the backbone, the decoder, and the expression recognition head to predict the LaTeX sequence. Note that the position forest coding and the position forest heads are removed during inference, which brings no additional latency or computational cost.

3.2 Position Forest Transformer

In this section, we will introduce the detail of our position forest transformer (PosFormer), which explicitly enhances the ability to perceive relative position relationships for recognizing complex and nested handwritten mathematical expressions. Specifically, we will introduce the position forest and the implicit attention correction module.

Position Forest Given a handwritten expression, we aim to encode the corresponding LaTeX sequence to model structure relationships between symbols, and parse them to assist the sequence-based encoder-decoder models.

1) **Position Forest Coding:** According to the syntax rules in LaTeX, expressions can be easily divided into several substructures as depicted in Figure 2,

Algorithm 1: Position Forest Coding

```

1 Input: A sequence  $\mathcal{Y}_c = \{y_c^{(t)}\}_{t=1}^T$  of length  $T$ 
2 Define the structure type set  $\{\theta_k\}_{k=1}^4 = \{“^”, “_”, “\sqrt{”}, “\frac{”}\}$ 
3 Define “M”/“L”/“R” as the identifiers for the root/left/right node of a tree
4 Define an identifier set  $\mathcal{I} = \{\mathbf{I}_t\}_{t=1}^T$  for encoding the positional relationship of all
  symbols, and initialize all elements to “M”
5 Function Substruct2Identifier( $l, r, y_c^{(l)}$ ):
6   if  $y_c^{(l)} \in \{\theta_1\}$  then
7      $\mathbf{I}_t += “L” \quad \forall t \in [l, r]$ 
8   if  $y_c^{(l)} \in \{\theta_2, \theta_3\}$  then
9      $\mathbf{I}_t += “R” \quad \forall t \in [l, r]$ 
10  if  $y_c^{(l)} \in \{\theta_4\}$  then
11     $r_1, r_2 = r$ 
12     $\mathbf{I}_t += “L” \quad \forall t \in [l, r_1]; \mathbf{I}_t += “R” \quad \forall t \in (r_1, r_2]$ 
13 Function Sequence2Substruct( $l, r$ ):
14 while  $l < r$  do
15   if  $y_c^{(l)} \in \{\theta_k\}_{k=1}^3$  then
16     Search its corresponding substructure to get the position index  $i_e \in \mathbb{R}^T$ 
17     of last symbol “}” in the substructure
18     Substruct2Identifier( $l, i_e, y_c^{(l)}$ ) // Position encoding
19     Sequence2Substruct( $l, i_e$ ) // Recursive search
20      $l \leftarrow i_e + 1$ 
21   else if  $y_c^{(l)} \in \{\theta_4\}$  then
22     Search its corresponding substructure (“\frac{””) to get two position
23     indexes  $i_{e1}, i_{e2} \in \mathbb{R}^T$  of symbol “}”
24     Substruct2Identifier( $l, (i_{e1}, i_{e2}), y_c^{(l)}$ ) // Position encoding
25     Sequence2Substruct( $l, i_{e2}$ ) // Recursive search
26      $l \leftarrow i_{e2} + 1$ 
27   else
28      $l \leftarrow l + 1$ 
29 Sequence2Substruct( $\theta, T$ )
30 Return  $\mathcal{I} = \{\mathbf{I}_t\}_{t=1}^T$  //  $\mathbf{I}_t$  denotes a string, consisting of M, L, and R

```

including superscript-subscript structures, fraction structures, radical structures, and special operator structures (*e.g.*, Product, Limit, and Equivalent).

These substructures exhibit either independent or nested relationships. Within each substructure, the relative position relationships of symbols are categorized into three types: “upper”, “lower”, and “middle”, based on their spatial positions in an image. Leveraging this prior knowledge, we model the LaTeX mathematical expression as a position forest structure. The construction follows three rules:

- 1) These substructures are encoded in a left-to-right order;
- 2) Each substructure is encoded into a tree based on the relative positions between symbols, with its main body as the root node, its upper part as the left node, and its lower part as the right node;
- 3) According to the substructure relationships, these encoded trees are arranged in series or nested to form a position forest structure.

