



# Tree-based data augmentation and mutual learning for offline handwritten mathematical expression recognition

Chen Yang, Jun Du\*, Jianshu Zhang, Changjie Wu, Mingjun Chen, Jiajia Wu

National Engineering Research Center of Speech and Language Information Processing (NERC-SLIP), University of Science and Technology of China, No. 96, JinZhai Road, Hefei, Anhui PR China

## ARTICLE INFO

### Article history:

Received 20 August 2021

Revised 13 May 2022

Accepted 16 July 2022

Available online 19 July 2022

### Keywords:

Tree-based data augmentation

Tree-based mutual learning

Encoder-decoder

Offline handwritten mathematical expression recognition

## ABSTRACT

Recently, thanks to the successful application of the attention-based encoder-decoder framework, handwritten mathematical expression recognition (HMER) has achieved significant improvement. However, HMER is still a challenging task in the handwriting recognition area, which suffers from the ambiguity of handwritten symbols, the two-dimensional structure of mathematical expressions, and the lack of labeled data. In this paper, we attempt to improve the recognition performance and generalization ability of the existing state-of-the-art method from two perspectives: data augmentation and model design. We first propose a tree-based multi-level (including symbol level, sub-expression level, and image level) data augmentation strategy, which can generate many synthetic images. Then, we present a novel encoder-decoder hybrid model via tree-based mutual learning to fully utilize the complementarity between tree decoder and string decoder. Benefitting from our data augmentation strategy, we achieve 58.47%/57.82%/62.67% and 74.45% expression recognition accuracy respectively on the CROHME14/16/19 competition datasets and the OffRaSHME20 competition dataset. Moreover, tree-based data augmentation is a key technology to our champion system for the OffRaSHME20 competition. Our tree-based mutual learning method further improves the recognition accuracy to 61.63%/59.81%/64.38% and 75.68% on these datasets. Further quantitative and qualitative analyses also demonstrate the effectiveness and robustness of our proposed methods.

© 2022 Elsevier Ltd. All rights reserved.

## 1. Introduction

Handwritten mathematical expressions (HMEs) are widely used in education, science, and engineering disciplines. Unlike English or other languages which are flat linear strings, mathematical expression (ME) is a more complex two-dimensional language with an internal tree structure. Therefore, handwritten mathematical expression recognition (HMER) not only needs to recognize mathematical semantic symbols but also needs to analyze spatial structure relationships. Besides, HMEs contain ambiguous symbols, complicated spatial structures, and handwritten styles, which further increases the challenge of recognizing mathematical formulas.

Recently, inspired by the success of sequence-to-sequence learning in many applications such as speech recognition, machine translation, and image caption [1–3], the attention-based encoder-decoder framework has been extensively adopted for HMER both in offline and online cases. Given an ME image or a tracepoint se-

quence, these approaches [4,5] are to generate the target markup representation in an end-to-end manner directly. Most of the works [5–7] are aimed at generating the string markup of a mathematical formula, i.e., LaTeX string, which is convenient to be implemented under the encoder-decoder framework. Compared with traditional methods [8–10], the encoder-decoder based LaTeX string decoders have substantially improved the expression recognition performance as they are end-to-end trainable and can be free of symbol segmentation. However, the paradigm based on the string decoder easily makes the model ignore the learning of the internal tree structure of mathematical expressions, which affects the generalization ability. To fully learn the tree structure information of mathematical expressions, Zhang et al. [11,12] proposed tree-structured decoders that can generate sub-tree structures step by step. The tree decoder improves the overall generalization ability of the recognition model and achieves state-of-the-art results since it can understand the sub-structures of an ME image rather than remember an entire LaTeX string.

Although string-markup based methods and tree-markup based methods have greatly improved the performance of HMER, they still suffer from recognition challenges caused by ambiguous sym-

\* Corresponding author.

E-mail address: [jundu@ustc.edu.cn](mailto:jundu@ustc.edu.cn) (J. Du).

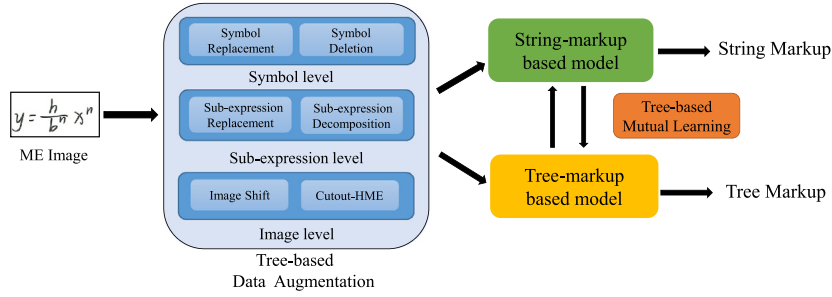


Fig. 1. The overall framework of our tree-based data augmentation and mutual learning method.

bols, complicated spatial structures, and handwritten styles. These problems can usually be alleviated to some extent by increasing the size of the training data. However, manual labeling costs are very expensive due to the complexity of mathematical expressions. In this study, we address the issues mentioned above in terms of both data augmentation and model design. The overall framework of our method is presented in Fig. 1.

To alleviate the problems of insufficient training data, we propose a tree-based multi-level data augmentation strategy that includes symbol level, sub-expression level, and image level generation. In general, image data augmentation usually creates artificial images through basic image manipulations, such as geometric transformations, color transformations, and noise injection, etc. Unfortunately, most of them are not suitable for HMER. On the one hand, typical image augmentation methods generally perform transformations globally (e.g., rotation, cropping) rather than locally in the images. On the other hand, most image augmentation methods focus on changing visual attributes while ignoring semantic consistency. Thus, our approach is concerned with generating semantically reasonable mathematical formulas. Assuming that the bounding box information of each symbol in the ME image is available, we perform various operations (e.g., replacement, deletion, and decomposition) on the image from different semantic levels, including symbol level, sub-expression level, and image level. Intuitively, the symbol level generation strategy strengthens the learning between different handwriting and ambiguous symbols by swapping symbols in different images. The sub-expression level generation strategy allows for a better understanding of local spatial relationships by replacing and decomposing subtrees. The image-level generation strategy aims to improve the accuracy of ambiguous spatial structures and to have better robustness to expressions under poor image quality. Ultimately, the mixture of the above three data types will provide a better source of training data.

In addition to generating more ME images, we also propose a simple yet effective method that can naturally incorporate the string decoder and the tree decoder during both the training and inference stage. In previous HMER competitions [13–16], training multiple models with different initialization parameters for the ensemble was an important way to improve recognition performance. To our best knowledge, this is the first study that takes advantage of the complementarity between the string decoder and the tree decoder. String decoder is data-driven; it does not explicitly employ math grammar, therefore in some bad situations, the string decoder will output the sequences that disobey the math grammar. However, the string decoder can learn an implicit string language model. In contrast, the tree decoder ensures that the output will always follow the tree-based structures, which helps alleviate bad situations. And tree decoder is more conducive to learning inherent sub-structures. Inspired by the above insights, we propose a tree-based mutual learning strategy based on the concept in [17] that allows the string decoder and the tree decoder to be trained and inferred collaboratively.

The main contributions of this paper are as follows:

- We propose a tree-based multi-level data augmentation strategy to effectively alleviate the problem of insufficient original annotation data, which is one of the critical technology to our champion system for the OffRaSHME20 competition.
- We introduce a novel tree-based mutual learning method to deeply integrate the string decoder and the tree decoder in both the training and inference stages, which fully complement the advantages of these two types of decoders.
- Our system significantly outperforms the other state-of-the-art results on both the OffRaSHME 20 dataset and the CROHME14/16/19 datasets.

## 2. Related work

In this section, we describe the related encoder-decoder based methods and data augmentation works for offline HMER.

### 2.1. Encoder-Decoder methods for offline HMER

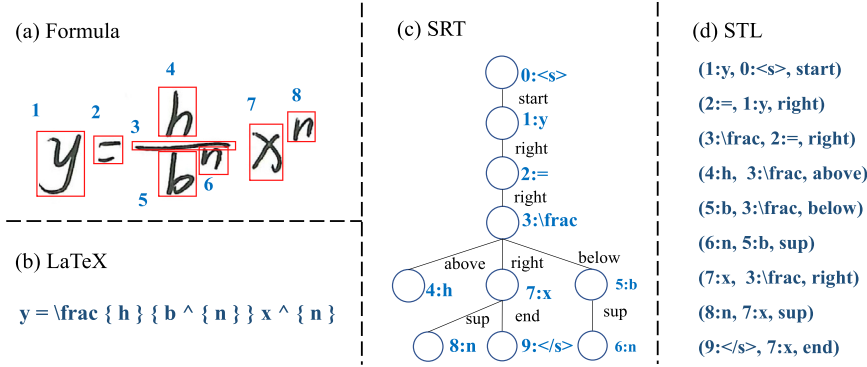
According to the forms of the generated sequence, we can divide the encoder-decoder based methods for HMER into two categories: string decoder based methods and tree decoder based methods. The former outputs a sequential LaTeX sequence, while the latter need to output a sequence of subtree structures.

In string decoder based methods, Deng et al. [5] proposed a model named WYGIWYS with coarse-to-fine attention for mathematical expression recognition and proved that the string decoder could be applied to other two-dimensional markup languages. Zhang et al. [6] introduced a model named WAP for offline HMER and achieved significant improvements in both recognition rate and efficiency compared with traditional methods. Then, an enhanced version of WAP was proposed [18], which used a DenseNet encoder and multi-scale attention mechanism. Besides, Truong et al. [19] employed a symbol classifier to improve the localization and classification of the high-level features on the encoder-decoder recognition model.