As shown in Figure 3, we illustrate our coding procedure for better understanding. Firstly, we divide the LaTeX sequence into eight substructures: “A”, “ y_1^3 ”, “+”, “ $\frac{y_2^{\beta_1} B}{C}$ ”, “ $y_2^{\beta_1}$ ”, “ β_1 ”, “B” and “C”. The last four substructures are nested within the fraction structure and the others are independent of each other. Secondly, different symbols within a substructure will produce different branches to form a relative position tree. For instance, in the second substructure “ y_1^3 ”, the power exponent “3” and the subscript “1” of the main body “y” belong to the upper and lower parts, which are encoded into the left node and right node of the corresponding tree, respectively. Once all substructures are encoded into trees, we combine them into a position forest structure. Finally, each symbol is assigned a position identifier in the forest to denote its relative spatial position.

The details of position forest coding are introduced in Algorithm 1. A LaTeX sequence $\mathcal{Y}_c = \{y_c^{(t)}\}_{t=1}^T$ is encoded to be an encoded identifier set, $\mathcal{I} = \{\mathbf{I}_t\}_{t=1}^T$, for representing the position information of all symbols within the sequence.

2) **Position Forest Decoding:** The auto-regressive decoding manner has been proven effective for the HMER task [6, 13, 18, 34, 43]. When decoding the t -th symbol, the previously identified $t-1$ symbols are regarded as priors and fed into the decoder. It operates sequentially for T steps, producing a symbol sequence of length T . Following this, we employ the type of decoder to parse the positions of all symbols. Differently, the position information of each symbol to be predicted is an identifier (*e.g.*, “MLLR”) rather than a category. To this end, we divide the position recognition task into two sub-tasks: the nested level prediction task and the relative position prediction task.

First of all, given an expression image and its corresponding identifier set $\mathcal{I} = \{\mathbf{I}_t\}_{t=1}^T$, we construct the ground truths of the nested level and relative position. Specifically, for the k -th identifier $\mathbf{I}_k = \{q_k^{(i)} | q_k^{(i)} \in \{\text{“M”}, \text{“L”}, \text{“R”}\}\}_{i=1}^{\eta_k}$, with η_k as the length of the identifier, we can easily determine that the nested level is $\eta_k - 1$ and the relative position is $q_k^{(\eta_k)}$. For example, when analyzing the identifier “MLLR”, we deduce that the symbol resides within a substructure comprising three nested levels, and its relative position is at the lower part (“R”) of the final nested substructure. Accordingly, the ground truth of the nested level is constructed as $\mathcal{Y}_n = \{y_n^{(t)} = \eta_t - 1 | y_n^{(t)} \in \{0, 1, \dots, U\}\}_{t=1}^T$, where U denotes the maximum number of nested levels. The ground truth of the relative position is constructed as $\mathcal{Y}_r = \{y_r^{(t)} = q_t^{(\eta_t)} | y_r^{(t)} \in \{\text{“M”}, \text{“L”}, \text{“R”}\}\}_{t=1}^T$.

Subsequently, considering these identifiers $\{\mathbf{I}_t\}_{t=1}^T$ with varying lengths, we pad them to a uniform length using the “[pad]” token. Additionally, we prepend the “[sos]” token and append the “[eos]” token to each identifier to signify the start and end of the identifier. The processed identifiers are organized into a matrix $\mathbf{Q} = [\mathbf{Q}_1, \mathbf{Q}_2, \dots, \mathbf{Q}_T] \in \mathbb{R}^{T \times \mathcal{Y}}$, with \mathcal{Y} denotes the uniform length.

Each vector in the matrix is transcribed into C dimensional identifier embeddings through a nonlinear layer ξ , which includes a linear projection, GELU activation, and layer normalization. Following the previous transformer-based works [10, 27, 43], a common absolute position encodings $\mathbf{Q}_{\text{pos}} \in \mathbb{R}^{T \times C}$ of symbol orders are added to the identifier embeddings. Thus, the generation of the

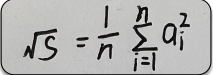
Latex sequence	Substructure Type	Expressions	Structure Symbols
$\sqrt{S} = \frac{1}{n} \sum_{i=1}^n a^{(2)}_{(i)}$	superscript-subscript	$a^{(2)}_{(i)}$	$\wedge, _ , \{ , \}$
	fraction	$\frac{1}{n}$	$\{ , \}$
	radical	\sqrt{S}	$\{$
	special operator	$\sum_{i=1}^n$	$\wedge, _ , \{ , \}$

Fig. 4: Illustration of structure symbols which are used to describe the position and hierarchical relationships between symbols in LaTeX.

identifier embedding vector is formulated in the following:

$$\mathbf{Q}_{\text{emb}} = [\xi(\mathbf{Q}_1); \xi(\mathbf{Q}_2); \dots; \xi(\mathbf{Q}_L)] + \mathbf{Q}_{\text{pos}} \quad (1)$$

where $[\cdot]$ refers to the concatenation operation.