In tree decoder based methods, Zhang et al. [20] employed the tree- lstm to improve the encoding of online handwriting mathematical expression since the mathematical expression is a specific tree-structured language. Zhang et al. [11] proposed a novel tree-structured decoder with a parent decoder part and a child decoder part, which can generate sub-trees at each decoding step.

### 2.2. Data augmentation for offline HMER

Data augmentation is a potent method to boost the performance of the deep network. Typical image data augmentation methods are based on basic image manipulations, such as geometric transformations, color transformations, and noise injection. The effectiveness of these simple transformations has usually limited the context of handwritten mathematical expression recognition.



**Fig. 2.** (a) An example of handwritten mathematical expression; (b) LaTeX format representation of the expression; (c) SRT format representation of the expression; (d) STL format representation of the expression.

Recent data augmentation works for HMER have used the properties of formulas or additional information related to formulas. Le and Nakagawa [21] proposed a local and global distortion method to generate shape variations for both symbols and HMEs. And they further employed a novel decomposition method to generate different images with complicated structures from original HMEs in [22], which is based on a set of heuristic rules that conform to the LaTeX syntax. Recently, Truong et al. [23] used syntactic parse trees to represent the syntactic structure of expressions to improve the decomposition method proposed by Le et al. [22], which can provide semantic information and generate more sub-expressions. Besides, they also interchanged the decomposed sub-expressions to generate new syntactically valid HMEs. Li et al. [24] proposed a dynamic scale augmentation approach instead of normalizing MEs to the same scale. In addition, some works [25–27] adopt the idea of adversarial learning to use additional adversarial samples in training. Wu et al. [25] proposed a paired adversarial learning (PAL) to learn semantic invariant features between handwritten and printed MEs in the feature space. Then, “PAL” method is further improved by designing better encoder part and decoder part in [26]. Similar to Wu et al. [25], Le [27] proposed a dual loss attention model to learn the semantic invariant features and latex grammar by mapping context vectors between handwritten and printed MEs. Compared with previous data augmentation methods for formula recognition, our augmentation method can generate more diverse data while considering the semantic consistency of formula context for the first time during augmentation.

### 3. Tree-based data augmentation

In this section, we first introduce the different representations of mathematical expressions. Then, we introduce the data preprocessing method to extract the required annotation information for subsequent image generation. Finally, we describe the proposed tree-based multi-level data augmentation strategy in detail. We divide the mathematical expression into three levels: symbol, sub-expression, and image, according to the granularity of the structure. Benefitting from the combination of coarse-grained and fine-grained data augmentation, we improve the richness and balance of original training data, allowing for better recognition accuracies and robustness.

#### 3.1. Mathematical expression representation

Regardless of whether the recognition model is based on the traditional methods or the encoder-decoder methods, the representation of mathematical expression plays an important role, and it involves the detailed design of the model.

LaTeX string is the most commonly used mathematical markup language by researchers. It is widely used in encoder-decoder models since string generation is easier to be implemented. Nevertheless, the mathematical formula has an internal tree structure. Symbol Relation Tree (SRT) [9] is a tree markup language that describes formulas at the symbol level. In SRT, nodes represent symbols, while labels on the edges indicate the relationships between symbols. For example, in Fig. 2, the spatial relationship between the symbol “y” and the symbol “=” is “right”. “<s>” and “</s>” denote the root node of the expression tree and the last leaf node of the expression tree, respectively, which are virtual symbols that do not appear in the ME image. The common relation symbol classes are “right, above, below, inside, superscript, subscript”. In order to handle the symbols “<s>” and “</s>”, the relationship “start” and “end” are newly defined. Compared with LaTeX string generation, generating tree markup like SRT is more challenging.

To allow tree generation and easy implementation under the encoder-decoder framework, we convert the SRT into a list of triplets named Sub-tree Triplet List (STL).

Each triplet corresponds to a sub-tree structure, consisting of a child symbol node, a parent symbol node, and a relation node. And we denote it as  $(y^c, y^p, y^{re})$ . The child symbol node and the parent symbol node describe the absolute spatial positions of the child math symbol and parent math symbol, respectively, and the relation node represents the spatial relationship between the two math symbols. Compared with LaTeX string, STL format representation has two distinctive advantages: (i) it facilitates us to design neural modules separately to output structured predictions at each decoding step; (ii) the length of it is the same as the LaTeX sequence without mute symbols (e.g., “{” and “^”), which means that it is a more compact representation that shortens the decoding length. Some work has been done on this triplet representation. Zhang et al. [11,12] used triplet representation for mathematical expressions recognition while Wang and Liu [28] used triplet representation for layout analysis of mathematical expressions. Our proposed data augmentation strategy will be based on the STL representation, which motivates us to implement more heuristic rules and bind the bounding box information of symbols to mathematical symbols.

#### 3.2. Data preprocessing

Here we introduce the method to generate the candidate STL sequence set and bounding boxes sequence set from the original offline train data. Given an original ME image  $I$ , we denote its corresponding STL sequences  $S$  and bounding box sequence  $B$  as:

$$S = \{S_i = (y_i^c, y_i^p, y_i^{re})\}_{i=1}^N \quad (1)$$

**Table 1**  
Symbol classes.

Classes	Symbols	Classes	Symbols
Digit	0,1,...,9	Named operator	sin, cos, tan
Lowercase letter	a,b,...,z	Binary operator	$\times, \div, +, -, \dots$
Uppercase letter	A,B,...,Z	Relation operator	$<, >, \geq, \dots$
Greek alphabet	$\alpha, \beta, \gamma, \dots$	Others	$\exists, \iota, \lim, \{, \dots$

$$B = \{B_i = (bbox_i, y_i^c, y_i^{\text{class}})\}_{i=1}^N \quad (2)$$

where  $N$  is the total number of symbols in the STL sequences of  $I$ ,  $bbox_i$  is the 4-dimensional vector  $(a_i^{\text{tl}}, b_i^{\text{tl}}, a_i^{\text{br}}, b_i^{\text{br}})$  of the top-left and the bottom-right coordinates of bounding box. We present the classes of different symbols in Table 1. Specially,  $bbox_N$  and  $y_N^{\text{class}}$  corresponding to the last leaf node (" $</s>$ ") are set to "None" as it is a mute symbol. Besides, we denote the STL sequences and bounding box sequences corresponding to a single symbol or sub-expression as  $S_{\text{sub}}$  and  $B_{\text{sub}}$ , respectively.

Given the STL sequence  $S$  and the bounding box sequence  $B$ , the symbol-level information can be extracted directly in order, while the extraction of sub-expression level information can be done differently. A sub-expression is a part of an expression containing one or more subtree structures and is a correct expression. There are two main ways to decompose an expression into sub-expressions, based on binary operators or spatial relations.

For example, in " $x + y = z$ ", we can obtain the sub-expression " $x + y$ " and " $y = z$ " by splitting at the operator of " $=$ " and " $+$ " respectively. In the fraction " $\frac{h}{b^{n+1}}$ ", the sub-expression " $b^{n+1}$ " and the sub-expression " $n + 1$ " are extracted based on the relationship of "above" and "superscript", respectively. We can also see that sub-expressions can be split recursively because of the nested structure.

To avoid generating too short sub-expressions, we extract the sub-expression information by Algorithm 1 to ensure that sub-expressions containing nested structures are not split multiple times. The core idea is to use a list  $p_{\text{used}}$  to store the parent nodes of nested structures in sub-expressions when traversing an STL sequence of sub-expressions. As long as the parent node of a subtree is in  $p_{\text{used}}$ , the nested structure is not treated as a new sub-expression. The detailed procedure is shown in Algorithm 1. By using Algorithm 1 to process all the images in the training set, we can get the final candidate set  $C_{\text{all}}$ . We can select suitable candidate expressions from  $C_{\text{all}}$  according to different data augmentation strategies.

### 3.3. Symbol level augmentation

Symbol level generation strategy consists of *Symbol Replacement* and *Symbol Deletion*. The former enables adequate training of mathematical symbols with different handwriting styles. The latter is expected to solve the problem of bias caused by different symbol spacings in attention learning. Note that randomly selecting candidate symbols for replacement or deletion often produces ungrammatical mathematical expressions. At the same time, ungrammatical mathematical expressions will introduce noise into the original data and then affect the ability of the model to fit the data. Therefore, we design multiple rules to ensure that the generated mathematical expressions have semantic consistency and rationality.

#### 3.3.1. Symbol replacement

Correctly identifying each individual mathematical symbol in an image is the basis of HMER. However, recognizing a single mathematical symbol is not easy due to ambiguous symbols and handwriting styles [29]. For example, the Arabic letter "z" and the num-

#### Algorithm 1: Extract symbol/sub-expression information.

---

**Input:** An ME image  $I$  with its STL sequence  $S$  and bounding boxes sequence  $B$ .

**Output:** symbol/sub-expression STL and bounding boxes  $C$ .

**Initialize:**  $S_{\text{sub}}, B_{\text{sub}}, p_{\text{used}}, C$  are empty list.  $R = \{\text{above, below, inside, superscript, subscript}\}$ .

```

// extract symbol level information
1 for  $i \leftarrow 1$  to  $N$  do
2   clear  $S_{\text{sub}}$  and  $B_{\text{sub}}$ ;
3    $y_i^c, y_i^p, y_i^{\text{re}} = S_i$ ;  $bbox_i, y_i^c, y_i^{\text{class}} = B_i$ ;
4    $S_{\text{sub}} = S_i$ ;  $B_{\text{sub}} = B_i$ ;
5   extract the local image  $I_{\text{sub}}$  by  $(a_i^{\text{tl}}, b_i^{\text{tl}}, a_i^{\text{br}}, b_i^{\text{br}})$ ;
6   append  $(S_{\text{sub}}, I_{\text{sub}})$  to  $C$ ;
7 end
// extract sub-expression level information
8 for  $i \leftarrow 1$  to  $N$  do
9    $y_i^c, y_i^p, y_i^{\text{re}} = S_i$ ;  $bbox_i, y_i^c, y_i^{\text{class}} = B_i$ 
10  if  $y_i^{\text{re}} \in R$  then
11    clear  $S_{\text{sub}}, B_{\text{sub}}$  and  $p_{\text{used}}$ ;
12    append  $y_i^p$  to  $p_{\text{used}}$ ;
13    for  $j \leftarrow i$  to  $T$  do
14      if  $y_j^p \in p_{\text{used}}$  and  $y_j^{\text{re}}$  is not "End" then
15        append  $S_j, B_j, y_j^p$  to  $S_{\text{sub}}, B_{\text{sub}}, p_{\text{used}}$ , respectively;
16      end
17    end
18    compute the bounding box coordinates of
    sub-expression  $(a_{\text{sub}}^{\text{tl}}, b_{\text{sub}}^{\text{tl}}, a_{\text{sub}}^{\text{br}}, b_{\text{sub}}^{\text{br}})$  by  $B_{\text{sub}}$ ;
19    extract the local image  $I_{\text{sub}}$  by  $(a_{\text{sub}}^{\text{tl}}, b_{\text{sub}}^{\text{tl}}, a_{\text{sub}}^{\text{br}}, b_{\text{sub}}^{\text{br}})$ ;
20    append  $(S_{\text{sub}}, I_{\text{sub}})$  to  $C$ ;
21  end
22 end

```

---

ber "2" are difficult to distinguish for the model since they look similar. To address the issues about ambiguity symbols and handwriting styles, we attempt to replace math symbols in the ME image by using other symbols in  $C_{\text{all}}$ . This strategy allows the same symbols to be fully trained in different formula contexts, thereby making the model better learn the intrinsic visual features of the handwritten style symbols. To guarantee the semantic consistency of the generated images, *Symbol Replacement* will follow two rules:

- 1 Ensure that the symbols before and after replacement belong to the same symbol class.
- 2 If the same symbol appears multiple times in the expression, the symbols selected from  $C_{\text{all}}$  are used for substitution in all occurrence positions.

As shown in Fig. 3, we select the symbol "m" to replace the symbol "n" in the original expression. It is worth noting that all symbols belonging to the class "Others" in Table 1 are not considered as candidates. Besides, in practice, the replacement operation will not only replace the original STL sequence, but also replace the corresponding bounding box region image in the original image. To avoid generating visually unreasonable images after replacement, we set a selection range according to the width ("w") and height ("h") of the replaced symbol, and only symbols whose width and height belong to this range can be used as valid candidates. At the same time, we will calculate the IoU (Intersection over Union) scores between the replaced symbol and the other symbols in the original ME image. If an IoU score greater than 0.15, the replacement operation will not proceed. The selection range is set to  $[w - \min(w, h)/10, w + \min(w, h)/10]$  and  $[h - \min(w, h)/10,$



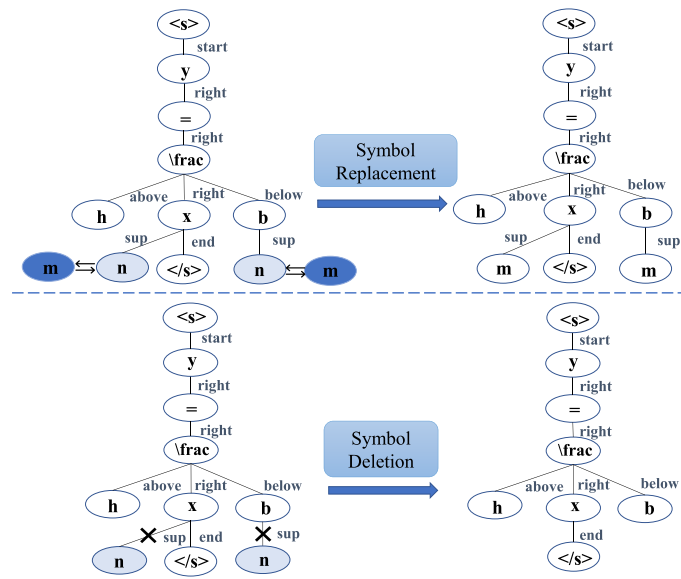


Fig. 3. Process of tree-based symbol level image generation.

$h + \min(w, h)/10]$  in our implementation, where “ $\min(w, h)$ ” represents the minimum value between the width and height of the replaced symbol. Two generated example of *Symbol Replacement* are shown in Fig. 5ig, from which we can see that after the replacement it is still semantically equivalent to the original formula.

### 3.3.2. Symbol deletion

The delete operation mimics the phenomenon of spacing between symbols, which is caused by writing speed and writing habits. Unlike replacement operation, improper deletion operations can lead to more unreasonable formulas. Therefore, for the *Symbol Deletion* strategy, we design more rules for restriction:

1. Do not delete mathematical symmetry symbols, such as “(, [, , ]”.
2. Do not delete when the left and right sides of the symbol are operators.
3. Do not delete the “Above”, “Below”, and “Inside” structures.

The generated examples are also shown in Fig. 5. We can see that the overall structure and semantics of the formula have not changed, only some symbols have been removed.

### 3.4. Sub-expression level augmentation

Sub-expression level generation strategy also includes two methods, namely *Sub-expression Replacement* and *Sub-expression Decomposition*. Under the grammar rules, an expression can be merged by combining symbols and/or sub-expressions [30,31]. Compared to the symbol level generation strategy, the sub-expression level generation strategy further enhances the richness of the training data because it can change the structural complexity of the original expressions. *Sub-expression Replacement* aims to generate mathematical expressions with higher structural complexity, while *Sub-expression Decomposition* can decompose local context in complex expressions. Both of them are expected to strengthen the model’s learning of local patterns and improve the robustness of recognizing complex expressions.

#### 3.4.1. Sub-expression replacement

In contrast to *Symbol Replacement* where rules can be effectively controlled, it is more challenging to maintain semantic consistency

for *Sub-expression Replacement* as it will be replaced with a sub-expression that contains multiple symbols at the selected position. Considering that sub-expression level generation is mainly for improving the learning of the structure, we appropriately relax the restrictions of the generation rules. First of all, we split the STL sequence of the original image according to spatial relationships as extracting sub-expression information in Section 3.2. Then, we can obtain the sub-expression positions that can be replaced. After finding the replaceable position, we then find a suitable sub-expression from  $C_{all}$  to replace. Although we discard the semantic consistency on the symbols, we require that the parent node of the candidate sub-expression is the same as the parent node of the replaced sub-expression. For example, when we want to replace the numerator and denominator of the fraction in Fig. 4, we will select the sub-expression whose parent node is “\frac” from  $C_{all}$  as the candidate. In this example, we replace the denominator “ $b$ ” with “ $a^3$ ”, and replace the numerator “ $h$ ” with “ $x^2$ ”. It can be seen that “ $a^3$ ” and “ $x^2$ ” both come from other fractions. The rule of *Sub-expression Replacement* is below:

1. The parent node of the candidate sub-expression is the same as the parent node of the replaced sub-expression.

Similar to symbol replacement, we also calculate the IoU (Intersection over Union) scores between the bounding box of replaced sub-expression and the bounding boxes of other symbols one by one, which is used to detect the size of the overlapping area. If none of the IoU scores exceeds 0.15, then we will select the candidate sub-expressions from  $C_{all}$  that match the width and height range as same as the symbol replacement. Finally, the selected candidate sub-expression is scaled to the same size as the replaced sub-expression, and then replaces the corresponding area on the image. Fig. 5 illustrates a generation sample after sub-expression replacement.

#### 3.4.2. Sub-expression decomposition

As mentioned above, a mathematical expression is composed of a sequence of sub-expressions under the grammar rules. A long expression with a complex structure can generate multiple sub-expressions. We split expressions not only by spatial relations but also by binary operators. The following Rule[SubDecompose] 1 and Rule[SubDecompose] 2 split the expressions according to

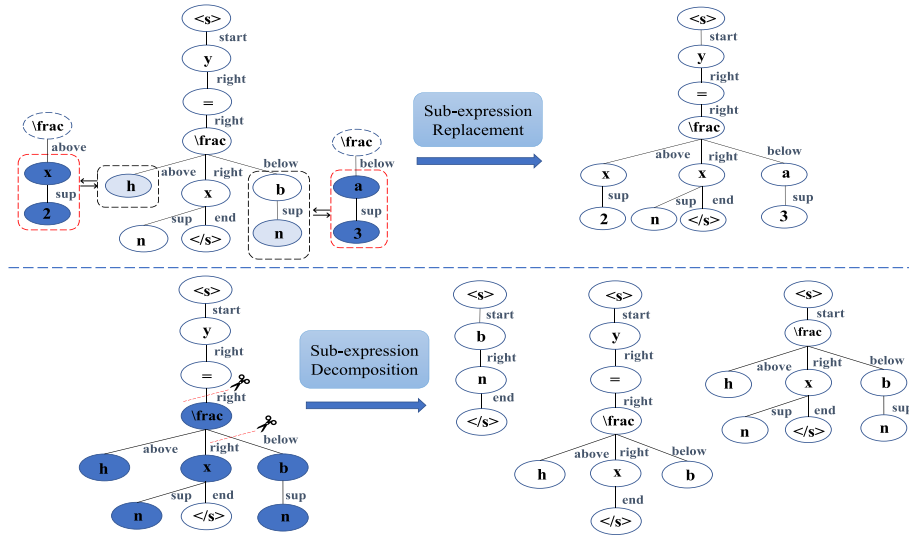


Fig. 4. Process of tree-based sub-expression level image generation.