These embedding vectors $\mathbf{Q}_{\text{emb}} \in \mathbb{R}^{T \times C}$, along with visual features $\mathbf{V}_{\text{vis}} \in \mathbb{R}^{H' \times W' \times C}$ extracted by the backbone, where $\frac{H'}{H} = \frac{W'}{W} = 16$, are fed into a three-layer Transformer-based decoder blocks. These blocks primarily consist of Multi-head Attention (MHA), Implicit Attention Correction (IAC, introduced later), and Feed-Forward Network (FFN), processing these inputs to produce output features $\mathbf{F} \in \mathbb{R}^{T \times C}$ for predicting nested levels and relative positions.

Specifically, at decoding step t , $\mathbf{F}_t \in \mathbb{R}^C$ is taken for predicting the current-step nested level

$$\begin{cases} p(y_n^{(t)}) = \text{softmax}(\mathbf{W}_n \mathbf{F}_t + b_n) \\ y_n^{(t)} \sim p(y_n^{(t)}), \end{cases} \quad (2)$$

and the current-step relative position

$$\begin{cases} p(y_r^{(t)}) = \text{softmax}(\mathbf{W}_r \mathbf{F}_t + b_r) \\ y_r^{(t)} \sim p(y_r^{(t)}), \end{cases} \quad (3)$$

where \mathbf{W}_n and \mathbf{W}_r are both trainable weights.

Implicit Attention Correction In HMER, there are over a hundred LaTeX symbols in total, typically divided into two categories: 1) Entity symbols, which have corresponding entities in images; 2) Structure symbols, which have no entity in images and are used to describe the position and hierarchical relationships between entity symbols, as illustrated in Figure 4. When decoding these symbols, coverage problems, *i.e.*, over-parsing and under-parsing, limit the recognition capabilities [18, 37, 41, 43].

To address these issues within a transformer structure for HMER, CoMER [43] refines the attention weights of the current decoding step by subtracting the attention of all previous steps for parsing accurately. The corrected attention is then exploited to extract fine-grained feature representations for recognition.

However, when decoding some structure symbols, we observed that the model allocates more attention to regions that have not yet been parsed or even to the overall image for understanding structure relationships. Following the subtraction operation, this mechanism leads to inaccuracies of refined attention

in subsequent decoding steps that rely on past alignment information. For instance, consider the substructure expression “ $4^x \{x \setminus \frac{1}{4}\}$ ” illustrated in Figure 5. When decoding the structure symbol “ \wedge ” and “ $\{$ ”, the CoMER model pays more attention to regions that contain “4” and “x” to understand the superscript relationship. In the subsequent decoding entity symbol step (*e.g.* “x”), since the previous attention weights are accumulated as a refinement term, the current attention minus the refinement term will lead the model to focus on unimportant regions.

To this end, we propose a simple and effective correction solution by introducing zero attention as our refinement term. Specifically, when an entity symbol is decoded, we reset the attention weights associated with the preceding structure symbols to zero. This is easy to explain: when we encourage the model to generate precise attention for decoding the entity symbol, it suffices to subtract these attention weights from the already parsed entity symbols, given that only entity symbols are present on mathematical expression images. Therefore, denoting $\mathbf{E}_k \in \mathbb{R}^{T \times H' \times W'}$ as the attention weights produced by the k -th decoder layer ($k \in \{2, 3\}$ in the experiment), we propose an indicator function I_Ω to introduce the corresponding refinement term $\mathbf{A}_k \in \mathbb{R}^{T \times H' \times W'}$ as follows:

$$I_\Omega(y) = \begin{cases} \mathbf{1}, & \text{if } y \notin \Omega, \\ \mathbf{0}, & \text{if } y \in \Omega, \end{cases} \quad (4)$$

$$\tilde{\mathbf{E}}_k = \text{softmax}(\mathbf{E}_k), \quad (5)$$

$$\mathbf{A}_k^{(t)} = \sum_{i=1}^{t-1} (\tilde{\mathbf{E}}_k^{(i)} \odot I_\Omega(y_c^{(i)})), \quad (6)$$

$$\mathbf{A}_k = [\mathbf{A}_k^{(1)}; \mathbf{A}_k^{(2)}; \dots; \mathbf{A}_k^{(T)}] \quad (7)$$

where Ω denotes the set of these structure symbols, \odot refers to the Hadamard product, and $[\cdot]$ represents the concatenation operation along the channel.