类型	原始图像	生成图像
Symbol Replacement	$\sum_{k=1}^n k = \frac{1}{2}(n^2+n)$	$\sum_{v=1}^n v = \frac{1}{2}(n^2+n)$
Symbol Deletion	$(a^2+b^2)(c^2+d^2), (ac+bd)^2$	$(a^2+b^2)(c^2+d^2), (a+b)^2$
Sub-expression Replacement	$V_i = \sum_{j=1}^n L_{ij} \frac{dL_j}{dt}$	$V_i = \sum_{j=2}^n L_{ij} \frac{i-t}{\sqrt{2}}$
Sub-expression Decomposition	$\frac{c}{d} - \frac{a}{b} = \frac{bc-ad}{bd}$	$(bc-ad)$
Local Shift	$A^{-1}(A+B)B^{-1}$	$A^{-1}(A+B)B^{-1}$
Cutout-HME	$A^{-1}(A+B)B^{-1}$	$A^{-1}(A+B)B^{-1}$

Fig. 5. Examples of generated images from OffRaSHME dataset by symbol level, sub-expression level and image level augmentation strategies, respectively. The changed symbols are highlighted by the box.

the spatial relationship while Rule[SubDecompose] 3 splits the expressions according to the binary operators. As shown in Fig. 4, in the first example, “ $y = \frac{h}{b}x$ ” and “ $b^n$ ” are generated through Rule[SubDecompose] 1 and Rule[SubDecompose] 2, respectively. In the second example, “ $\frac{h}{b^n}x^n$ ” is split from the right side of the binary operator “ $=$ ”. In practice, for the convenience of implementation, we define the concept of the main branch of the expression tree. To be specific, the path between the root node “ $\langle s \rangle$ ” and the end node “ $\langle /s \rangle$ ” is the main branch, where the spatial relationships of all subtrees are “right” in this path. We will only choose binary operators in the main branch for decomposition operation. Our rules mainly refer to the approach in [22], which are listed as follows.

1. If an expression contains “Subscript”, “Superscript”, the new sub-expression is generated by removing all the “Subscript”, “Superscript” parts.
2. Sub-expressions are obtained by splitting the sub-expressions contained in “Subscript”, “Superscript”, “Above”, “Below” and “Inside” from STL.

3. For every binary operator in the main branch of the original expression, we generate new sub-expressions from the left and right parts of the operator.

It is worth noting that Rule[SubDecompose] 3 is not applied to binary operators inside brackets to avoid generating invalid HMEs. By the way, we will also remove the generated individual symbols (e.g., “y” in this case). A decomposed expression is shown in Fig. 5.

### 3.5. Image level augmentation

Image level generation strategy changes the ME images from local and global perspectives but never modifies the ground-truth STL sequences corresponding to the images.

#### 3.5.1. Image shift

The image shift strategy aims to solve the ambiguity problem caused by spatial position relations in mathematical expression recognition. For example, the ground-truth LaTeX of the HME Pa can be labeled as “P a” or “P \_ { a }” in different situations, which

corresponds to “right” and “subscript” between “P” and “a” respectively. This kind of ambiguity in the relationship between parent nodes and child nodes also often occurs. Based on the above observations, we proposed the local and global shift methods. “Local Shift” operation shifts up the symbol or sub-expression located at the superscript position and shifts down the symbol or sub-expression located at the subscript position. We hope to prevent the model from being deviated by some samples with ambiguous spatial relationships by adding formula images with more explicit spatial relationships. Assuming  $H$  is the symbol height of the parent node corresponding to the superscript or subscript part, then the distance of shift up or shift down is  $\alpha * H$ , and the value of  $\alpha$  is in the range [0.2, 0.3]. “Global Shift” transforms an HME by using rotation. The angles of the rotation operator are sampled from {5, 10, 15, 25, -5, -10, -15, -25}. We only show two examples of “Local Shift” in Fig. 5 as “Global Shift” is easy to associate with what the augmented image will be.

### 3.5.2. Cutout-HME

Symbols are often obscured in mathematical expressions under real scenarios. Cutout [32] is a regularization technique that has been proven effective in many computer vision tasks, and the primary motivation of the Cutout method also comes from the problem of object occlusion. The core idea of Cutout is to mask the input image during the training stage randomly. Inspired by this, we design a customized masking augmentation method for offline HMEs, named Cutout-HME. We improve the original cutout method in two aspects. On the one hand, it is inefficient to directly add random square masking regions on top of the mathematical expression images, as the background information is meaningless compared to natural scene images. Accordingly, we propose to use each symbol level bounding box as an optional range of maskable regions. On the other hand, considering that square patches often obscure important discernible parts of the symbols, we choose vertical rectangular patches for the mask.

Given an ME image, we first mark the small symbols based on the bounding box information and leave them unmasked. An adaptive vertical patch masking is generated for the remaining symbols based on the width of the corresponding original symbols. The width of the vertical patch is limited to the range [0.3-0.5] of the original symbol width. Also, for each symbol, we use a probability of 0.5 to decide whether it will be masked or not. It is worth noting that even in the absence of symbol level annotation information, the Cutout-HME method can also be implemented by connected domain analysis. Please see the ME images after performing the symbol-level mask in Fig. 5.

## 4. Tree-based mutual learning for encoder-decoder HMER

In this section, we introduce our system framework of tree-based mutual learning. We first describe the model architectures for recognizing handwritten mathematical images and then introduce the method of combining string decoder and tree decoder in both end-to-end training and model inference. As shown in Fig. 6, we use the same convolutional neural network as the encoder to extract image features, regardless of the structure of the decoder. For the string decoder in the auxiliary branch, we adopt a GRU equipped with an attention model to generate a one-dimensional LaTeX string. For the tree decoder in the main branch, it includes a parent decoder, a child decoder, and a relation prediction module so that a subtree can be generated at each time step.

### 4.1. Dense encoder

We first employ a dense convolutional network (DenseNet) [33] as the encoder to extract high-level visual features from im-

ages, which is widely adopted in various computer vision tasks. Instead of adding a fully connected layer after the final convolutional layer, the dense encoder contains only convolution, pooling and activation layers. Therefore, we can obtain a three-dimensional tensor. Then, we transform this tensor into a variable-length vector sequence as the encoder output features.

### 4.2. String decoder

Our string decoder is based on the method proposed in [6]. It takes the encoder output features as the input and generates a corresponding LaTeX sequence as shown in Fig. 6. The output sequence is denoted as  $\{s_1, s_2, \dots, s_T\}$ ,  $T$  is the sequence length. Note that the number of encoder features and the length of output sequences are both variables with different input images; we have to compute an intermediate fixed-size vector. We employ a unidirectional GRU [2] with the coverage-based spatial attention mechanism [6] to compute the fixed-size context vector. It is represented by the weighted sum of the visual features. After obtaining the context vector, another unidirectional GRU layer is adopted to produce the LaTeX sequence. More details can be referred to [6].

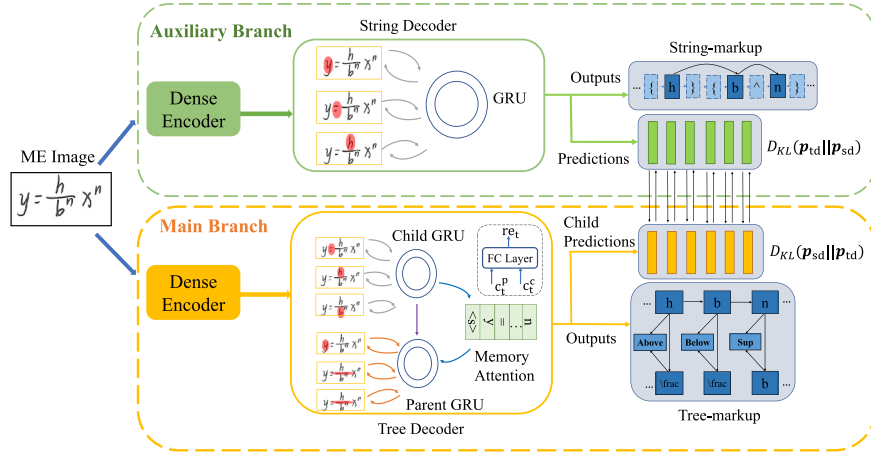
### 4.3. Tree decoder

Our tree decoder is based on the method proposed in [11]. Compared to string decoder directly generating the continuous LaTeX sequence, which consists of both symbols and spatial relationships, tree decoder divides symbol recognition and structural analysis into two parts. Specifically, the tree decoder contains a parent decoder part, a child decoder part, and a relation prediction part to produce a sequence of sub-tree structures. Each sub-tree includes a child node  $y^c$  and a parent node  $y^p$ . The target symbol sequence is denoted as  $(y_1^c, y_1^p), (y_2^c, y_2^p), \dots, (y_T^c, y_T^p)$ ; the target relation sequence is denoted as  $(y_1^{re}, y_2^{re}, \dots, y_T^{re})$ , where  $T$  is the length of sequence.

We first employ a parent decoder module to predict the current parent node, and it is composed of two GRU layers and a spatial attention model for parent node alignment. After identifying the current parent node, we employ a child decoder module to predict the current child node, and it is again made of two GRU layers and a spatial attention model for child node alignment. We propose a memory module to address the ambiguity problem caused by the repeated symbols in the formula. We employ a memory block to store each child node after they are generated. Then, each symbol will have its specific position code in the memory, where the position code is the memory index. This position code will help distinguish parent nodes. We also use memory attention model to generate an alignment between the current parent node and existing child nodes, and choose the child node with the highest attention probability to be the parent node. For the relation prediction part, as our parent decoder and child decoder both get the context vector after using the attention mechanism, which contains both the spatial information and the content information of the corresponding node, we directly concatenate the context vectors that come from the parent decoder and child decoder and then send it to a softmax layer for relation prediction. Based on the above neural network module, we can get the subtree structure sequence at every decoding step. More details can be referred to [11].

### 4.4. Tree-based mutual learning

Recognition of handwritten mathematical expressions should consider both visual cues and linguistic rules simultaneously. String decoder and tree decoder have their advantages and disadvantages in two aspects. String decoder autoregressively solves the



**Fig. 6.** The tree-based mutual learning framework. The string decoder based model as auxiliary branch (top) and the tree decoder based model as main branch (bottom), and they are connected via KL divergence and trained collaboratively. The symbol in the dashed box indicates that it will not be part of the target sequence.

symbol and spatial relationship according to the LaTeX markup order, leading to more robust language model characteristics. Therefore, sometimes string decoder can still recognize symbols based on an implicit language model when they encounter insufficient visual information (occlusive, incomplete or blurred) or ambiguous symbols. Conversely, the tree decoder pays more attention to correctly learning the inner subtree structure of the mathematical formula. It has a stronger zero-shot learning ability for formulas with complex structures than string decoder [11]. Based on the above observations, we believe that the string decoder does better than the tree decoder in symbol recognition. In contrast, the tree decoder has better structure learning capabilities than the string decoder.

To integrate the complementarity of string decoders and tree decoders, we propose a tree-based mutual learning method inspired by the idea of deep mutual learning [17]. Mutual learning of two models is a training strategy where models learn collaboratively. Unfortunately, the string decoder and the tree decoder cannot be jointly trained directly due to the presence of mute symbols. Therefore, we remove the symbols without explicit spatial alignment to input images from the LaTeX markup, and use the simplified LaTeX markup as the target sequence of the string decoder as shown in the part of “String-markup” in Fig. 6. Note that there is no mute symbol in the predicted output of the string decoder, so it cannot recover a complete and correct LaTeX sequence. Therefore we regard this branch as an auxiliary module.

Specifically, we denote the  $\Theta_{sd}$  and  $\Theta_{td}$  as the string decoder parameter and tree decoder parameter respectively. The training loss of string decoder is denoted as  $\mathcal{L}_{sd}$  while the training loss of tree decoder is represented by  $\mathcal{L}_{td}$ . To improve the generalization performance of  $\Theta_{td}$  in testing instances, we use the string decoder  $\Theta_{sd}$  to provide training experience in the form of its posterior probability  $\mathbf{p}_{sd}$  as shown in Fig. 6. To measure the match of the two networks’ predictions  $\mathbf{p}_{sd}$  and  $\mathbf{p}_{td}$ , we adopt the Kullback Leibler (KL) Divergence. The KL distance from  $\mathbf{p}_{sd}$  to  $\mathbf{p}_{td}$  is computed as:

$$D_{KL}(\mathbf{p}_{td} \parallel \mathbf{p}_{sd}) = \sum_{t=1}^T \sum_{k=1}^K p_{td}^k(y_t^c) \log \frac{p_{td}^k(y_t^c)}{p_{sd}^k(s_t)} \quad (3)$$

where  $\mathbf{p}_{sd}$ ,  $\mathbf{p}_{td}$  are the output probabilities by  $\Theta_{sd}$  and  $\Theta_{td}$ , respectively.  $T$  is the length of targeted label and  $K$  is the number of total words in the vocabulary. Note that  $s_t$  and  $y_t^c$  are equivalent when we remove the mute symbols from the LaTeX string.

We add the KL divergence to the original training loss as follows:

$$\mathcal{L}_{sd}^{ml} = D_{KL}(\mathbf{p}_{td} \parallel \mathbf{p}_{sd}) + \mathcal{L}_{sd} \quad (4)$$

$$\mathcal{L}_{td}^{ml} = D_{KL}(\mathbf{p}_{sd} \parallel \mathbf{p}_{td}) + \mathcal{L}_{td} \quad (5)$$

Then, the mutual learning strategy is performed in each mini-batch based model update step and throughout the whole training process. At each iteration, we compute the predictions of the two models and update the networks’ parameters according to the predictions of the other. The optimization of  $\Theta_{sd}$  and  $\Theta_{td}$  is conducted iteratively until convergence.

Based on our mutual learning approach, we can naturally combine the prediction probability of the tree decoder with the prediction probability of the string decoder at each inference step, while the previous string decoder based models are unable to fuse with the tree decoder based models at each time step. This tree-based mutual learning method makes full use of the complementarity between the models instead of re-scoring the output sequences of the string decoder and the tree decoder. As mentioned above, the tree decoder generates three paths in the inference stage, including the child node path, the parent node path and the relation node path, while the string decoder generates only one LaTeX sequence path without mute symbols. However, we can notice the identified target sequences corresponding to the child node path and the simplified LaTeX string path. Thus, we average the prediction probabilities of the child node path and LaTeX paths’ prediction probabilities at each inference step and use them as the fused child node prediction probabilities to generate the final STL output sequence.

## 5. Experiments

In this section, we will show the effectiveness of the proposed data generation strategies and tree-based mutual learning method for offline handwritten mathematical expression recognition. Note that our ablation studies are based on the tree decoder, which has proven to be a powerful baseline model in [11].

### 5.1. Dataset and metric

Our experiments are conducted on both the CROHME competition datasets and OffRaSHME competition dataset [13–16]. CROHME competition datasets are the most widely used dataset



for HMER. The CROHME14 competition dataset consists of a training set of 8836 HMEs and a testing set of 986 HMEs. The CRHOME16 and CRHOME19 datasets include 1147 and 1199 testing samples, respectively. We apply the CROHME14 training set as our training set and evaluate the performance of our models on the CROHME14/16/19 testing sets. Considering CROHME datasets contain the segmentation and label information of each symbol of the expression, we can easily get the corresponding bounding boxes for each symbol by using the above symbol level annotations and tracepoints.

Offline Recognition and Spotting of Handwritten Mathematical Expressions (OffRaSHME) competition provided a new HMER dataset. Unlike offline images in the CROHME dataset converted from online data, the OffRaSHME dataset was collected by scanning papers containing expressions. It contains 19,749 HMEs for training and 2000 HMEs for testing. We choose 1800 HMEs from training set for validation. Meanwhile, this offline dataset was annotated at the symbol level: the latex transcripts and the corresponding bounding boxes of symbols are provided, which can facilitate re-search of offline recognition.

The primary metric in this study is expression recognition rate (ExpRate) [34], i.e., the percentage of predicted mathematical expressions matching the ground truth. We also reported the comparison results with  $\text{ExpRate} \leq 1$ ,  $\leq 2$ , and  $\leq 3$ , which denote the expression recognition rates when one, two or three symbol-level errors are tolerable. Besides, we list the structure recognition rate (StruRate) [34], which only focuses on whether the structure is correctly recognized and ignores symbol recognition errors.

## 5.2. Implementation details

The string decoder model (auxiliary branch) is guided by minimizing  $\mathcal{L}_{sd}$  or  $\mathcal{L}_{sd}^{ml}$ , depending on whether the tree-based mutual learning method is used. Correspondingly, the tree decoder model (main branch) is guided by minimizing  $\mathcal{L}_{td}$  when training alone, while the  $\mathcal{L}_{td}^{ml}$  is used as the supervision to training collaboratively.

For the dense encoder, we first employ a 77 convolution layer with 48 output channels before entering the first dense block for the dense encoder. Each DenseBlock contains 22  $1 \times 1$  convolution layers and 22  $3 \times 3$  convolution layers. Then, we use  $1 \times 1$  convolution followed by  $2 \times 2$  average pooling as a transition layer to reduce the feature maps by half between every two DenseBlocks. And the growth rate is 16.

For the string decoder, we use the same setting employed in the WAP model [6], which contains a single layer with the hidden size 256 and an embedding layer with the size of 256. For the tree decoder, both the child and parent decoders adopt two unidirectional GRU layers, and each layer has the hidden size of 256, which is the same as in [11]. The child attention dimension, parent attention dimension, and memory attention dimension are set to 512. The embedding dimensions for both child node and parent node are set to 256.