Subsequently, following the work [43], the refinement item is introduced to indicate whether an image feature vector has been parsed or not, leading the model to pay more attention on the unparsed regions. Specifically, the corrected attention weights $\hat{\mathbf{E}}_k$ can be calculated using the following formulas:

$$\hat{\mathbf{E}}_k = \mathbf{E}_k - \phi(\mathbf{A}_k), \quad (8)$$

where $\phi(\cdot)$ denotes a convolutional layer and a linear layer to extract local coverage information. Finally, the mechanism achieves a step-wise refinement by collecting past alignment information for each step.

3.3 Loss Function

The model is trained end-to-end under a multi-task setting, whose objective is

$$L_{\text{rec}} = -\frac{1}{T} \sum_{t=1}^T y_c^{(t)} \log p(y_c^{(t)}), \quad (9)$$

4.2 Metrics

ExpRate, ≤ 1 , ≤ 2 , and ≤ 3 metrics are widely used to measure the performance in HMER. These metrics represent the expression recognition rate when we tolerate 0 to 3 symbol-level errors. We also follow [34] and adopt the Character Error Rate (CER) to evaluate the performance on the M²E dataset.

4.3 Implementation Details

Following the CoMER [43] method, we employ a DenseNet architecture as the backbone, a three-layer decoder as our position forest decoder, and a linear layer for the symbol classification. For the position recognition task, two linear layers with different output dimensions are employed, as the maximum number of nested levels is 3 and the vocabulary size of the relative position is 6, including “[sos]”, “[eos]”, “[pad]”, “M”, “L”, and “R”. For multi-line formulas, each line is first converted into several trees corresponding to the substructures. Thanks to our forest structure, the trees corresponding to different lines can be easily arranged in series to form a forest. Furthermore, we also follow the same training parameters of CoMER [43] and LAST [34], including batch size, learning rate, optimizer, and training epochs for fair comparisons on single-line CROHME-series datasets and multi-line M²E dataset, respectively. All experiments are executed on a server equipped with a single NVIDIA A800 GPU.

4.4 Comparison with State-of-the-Art Methods

Results on Single-line Datasets First, we exhibit comparison results between PosFormer and previous SOTA methods on the CROHME datasets in Table 1. Specifically, we provide performance results of PosFormer with and without scale augmentation for a fair comparison. Without scale augmentation, PosFormer surpasses the previous SOTA results by 3.19%, 4.79% and 5.88% on the CROHME 14/16/19 test set, respectively. When using the scale augmentation, PosFormer further refreshes the recognition results, achieving gains of 2.03%, 1.22%, and 2.00% in the ExpRate metric. Second, we also conduct the comparative experiments on large-scale HME100k dataset in Table 2 and PosFormer significantly exceeds previous works. Finally, PosFormer exhibits a significant performance improvement in terms of symbol-level error tolerance metrics, which demonstrates the correction ability of PosFormer when recognizing complex expressions.

Results on Multi-line Dataset Compared to the single-line CROHME-series datasets, the multi-line handwritten mathematical expression dataset, M²E, contains a larger number of images with complex structures and long sequences. To demonstrate the effectiveness and robustness of our model, we compare PosFormer with the previous SOTA methods on the M²E dataset in Table 3. Specifically, PosFormer achieves the highest performance, with an ExpRate metric of 58.33% and a CER metric of 0.0366. This represents a 2.13% and 1.83% improvement over the CoMER method and the latest LAST [34] method, respectively. The latter is specifically tailored for decoding multi-line expressions.

Table 1: Performance comparison with previous SOTA methods on CROHME 2014/2016/2019 test sets (in %). “Scale-aug” refers to the scale augmentation [19].