For optimization, we adopt the cross entropy loss and the AdaDelta algorithm [35] with the hyperparameters  $\rho = 0.9$  and  $\varepsilon = 10^{-6}$ . The experiments are all implemented with PyTorch and optimized on NVIDIA TeslaV100 GPU.

## 5.3. Effectiveness of tree-based data augmentation method

In this section, we first show the performance improvements from gradually adding symbol-level generation data, sub-expression-level generation data, and image-level generation data. For the CROHME dataset, our sample size increases to approximately 137,000, 214,000, and 286,000 with the gradual addition of symbol level, image level, and sub-expression level generation

**Table 2**

Ablation study of our data augmentation strategies on CROHME14/16/19 test sets and OffRaSHME20 test set.

Dataset	Symbol	Sub-expression	Image	ExpRate	StruRate
<b>CROHME14</b>	x	x	x	49.10	68.6
	✓	x	x	<b>55.52</b>	<b>74.20</b>
	✓	x	✓	<b>57.20</b>	<b>75.68</b>
<b>CROHME16</b>	✓	✓	✓	<b>58.47</b>	<b>77.66</b>
	x	x	x	48.50	65.9
	✓	x	x	<b>54.70</b>	<b>72.51</b>
<b>CROHME19</b>	✓	x	✓	<b>56.38</b>	<b>73.54</b>
	✓	✓	✓	<b>57.82</b>	<b>76.30</b>
	x	x	x	51.40	69.8
<b>OffRaSHME20</b>	✓	x	x	<b>58.74</b>	<b>75.27</b>
	✓	x	✓	<b>60.38</b>	<b>76.14</b>
	✓	✓	✓	<b>62.67</b>	<b>79.27</b>
<b>OffRaSHME20</b>	x	x	x	69.50	–
	✓	x	x	<b>72.58</b>	–
	✓	x	✓	<b>73.65</b>	–
	✓	✓	✓	<b>74.45</b>	–

data. For the OffRaSHME dataset, our sample size increases to approximately 274,000, 375,000, and 448,000 with the gradual addition of symbol level, image level, and sub-expression level generation data. From the results of Table 2, we can see that the ExpRates are increased to 72.58%, 73.65% and 74.45% respectively with the gradual stacking of the different level augmentation strategies on the OffRaSHME20 dataset. They also receive a consistent boost on the CRHOME14/16/19 test sets. The recognition rate improved from 49.10% to 58.47% on the CROHME14, from 48.50% to 57.82% on the CROHME16 and from 51.40% to 62.67% on CROHME19. Considering that the baseline performance of CROHME datasets is lower than that of the OffRaSHME dataset, the gain of data augmentation is more significant, which brings the absolute accuracy improvements of 9.37%, 9.32%, 11.27%, respectively. After adding the sub-expression level augmented data, we can observe that the improvement of StruRate is greater than that of ExpRate on all CROHME test sets. The gains of StruRate are 2.23%, 3.42% and 4.26% on the test set of CROHME14, CROHME16 and CROHME19, respectively. Since the OffRaSHME competition does not provide a tool for testing StruRate, we have omitted this result in the table.

Furthermore, we analyze the distribution of accuracy for the length of LaTeX sequences. In Fig. 7, “baseline”, “w/o sub-expression level” and “all levels” represent the cases where the recognition model used only raw training data, used symbol level and image level generation data, and used all levels generation data, respectively. We divide the original test set into two parts by the length of expressions, where “1-20” indicates the lengths of expressions are between 1 and 20, and “+20” indicates the lengths of expressions are greater than 20. Each bar represents the recognition performance gain on the test set after using different training data. Intuitively, different lengths of expressions bring different recognition difficulties. It is interesting that our model could obtain more significant gains as the length interval grows, which confirms that our data augmentation strategy can effectively improve the recognition performance of complicated mathematical formulas. The problem of symbol recognition is well resolved with the help of symbol-level and image-level generation data, so there is a significant improvement in both subsets “1-20” and “+20”. Adding sub-expression level generation data to the low-level generation data brings further gains in subset “+20”, which illustrates the effectiveness of sub-expression generation data in helping the model learn structural information.

Besides, we know that continually increasing the data size does not constantly improve performance proportionally. Therefore, we explored the influence of the scale of data augmentation for recognition performance on the OffRaSHME20 dataset. We also develop

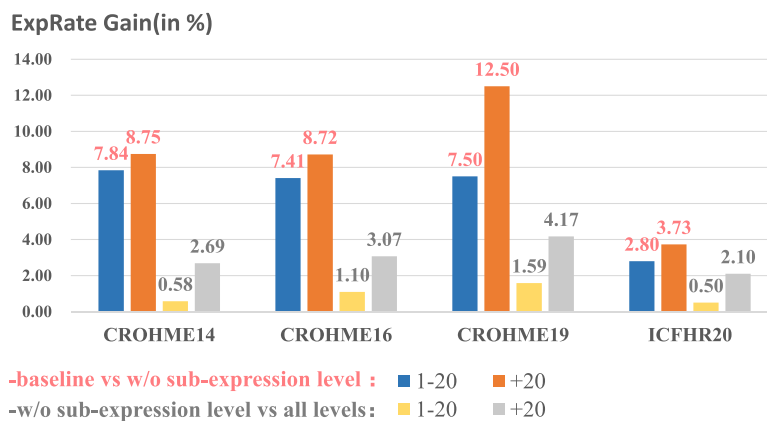


Fig. 7. Performance gain comparison of different ground-truth lengths (in %) from tree-based data augmentation on CROHME14/16/19 test sets and OffRaSHME20 test set.

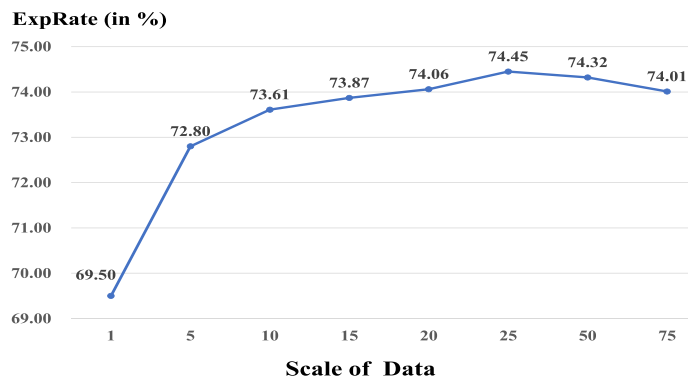


Fig. 8. Performance of different data scales (in %) on OffRaSHME20 test set.

a simple strategy to get different sizes of generated data. For *Sub-expression Decomposition*, the data that can be generated by this strategy depends on the amount of original training data. For the *Cutout-HME* method, it does not increase the size of the training set. While *Symbol Replacement*, *Symbol Deletion*, *Sub-expression Replacement* and *Image Shift*, we can apply these strategies multiple times to a sample to get a larger sample size. Therefore, we can set according to the need to get different scale data. We reach the optimal performance when we scale up the data size to 25 times the original size. After that, the accuracy with further increased training data size tends to saturate, as shown in Fig. 8.

#### 5.4. Effectiveness of tree-based mutual learning method

In this section, we first verify the effectiveness of our mutual learning strategy on the CROHME datasets and the OffRaSHME dataset. Please note that the systems in this section will be built on top of the tree-based data augmentation method. As shown in Table 3, after training with our mutual learning strategy, the ExpRates are increased to 60.50%, 58.74%, 63.92% and 75.20% on CROHME14/16/19 testing sets and OffRaSHME testing set, respectively.

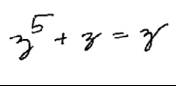
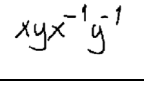
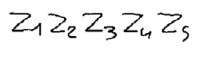
Furthermore, we utilize the string decoder and the tree decoder after using mutual learning for inference fusion. We average the prediction probability of the string decoder and the prediction probability of the child decoder during the beam search process to help improve the performance of the main branch. Compared to training multiple models with different initialization parameters but the same network structure, our method combines two different structure decoders. Finally, we achieve the ExpRate of 61.63%, 59.81%, 64.38% and 75.68% on four testing sets, respectively.

Table 3

Ablation study of tree-based mutual learning method on CROHME14/16/19 test sets and OffRaSHME20 test set.