Dataset	Scale-aug	Model	ExpRate \uparrow	$\leq 1 \uparrow$	$\leq 2 \uparrow$	$\leq 3 \uparrow$
CROHME 14	×	DWAP [37]	50.10	-	-	-
		BTTR [44]	53.96	66.02	70.28	-
		TSDNet [45]	54.70	68.85	74.48	-
		SAN [35]	56.2	72.6	79.2	-
		CAN [18]	57.26	74.52	82.03	-
		PosFormer (ours)	60.45 _(+3.19)	77.28	83.68	87.83
	✓	Li <i>et al.</i> [19]	56.59	69.07	75.25	78.60
		CoMER [43]	59.33	71.70	75.66	77.89
		BPD-Coverage [20]	60.65	-	-	-
		PosFormer (ours)	62.68 _(+2.03)	79.01	84.69	88.84
CROHME 16	×	DWAP [37]	47.50	-	-	-
		BTTR [44]	52.31	63.90	68.61	-
		TSDNet [45]	52.48	68.26	73.41	-
		SAN [35]	53.6	69.6	76.8	-
		CAN [18]	56.15	72.71	80.30	-
		PosFormer (ours)	60.94 _(+4.79)	76.72	83.87	88.06
	✓	Li <i>et al.</i> [19]	54.58	69.31	73.76	76.02
		BPD-Coverage [20]	58.50	-	-	-
		CoMER [43]	59.81	74.37	80.30	82.56
		PosFormer (ours)	61.03 _(+1.22)	77.86	84.74	89.28
CROHME 19	×	BTTR [44]	52.96	65.97	69.14	-
		SAN [35]	53.5	69.3	70.1	-
		CAN [18]	55.96	72.73	80.57	-
		TSDNet [45]	56.34	72.97	77.84	-
		PosFormer (ours)	62.22 _(+5.88)	79.40	86.57	89.99
		Ding <i>et al.</i> [7]	61.38	75.15	80.23	82.65
	✓	BPD-Coverage [20]	61.47	-	-	-
		CoMER [43]	62.97	77.40	81.40	83.07
		PosFormer (ours)	64.97 _(+2.00)	82.49	87.24	90.16

5 Ablations and Analysis

Effectiveness of Network Components. We demonstrate the performance gains brought by the two components of PosFormer, Position Forest (PF) and Implicit Attention Correction (IAC), respectively, as shown in Table 4. When the baseline model introduces PF to assist expression recognition by explicitly parsing the relative position relationships of symbols in mathematical formulas, its performance improves by 2.80%, 1.22%, and 0.83% on the CROHME 14/16/19 test sets, respectively. This improvement highlights that taking position recognition as an auxiliary task can effectively improve the recognition ability of sequence-based methods. Additionally, the performance gains significantly improve when allowing for 1 and 2 errors, indicating that the PF module effectively boosts the model’s correction capacity to recognize complex mathematical

Table 2: Comparisons with previous SOTA methods on the HME100k dataset.

Model	ExpRate \uparrow	$\leq 1 \uparrow$	$\leq 2 \uparrow$	$\leq 3 \uparrow$
SAN [35]	67.10	-	-	-
CAN [18]	67.31	82.93	89.17	-
PosFormer (ours)	69.51 _(+2.20)	84.91	90.51	93.25

Table 3: Performance comparison with previous SOTA methods on the multi-line M²E dataset. ExpRate, ≤ 1 , ≤ 2 , ≤ 3 are shown in percentage (%).

Model	ExpRate \uparrow	$\leq 1 \uparrow$	$\leq 2 \uparrow$	$\leq 3 \uparrow$	CER \downarrow
DWAP [37]	53.14	70.15	78.34	82.69	0.0517
Vanilla Transformer [27]	55.15	71.79	79.46	83.39	0.0576
WS-WAP [26]	55.20	72.13	80.13	83.98	0.0555
BTTR [44]	55.25	72.09	80.17	84.33	0.0528
ABM [2]	55.48	72.05	80.06	83.96	0.0552
CoMER [43]	56.20	73.39	81.11	84.94	0.0499
LAST [34]	56.50	72.80	80.81	84.61	0.0530
PosFormer (ours)	58.33 _(+1.83)	75.58	83.22	86.86	0.0366

Table 4: Evaluate the effectiveness of the proposed modules on the CROHME-series datasets. ‘‘PF’’ and ‘‘IAC’’ denote the Position Forest and Implicit Attention Correction module, respectively. ExpRate, ≤ 1 , ≤ 2 are shown in percentage (%).