Dataset	Mutual Learning		ExpRate	StruRate
	Training	Inference		
CROHME14	x	x	58.47	77.66
	✓	x	<b>60.50</b>	<b>77.91</b>
	x	✓	<b>61.63</b>	<b>79.03</b>
CROHME16	x	x	57.82	76.03
	✓	x	<b>58.74</b>	<b>76.30</b>
	x	✓	<b>59.81</b>	<b>76.52</b>
CROHME19	x	x	62.67	79.27
	✓	x	<b>63.92</b>	<b>80.10</b>
	x	✓	<b>64.38</b>	<b>80.23</b>
OffRaSHME20	x	x	74.45	–
	✓	x	<b>75.20</b>	–
	x	✓	<b>75.68</b>	–

We also show some recognition samples which are corrected after applying mutual learning in Fig. 9. For the first example, according to the semantic consistency, the symbols in red should be consistent with the symbols of the variables that have already appeared before. For the second example, since the multiplication symbol can never have superscripts in a normal mathematical formula context, it should be recognized as “x”. For the last example, the LaTeX ground-truth of this ME image is “ $z_{\{1\}} z_{\{2\}} z_{\{3\}} z_{\{4\}} z_{\{5\}}$ ”. It is obvious that the subscripts in the formula are numeric and continuous. Without mutual learning, the tree decoder will mistakenly recognize it as “s” where the semantic information mainly comes from the parent node “z” when predicting the current node “5”. However, when the string decoder predicts the last symbol “5”, its input hidden state still retains the

Input Image	Output LaTeX
	Before: $z^{\wedge}\{5\} + z = \backslash\text{gamma}$ After: $z^{\wedge}\{5\} + z = z$ Ground-truth: $z^{\wedge}\{5\} + z = z$
	Before: $x y \backslash\text{times}^{\wedge}\{-1\} y^{\wedge}\{-1\}$ After: $x y x^{\wedge}\{-1\} y^{\wedge}\{-1\}$ Ground-truth: $x y x^{\wedge}\{-1\} y^{\wedge}\{-1\}$
	Before: $z_{\{1\}} z_{\{2\}} z_{\{3\}} z_{\{4\}} z_{\{s\}}$ After: $z_{\{1\}} z_{\{2\}} z_{\{3\}} z_{\{4\}} z_{\{5\}}$ Ground-truth: $z_{\{1\}} z_{\{2\}} z_{\{3\}} z_{\{4\}} z_{\{5\}}$

**Fig. 9.** Examples of recognized results before and after tree-based mutual learning. The red symbols are corrected by tree-based mutual learning. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

**Table 4**

Performance comparison of our proposed method and the state-of-the-art recognition systems on CROHME14 test set (in %).

System	DA	ExpRate	$\leq 1$	$\leq 2$	$\leq 3$
PAL [25]	yes	39.66	-	-	-
PAL <sup>*5</sup> [25]	yes	47.06	-	-	-
DenseMSA <sup>*5</sup> [18]	no	52.80	68.10	72.00	72.70
PGS [22]	yes	48.78	66.13	73.94	79.01
Weakly Supervised WAP [19]	no	53.65	-	-	-
Weakly Supervised WAP <sup>*6</sup> [19]	no	55.68	-	-	-
Dual Loss Attention [27]	yes	51.88	-	-	-
PAL-v2 [26]	yes	48.88	64.50	69.78	73.83
PAL-v2 <sup>*5</sup> [26]	yes	54.87	70.69	75.76	78.60
SDG [23]	yes	55.68	-	-	-
SDG <sup>*5</sup> [23]	yes	57.20	-	-	-
Scale-Drop [24]	yes	56.59	69.07	75.25	78.60
Scale-Drop <sup>*5</sup> [24]	yes	60.45	73.43	77.69	<b>80.12</b>
Ours <sup>*2</sup>	yes	<b>61.63</b>	<b>76.64</b>	<b>79.38</b>	79.44

semantic information of the previous symbols instead of just the parent node information.

### 5.5. Comparison with the state-of-the-art

In this section, we compare our method with the current state-of-the-art methods on both the CROHME test sets and the OffRaSHME test set. In the experiment tables, \* $n$  means that an ensemble of  $n$  different recognition models and “DA” means data augmentation. Table 4 shows the performance comparison of different approaches on CROHME14 test set. “Dense MSA” [18] used an enhanced DenseNet with an extra branch to deal with different sizes of symbols. “PAL” [25], “PAL-v2” and “Dual Loss Attention” are all trained by adversarial learning. “PGS” [22] and “SDG” [23] used different strategies to expand the number of training sets, while “Scale-Drop” [24] randomly scaled the images in each training iteration. We can see that the performance of our model achieves 61.63% recognition accuracy on CROHME14 test set, which significantly outperforms the other methods. Compared to other approaches that use five or six models for ensemble, we achieve better performance with only two models (1.18% ExpRate gain compared to the “Scale-Drop<sup>\*5</sup>” method). Meanwhile, a gap existed between the correct and error percentages ( $\leq 1$ ), showing that our system still has a large room for further improvements.

To further confirm the generalization capability of our methods, we evaluate on CROHME16 competition test set in Table 5. Our system yields the best performance. It performs better than the performance of “SDG” and “SDG<sup>\*5</sup>” by 2.53% and 0.61% in terms of ExpRate, respectively. Besides, we also compare the performance with current state-of-the-art approaches on CROHME19 competition dataset in Table 6. “QD-GGA” method proposed a novel graph attention model and used multi-task learning for HMER.

**Table 5**

Performance comparison of our proposed method and the state-of-the-art recognition systems on CROHME16 test set (in %).

System	DA	ExpRate	$\leq 1$	$\leq 2$	$\leq 3$
DenseMSA <sup>*5</sup> [18]	no	52.8	63.80	67.40	68.50
PGS [22]	yes	45.60	62.25	70.44	75.76
Weakly Supervised WAP [19]	no	51.96	64.34	70.10	72.97
Weakly Supervised WAP <sup>*6</sup> [19]	no	52.57	61.26	66.33	70.80
Dual Loss Attention [27]	yes	51.53	-	-	-
PAL-v2 [26]	yes	49.61	64.08	70.27	73.50
PAL-v2 <sup>*5</sup> [26]	yes	57.89	70.44	<b>76.29</b>	<b>79.16</b>
Scale-Drop [24]	yes	54.58	69.31	73.76	76.02
Scale-Drop <sup>*5</sup> [24]	yes	58.06	71.67	75.79	77.59
SDG [23]	yes	57.28	-	-	-
SDG <sup>*5</sup> [23]	yes	59.20	-	-	-
Ours <sup>*2</sup>	yes	<b>59.81</b>	<b>73.91</b>	76.09	76.52

**Table 6**

Performance comparison of our proposed method and the state-of-the-art recognition systems on CROHME19 test set (in %).

System	DA	ExpRate	$\leq 1$	$\leq 2$	$\leq 3$
QD-GGA [36]	no	40.65	60.01	64.96	-
Univ. Linz [15]	no	41.49	54.13	58.88	-
BTTR [37]	no	52.96	65.97	69.14	-
SDG [23]	yes	54.30	-	-	-
SDG <sup>*5</sup> [23]	yes	56.13	-	-	-
WAP+MHA+StackedDecoder [38]	yes	61.38	75.15	<b>80.23</b>	<b>82.65</b>
PAL-v2 <sup>*6</sup> [15]	yes	62.89	74.98	78.40	-
Ours <sup>*2</sup>	yes	<b>64.38</b>	<b>78.01</b>	79.97	80.31

**Table 7**

Performance comparison of our proposed method and other competition recognition systems on OffRaSHME20 test set (in %).

System	DA	ExpRate	$\leq 1$	$\leq 2$	StruRate
SYSU [16]	no	61.35	77.30	81.55	82.90
HCMUS-186 [16]	no	66.95	79.65	83.60	84.00
TUAT* [16]	no	71.75	82.70	85.80	86.60
SCUT-DLVLab [16]	yes	72.90	86.05	88.80	89.45
Ours <sup>*2</sup>	yes	75.68	87.35	90.70	-
USTC-iFLYTEK*	yes	<b>79.85</b>	<b>89.85</b>	<b>92.00</b>	<b>92.55</b>

Both of the methods “BTTR” and “WAP+MHA+StackedDecoder” are equipped with multi-head attention in the network structure. “Univ. Linz” and “PAL-v2<sup>\*6</sup>” are the competition systems participating in CROHME19. Our method achieves a 3% higher ExpRate over the current method of “WAP+MHA+StackedDecoder” and also better than the system “PAL-v2<sup>\*6</sup>”.

Finally, we conduct the comparisons on the OffRaSHME20 competition test set in Table 7, which is collected from the real-offline scenario and is much more challenging. The system of “SYSU” transformed offline recognition to online recognition by applying

the stroke extraction algorithm. The team of “HCMUS-186” firstly used DenseNet-161 to extract the visual feature maps from the image and then adopted the Transformer architecture in both the encoder and decoder. TUAT’s system equipped a symbol classifier with the basic encoder-decoder model to improve the localization and classification of the CNN features. And they used the ensemble method to increase the results.

Second place team “SCUT-DLVCLab” applied scale augmentation and drop attention [24] in the training stage. We (“USTC-iFLYTEK”) used the data augmentation method in this paper as the core technology and achieved first place. The main differences between our system and the competition system in this paper are as follows. Firstly, we used an enhanced DenseNet-135 network with SE-Net [39], which can further enhance the visual feature learning ability. Secondly, we trained dozens of models for the final ensemble. And our ensemble strategy at that time was to rescore the overall path of the output LaTeX markup and the STL markup. Thirdly, we employed the training set data and our augmented data to train a GRU-based language model further to help the recognition model in the inference stage. The performance of “Ours\*2” is 75.68%, which is quite a competitive result compared with other participating systems. Since “StruRate” results in the leaderboard are officially published and the test scripts are not publicly available, our model does not show the result of this metric.