Model	CROHME 2014			CROHME 2016			CROHME 2019		
	ExpRate \uparrow	$\leq 1 \uparrow$	$\leq 2 \uparrow$	ExpRate \uparrow	$\leq 1 \uparrow$	$\leq 2 \uparrow$	ExpRate \uparrow	$\leq 1 \uparrow$	$\leq 2 \uparrow$
baseline	59.33	71.70	75.66	59.81	74.37	80.30	62.97	77.40	81.40
+ PF	62.13 _(+2.80)	78.88	84.77	61.03 _(+1.22)	77.59	85.35	63.80 _(+0.83)	80.90	86.49
+ IAC	62.64 _(+3.31)	79.29	85.08	61.20 _(+1.39)	78.29	84.83	64.64 _(+1.67)	82.40	87.16

formulas. Building upon this, integrating the IAC module further enhances performance on the test sets, with increases of 0.51%, 0.17%, and 0.84% on the CROHME 14/16/19 test sets, respectively.

Extensibility to RNN-based methods. Position Forest (PF) can easily be encapsulated into a plug-in component. To demonstrate its robustness and generalizability, we extend it to RNN-based methods, as shown in Table 5. Specifically, we selected three mainstream methods [2, 18, 37] as baseline models to conduct comparative experiments, and used the same decoder as these methods to parse positions. As a result, ‘‘DWAP+PF’’ exhibits performance enhancements of 7.00%/8.73% on the CROHME 14/16 test sets, respectively. ‘‘ABM+PF’’ improves model accuracy by 1.26%/3.58%/0.34%, respectively. Similarly, with CAN as the baseline, ‘‘CAN+PF’’ increases recognition performance by 2.03%/2.87%/4.17%, respectively. This implies that PF can be generalized to existing RNN-based models to significantly boost their performance.

Table 5: Performance gains on CROHME-series datasets [22–24] when extended our proposed Position Forest (PF) to previous RNN-based methods. † indicates our reproduced result. ExpRate, ≤ 1 , ≤ 2 are shown in percentage (%).

Model	CROHME 2014			CROHME 2016			CROHME 2019		
	ExpRate †	≤ 1 †	≤ 2 †	ExpRate †	≤ 1 †	≤ 2 †	ExpRate †	≤ 1 †	≤ 2 †
DWAP [37]	50.10	-	-	47.50	-	-	-	-	-
DWAP+PF (ours)	57.10 _(+7.00)	73.02	80.53	56.23 _(+8.73)	72.24	81.17	57.30	75.56	82.24
ABM [2]	56.85	73.73	81.24	52.92	69.66	78.73	53.96	71.06	78.65
ABM+PF (ours)	58.11 _(+1.26)	74.54	80.02	56.50 _(+3.58)	72.89	80.47	54.30 _(+0.34)	73.90	82.90
CAN† [18]	55.27	72.41	80.43	53.97	70.27	78.03	52.96	72.31	79.82
CAN+PF (ours)	57.30 _(+2.03)	73.94	80.12	56.84 _(+2.87)	73.76	82.30	57.13 _(+4.17)	74.73	82.07

Table 6: Average ExpRate results on CHROME 14/16/19.

Model	DWAP	DWAP+RobustScanner	DWAP+PF	Model	RLFN	PosFormer
ExpRate †	49.17	53.18	56.87 _(+3.69)	ExpRate †	55.01	61.26 _(+6.25)

Other comparisons. 1) Position-aware model: we add a comparison with positional enhancement work [36] on the CROHME dataset in Table 6, and the average gain is 3.69%. 2) Language-aware model: introducing language models is a promising direction to further improve performance. The visual outputs of some HMER methods (*e.g.*, RLFN [5]) are fed into a language model [21] to implement recognition correction with linguistic context. Although PosFormer is a language-free method, it still gets a 6.25% gain as shown in Table 6.

6 Conclusion

We propose an effective position forest transformer (PosFormer) for handwritten mathematical expression recognition (HMER) by adding a component of positional understanding to sequence-based methods. For each mathematical expression, we first encode it as a forest structure without requiring extra annotations, and then parse their nested levels and relative positions in the forest. By optimizing the position recognition task to assist HMER, PosFormer explicitly enables position-aware symbol-level feature representation learning in complex and nested mathematical expressions. Extensive experiments validate the performance superiority of PosFormer without introducing additional latency or computational cost during inference. This highlights the significance of explicitly modelling the position relationship of expressions in sequence-based methods.

Acknowledgment. This work was supported by the NSFC under Grant 62176159 and 62322604, and in part by the Shanghai Municipal Science and Technology Major Project under Grant 2021SHZDZX0102. The computations in this paper were run on the AI for Science Platform supported by the Artificial Intelligence Institute at Shanghai Jiao Tong University.