## 6. Conclusion and future work

In this paper, we proposed a novel tree-based data augmentation and tree-based mutual learning to build a better handwritten mathematical expressions recognition system. It gives significant performance improvements on both all CROHME competition datasets and the OffRaSHME competition dataset. We also demonstrated the rationality and effectiveness of the methods through experiments. In the future, we will further refine data augmentation methods to help the recognition model improve the robustness and generalization. Besides, we try to design a better network that can combine the advantages of the string decoder and the tree decoder.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

- [1] W. Chan, N. Jaitly, Q. Le, O. Vinyals, Listen, attend and spell: a neural network for large vocabulary conversational speech recognition, in: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2016, pp. 4960–4964.
- [2] K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, *Empirical Methods in Natural Language Processing (EMNLP)* (2014) 1724–1734.
- [3] L. Li, S. Tang, Y. Zhang, L. Deng, Q. Tian, GLA: global-local attention for image description, *IEEE Trans. Multimedia* 20 (3) (2017) 726–737.
- [4] J. Zhang, J. Du, L. Dai, A GRU-based encoder-decoder approach with attention for online handwritten mathematical expression recognition, in: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 1, IEEE, 2017, pp. 902–907.
- [5] Y. Deng, A. Kanervisto, J. Ling, A.M. Rush, Image-to-markup generation with coarse-to-fine attention, in: International Conference on Machine Learning, PMLR, 2017, pp. 980–989.
- [6] J. Zhang, J. Du, S. Zhang, D. Liu, Y. Hu, J. Hu, S. Wei, L. Dai, Watch, attend and parse: an end-to-end neural network based approach to handwritten mathematical expression recognition, *Pattern Recognit.* 71 (2017) 196–206.
- [7] J. Zhang, J. Du, L. Dai, Track, attend, and parse (tap): an end-to-end framework for online handwritten mathematical expression recognition, *IEEE Trans. Multimedia* 21 (1) (2018) 221–233.
- [8] K.-F. Chan, D.-Y. Yeung, Mathematical expression recognition: a survey, *Int. J. Doc. Anal. Recognit.* 3 (1) (2000) 3–15.
- [9] R. Zanibbi, D. Blostein, Recognition and retrieval of mathematical expressions, *Int. J. Doc. Anal. Recognit. (IJ DAR)* 15 (4) (2012) 331–357.
- [10] F. Álvaro, J.-A. Sánchez, J.-M. Benedí, An integrated grammar-based approach for mathematical expression recognition, *Pattern Recognit.* 51 (2016) 135–147.
- [11] J. Zhang, J. Du, Y. Yang, Y.-Z. Song, S. Wei, L. Dai, A tree-structured decoder for image-to-markup generation, in: International Conference on Machine Learning, PMLR, 2020, pp. 11076–11085.
- [12] J. Zhang, J. Du, Y. Yang, Y.-Z. Song, L. Dai, SRD: a tree structure based decoder for online handwritten mathematical expression recognition, *IEEE Trans. Multimedia* (2020).
- [13] H. Mouchère, C. Viard-Gaudin, R. Zanibbi, U. Garain, ICFHR 2014 competition on recognition of on-line handwritten mathematical expressions (CROHME 2014), in: 2014 14th International Conference on Frontiers in Handwriting Recognition, IEEE, 2014, pp. 791–796.
- [14] H. Mouchère, C. Viard-Gaudin, R. Zanibbi, U. Garain, ICFHR2016 CROHME: competition on recognition of online handwritten mathematical expressions, in: 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR), IEEE, 2016, pp. 607–612.
- [15] M. Mahdavi, R. Zanibbi, H. Mouchère, C. Viard-Gaudin, U. Garain, ICDAR 2019 CROHME+ TFD: competition on recognition of handwritten mathematical expressions and typeset formula detection, in: 2019 International Conference on Document Analysis and Recognition (ICDAR), IEEE, 2019, pp. 1533–1538.
- [16] D.-H. Wang, F. Yin, J.-W. Wu, Y.-P. Yan, Z.-C. Huang, G.-Y. Chen, Y. Wang, C.-L. Liu, ICFHR 2020 competition on offline recognition and spotting of handwritten mathematical expressions-offrashme, in: 2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR), IEEE, 2020, pp. 211–215.
- [17] Y. Zhang, T. Xiang, T.M. Hospedales, H. Lu, Deep mutual learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4320–4328.
- [18] J. Zhang, J. Du, L. Dai, Multi-scale attention with dense encoder for handwritten mathematical expression recognition, in: 2018 24th International Conference on Pattern Recognition (ICPR), IEEE, 2018, pp. 2245–2250.
- [19] T.-N. Truong, C.T. Nguyen, K.M. Phan, M. Nakagawa, Improvement of end-to-end offline handwritten mathematical expression recognition by weakly supervised learning, in: 2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR), IEEE, 2020, pp. 181–186.
- [20] T. Zhang, H. Mouchère, C. Viard-Gaudin, A tree-BLSTM-based recognition system for online handwritten mathematical expressions, *Neural Comput. Appl.* 32 (9) (2020) 4689–4708.
- [21] A.D. Le, M. Nakagawa, Training an end-to-end system for handwritten mathematical expression recognition by generated patterns, in: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 1, IEEE, 2017, pp. 1056–1061.
- [22] A.D. Le, B. Indurkha, M. Nakagawa, Pattern generation strategies for improving recognition of handwritten mathematical expressions, *Pattern Recognit. Lett.* 128 (2019) 255–262.
- [23] T.-N. Truong, C.T. Nguyen, M. Nakagawa, Syntactic data generation for handwritten mathematical expression recognition, *Pattern Recognit. Lett.* 153 (2022) 83–91.
- [24] Z. Li, L. Jin, S. Lai, Y. Zhu, Improving attention-based handwritten mathematical expression recognition with scale augmentation and drop attention, in: 2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR), IEEE, 2020, pp. 175–180.
- [25] J.-W. Wu, F. Yin, Y.-M. Zhang, X.-Y. Zhang, C.-L. Liu, Image-to-markup generation via paired adversarial learning, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2018, pp. 18–34.
- [26] J.-W. Wu, F. Yin, Y.-M. Zhang, X.-Y. Zhang, C.-L. Liu, Handwritten mathematical expression recognition via paired adversarial learning, *Int. J. Comput. Vis.* (2020) 1–16.
- [27] A.D. Le, Recognizing handwritten mathematical expressions via paired dual loss attention network and printed mathematical expressions, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 566–567.
- [28] X. Wang, J.-C. Liu, A content-constrained spatial (CCS) model for layout analysis of mathematical expressions, in: 2017 Twelfth International Conference on Digital Information Management (ICDIM), IEEE, 2017, pp. 334–339.
- [29] E.G. Miller, P.A. Viola, Ambiguity and constraint in mathematical expression recognition, in: AAAI/IAAI, 1998, pp. 784–791.
- [30] P.A. Chou, Recognition of equations using a two-dimensional stochastic context-free grammar, in: Visual Communications and Image Processing IV, vol. 1199, International Society for Optics and Photonics, 1989, pp. 852–865.
- [31] D. Průša, V. Hlaváč, Mathematical formulae recognition using 2D grammars, in: Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), vol. 2, IEEE, 2007, pp. 849–853.
- [32] T. DeVries, G.W. Taylor, Improved regularization of convolutional neural networks with cutout, *arXiv preprint arXiv:1708.04552*(2017).
- [33] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.
- [34] H. Mouchère, R. Zanibbi, U. Garain, C. Viard-Gaudin, Advancing the state of the art for handwritten math recognition: the CROHME competitions, 2011–2014, *Int. J. Doc. Anal. Recognit. (IJ DAR)* 19 (2) (2016) 173–189.
- [35] M.D. Zeiler, ADADELTA: an adaptive learning rate method, *arXiv preprint arXiv:1212.5701*(2012).



- [36] M. Mahdavi, R. Zanibbi, Visual parsing with query-driven global graph attention (QD-GGA): preliminary results for handwritten math formula recognition, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 570–571.
- [37] W. Zhao, L. Gao, Z. Yan, S. Peng, L. Du, Z. Zhang, in: Handwritten mathematical expression recognition with bidirectionally trained transformer, in: International Conference on Document Analysis and Recognition, Springer, 2021, pp. 570–584.
- [38] H. Ding, K. Chen, Q. Huo, An encoder-decoder approach to handwritten mathematical expression recognition with multi-head attention and stacked decoder, in: International Conference on Document Analysis and Recognition, Springer, 2021, pp. 602–616.
- [39] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7132–7141.



**Jun Du** received his BEng and PhD degrees from the Department of Electronic Engineering and Information Science, University of Science and Technology of China (USTC) in 2004 and 2009, respectively. From 2004 to 2009, he was with iFlytek Speech Lab of USTC. During the above years, he worked as an Intern for two 9-month periods at Microsoft Research Asia (MSRA), Beijing. In 2007, he worked as a Research Assistant for 6 months in the Department of Computer Science, University of Hong Kong. From July 2009 to June 2010, he worked at iFLYTEK Research on speech recognition. From July 2010 to January 2013, he joined MSRA as an Associate Researcher, working on handwriting recognition, OCR, and speech recognition. Since February 2013, he has been with the National Engineering Laboratory for Speech and Language Information Processing (NEL-SLIP) of USTC.



**Jianshu Zhang** received his BEng degree from the Department of Electronic Engineering and Information Science, University of Science and Technology of China (USTC) in 2015. He is currently a PhD candidate of USTC. In 2018, he worked as a visiting student for 6 months in the Queen Mary University of London. His current research areas include deep learning, handwriting mathematical expression recognition, Chinese document analysis and speech analysis.



**Chen Yang** received his BEng degree from the School of Computer Science and Information Engineering, Hefei University of Technology, in 2020. He is currently a Master's candidate at University of Science and Technology of China (USTC). His current research area includes deep learning, handwriting mathematical expression recognition and multimodal learning.