References

1. Anderson, R.H.: Syntax-directed recognition of hand-printed two-dimensional mathematics. In: Symposium on interactive systems for experimental applied mathematics: Proceedings of the Association for Computing Machinery Inc. Symposium. pp. 436–459 (1967)
2. Bian, X., Qin, B., Xin, X., Li, J., Su, X., Wang, Y.: Handwritten mathematical expression recognition via attention aggregation based bi-directional mutual learning. In: AAAI. vol. 36, pp. 113–121 (2022)
3. Chan, K.F., Yeung, D.Y.: Elastic structural matching for online handwritten alphanumeric character recognition. In: ICPR. vol. 2, pp. 1508–1511 (1998)
4. Chan, K.F., Yeung, D.Y.: Mathematical expression recognition: a survey. IJDAR **3**, 3–15 (2000)
5. Chen, Z., Han, J., Yang, C., Zhou, Y.: Language model is suitable for correction of handwritten mathematical expressions recognition. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. pp. 4057–4068 (2023)
6. Deng, Y., Kanervisto, A., Ling, J., Rush, A.M.: Image-to-markup generation with coarse-to-fine attention. In: ICML. pp. 980–989 (2017)
7. Ding, H., Chen, K., Huo, Q.: An encoder-decoder approach to handwritten mathematical expression recognition with multi-head attention and stacked decoder. In: ICDAR. pp. 602–616 (2021)
8. Guan, T., Gu, C., Lu, C., Tu, J., Feng, Q., Wu, K., Guan, X.: Industrial scene text detection with refined feature-attentive network. IEEE TCSVT **32**(9), 6073–6085 (2022)
9. Guan, T., Gu, C., Tu, J., Yang, X., Feng, Q., Zhao, Y., Shen, W.: Self-supervised implicit glyph attention for text recognition. In: CVPR. pp. 15285–15294 (2023)
10. Guan, T., Shen, W., Yang, X., Feng, Q., Jiang, Z., Yang, X.: Self-supervised character-to-character distillation for text recognition. In: ICCV. pp. 19473–19484 (2023)
11. Guan, T., Shen, W., Yang, X., Wang, X., Yang, X.: Bridging synthetic and real worlds for pre-training scene text detectors. arXiv preprint arXiv:2312.05286 (2023)
12. Hu, L., Zanibbi, R.: Hmm-based recognition of online handwritten mathematical symbols using segmental k-means initialization and a modified pen-up/down feature. In: ICDAR. pp. 457–462 (2011)
13. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: CVPR. pp. 4700–4708 (2017)
14. Keshari, B., Watt, S.: Hybrid mathematical symbol recognition using support vector machines. In: ICDAR. vol. 2, pp. 859–863 (2007)
15. Kosmala, A., Rigoll, G., Lavirotte, S., Pottier, L.: On-line handwritten formula recognition using hidden markov models and context dependent graph grammars. In: ICDAR. pp. 107–110 (1999)
16. Le, A.D.: Recognizing handwritten mathematical expressions via paired dual loss attention network and printed mathematical expressions. In: CVPRW. pp. 566–567 (2020)
17. Le, A.D., Indurkha, B., Nakagawa, M.: Pattern generation strategies for improving recognition of handwritten mathematical expressions. Pattern Recognition Letters **128**, 255–262 (2019)
18. Li, B., Yuan, Y., Liang, D., Liu, X., Ji, Z., Bai, J., Liu, W., Bai, X.: When counting meets hmer: counting-aware network for handwritten mathematical expression recognition. In: ECCV. pp. 197–214 (2022)

19. Li, Z., Jin, L., Lai, S., Zhu, Y.: Improving attention-based handwritten mathematical expression recognition with scale augmentation and drop attention. In: ICFHR. pp. 175–180 (2020)
20. Li, Z., Yang, W., Qi, H., Jin, L., Huang, Y., Ding, K.: A tree-based model with branch parallel decoding for handwritten mathematical expression recognition. PR **149**, 110220 (2024)
21. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
22. Mahdavi, M., Zanibbi, R., Mouchere, H., Viard-Gaudin, C., Garain, U.: Icdar 2019 crohme+ tfd: Competition on recognition of handwritten mathematical expressions and typeset formula detection. In: ICDAR. pp. 1533–1538 (2019)
23. Mouchere, H., Viard-Gaudin, C., Zanibbi, R., Garain, U.: Icfhr 2014 competition on recognition of on-line handwritten mathematical expressions (crohme 2014). In: ICFHR. pp. 791–796 (2014)
24. Mouchère, H., Viard-Gaudin, C., Zanibbi, R., Garain, U.: Icfhr2016 crohme: Competition on recognition of online handwritten mathematical expressions. In: ICFHR. pp. 607–612 (2016)
25. Shi, B., Bai, X., Yao, C.: An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE TPAMI **39**(11), 2298–2304 (2016)
26. Truong, T.N., Nguyen, C.T., Phan, K.M., Nakagawa, M.: Improvement of end-to-end offline handwritten mathematical expression recognition by weakly supervised learning. In: ICFHR. pp. 181–186 (2020)
27. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: NeurIPS. vol. 30 (2017)
28. Vuong, B.Q., He, Y., Hui, S.C.: Towards a web-based progressive handwriting recognition environment for mathematical problem solving. Expert Systems with Applications **37**(1), 886–893 (2010)
29. Wang, J., Du, J., Zhang, J., Wang, Z.R.: Multi-modal attention network for handwritten mathematical expression recognition. In: ICDAR. pp. 1181–1186 (2019)
30. Winkler, H.J.: Hmm-based handwritten symbol recognition using on-line and off-line features. In: ICASSP. vol. 6, pp. 3438–3441 (1996)
31. Wu, C., Du, J., Li, Y., Zhang, J., Yang, C., Ren, B., Hu, Y.: Tdv2: A novel tree-structured decoder for offline mathematical expression recognition. In: AAAI. vol. 36, pp. 2694–2702 (2022)
32. Wu, J.W., Yin, F., Zhang, Y.M., Zhang, X.Y., Liu, C.L.: Handwritten mathematical expression recognition via paired adversarial learning. IJCV **128**, 2386–2401 (2020)
33. Wu, J.W., Yin, F., Zhang, Y.M., Zhang, X.Y., Liu, C.L.: Graph-to-graph: towards accurate and interpretable online handwritten mathematical expression recognition. In: AAAI. vol. 35, pp. 2925–2933 (2021)
34. Yang, W., Li, Z., Peng, D., Jin, L., He, M., Yao, C.: Read ten lines at one glance: Line-aware semi-autoregressive transformer for multi-line handwritten mathematical expression recognition. In: ACM MM. pp. 2066–2077 (2023)
35. Yuan, Y., Liu, X., Dikubab, W., Liu, H., Ji, Z., Wu, Z., Bai, X.: Syntax-aware network for handwritten mathematical expression recognition. In: CVPR. pp. 4553–4562 (2022)
36. Yue, X., Kuang, Z., Lin, C., Sun, H., Zhang, W.: Robustscanner: Dynamically enhancing positional clues for robust text recognition. In: ECCV. pp. 135–151 (2020)

37. Zhang, J., Du, J., Dai, L.: Multi-scale attention with dense encoder for handwritten mathematical expression recognition. In: ICPR. pp. 2245–2250 (2018)
38. Zhang, J., Du, J., Dai, L.: Track, attend, and parse (tap): An end-to-end framework for online handwritten mathematical expression recognition. *IEEE TMM* **21**(1), 221–233 (2018)
39. Zhang, J., Du, J., Yang, Y., Song, Y.Z., Dai, L.: Srd: A tree structure based decoder for online handwritten mathematical expression recognition. *IEEE TMM* **23**, 2471–2480 (2021)
40. Zhang, J., Du, J., Yang, Y., Song, Y.Z., Wei, S., Dai, L.: A tree-structured decoder for image-to-markup generation. In: ICML. pp. 11076–11085 (2020)
41. Zhang, J., Du, J., Zhang, S., Liu, D., Hu, Y., Hu, J., Wei, S., Dai, L.: Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition. *PR* **71**, 196–206 (2017)
42. Zhang, T., Mouchere, H., Viard-Gaudin, C.: Tree-based blstm for mathematical expression recognition. In: ICDAR. vol. 1, pp. 914–919 (2017)
43. Zhao, W., Gao, L.: Comer: Modeling coverage for transformer-based handwritten mathematical expression recognition. In: ECCV. pp. 392–408 (2022)
44. Zhao, W., Gao, L., Yan, Z., Peng, S., Du, L., Zhang, Z.: Handwritten mathematical expression recognition with bidirectionally trained transformer. In: ICDAR. pp. 570–584 (2021)
45. Zhong, S., Song, S., Li, G., Chan, S.H.G.: A tree-based structure-aware transformer decoder for image-to-markup generation. In: ACM MM. p. 5751–5760 (2022)