Syntax-Aware Network for Handwritten Mathematical Expression Recognition

Ye Yuan[1*], Xiao Liu[1*], Wondimu Dikubab[2], Hui Liu[1], Zhilong Ji[1], Zhongqin Wu[1], Xiang Bai[2‡]
[1]Tomorrow Advancing Life [2]Huazhong University of Science and Technology
{yuanye8, jizhilong}@tal.com, {ender.liux, wondiyeaby, ryuki122382}@gmail.com
30388514@qq.com,xbai@hust.edu.cn

Abstract

Handwritten mathematical expression recognition (HMER) is a challenging task that has many potential applications. Recent methods for HMER have achieved outstanding performance with an encoder-decoder architecture. However, these methods adhere to the paradigm that the prediction is made "from one character to another", which inevitably yields prediction errors due to the complicated structures of mathematical expressions or crabbed handwritings. In this paper, we propose a simple and efficient method for HMER, which is the first to incorporate syntax information into an encoder-decoder network. Specifically, we present a set of grammar rules for converting the LaTeX markup sequence of each expression into a parsing tree; then, we model the markup sequence prediction as a tree traverse process with a deep neural network. In this way, the proposed method can effectively describe the syntax context of expressions, alleviating the structure prediction errors of HMER. Experiments on three benchmark datasets demonstrate that our method achieves better recognition performance than prior arts. To further validate the effectiveness of our method, we create a large-scale dataset consisting of 100k handwritten mathematical expression images acquired from ten thousand writers. The source code, new dataset[†], and pre-trained models of this work will be publicly available.

1. Introduction

With the development of deep learning methods, the existing text recognition approaches are good at handling text lines in an image-to-sequence manner [24–26, 35]. However, they may fail to deal with complicated structures such as mathematical expressions (ME). This paper investigates offline handwritten mathematical expression recognition (HMER), an important OCR task required by many applications like office automation, answer sheet correction, and assistance for visually disabled persons to understand mathematics. HMER is quite challenging, as a 2D structure relationship is essential for understanding mathematical expressions, which is seldom considered in previous deep learning-based methods. Besides, the ambiguities brought by handwritten input further increase the difficulty of HMER.

Early works have well studied ME's syntax structures, and the proper grammars are defined for HMER [1, 6, 14, 34]. These grammars are only used for grouping the recognized symbols into a structural output, heavily relying on the performance of symbol recognition. Moreover, as these methods are mainly designed with handcraft features, their performance is far from the requirement of real-world applications.

Due to the recent advancement of deep neural networks, some recent studies [8, 39, 43] handle HMER as an image-to-sequence prediction procedure using an encoder-decoder architecture, achieving significant performance improvements. However, these methods more or less neglect the syntax information contained in MEs. To clearly illustrate this limitation, we take two recent network architectures as examples in Fig. 1. Zhang *et al.* [43] propose the Watch, Attend and Parse (WAP) method that employs a fully convolutional network to encode handwritten images and a recurrent neural network as the string decoder to generate sequence outputs (Fig. 1(a)). [42] (DWAP-TD) attempt to consider the syntax information by decomposing the target syntactic structure tree into a sequence of sub-trees, where each sub-tree is composed of a parent node and a child node (Fig. 1(b)). Though DWAP-TD can produce the output of a tree structure, it still follows the "from one character to another" paradigm that the next symbol prediction is mainly based on the current symbol. We argue that such methods don't explicitly consider the syntactic relationship of MEs in the learning process, which lacks the syntax constraints for generating a reasonable tree prediction.

To solve structure prediction error and improve the complex syntax tree understanding, we propose an elaborate

*Authors contribute equally.
†https://ai.100tal.com/dataset
‡Corresponding author

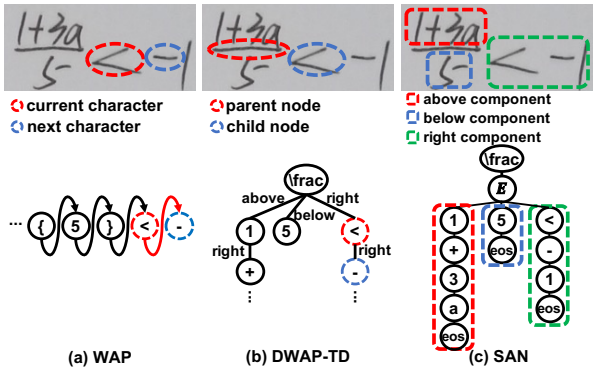

Figure 1. **Comparison of different architectures:** (a) An encoder-decoder framework WAP (b) A tree decoder DWAP-TD (c) Our model Syntax-Aware Network (SAN)

grammar, which can naturally divide a syntax tree into different components and efficiently reduce tree structure ambiguity. Then, we establish an encoder-decoder network named syntax-aware network (SAN), which incorporates grammatical constraints and feature learning in a unified framework. Our intuition is that an ideal HMER model should parse handwritten mathematical expression images according to syntactic relationships, meanwhile effectively alleviating prediction errors caused by complex structures and crabbed writings. As shown in Fig. 1 (c), the prediction process of SAN follows the traverse process of a grammar tree, whose subtree is a significant component of a mathematical expression. In this manner, the syntactic relationship of adjacent components can be encoded in the proposed SAN model. Consequently, the prediction of SAN is made from one component to another component during the parsing procedure.

To evaluate the proposed SAN, we conduct the experiments on the three popular datasets, CROHME 2014 [20], CROHME 2016 [21] and CROHME 2019 [19]. To further confirm the effectiveness of SAN, we collect and annotate a large-scale dataset for the evaluation, termed as HME100K. HME100K contains 100k handwritten mathematical expression images from ten thousand writers, mainly captured by cameras. Compared with the CHROME datasets [19–21], the data size of HME100K is increased tenfold. The results on CHROME 2014, CHROME 2016, CHROME 2019 and HME100K show that our method consistently achieves higher recognition rates over the state-of-the-art methods, demonstrating the advantage of embedding syntax cues for HMER.

The main contribution of this paper is the proposed syntax-aware network, which effectively embeds syntactic information into deep neural networks at the first time. Another contribution of this paper is the proposed large and diverse dataset HME100K. Compared with the existing benchmark datasets, our dataset includes the HME with

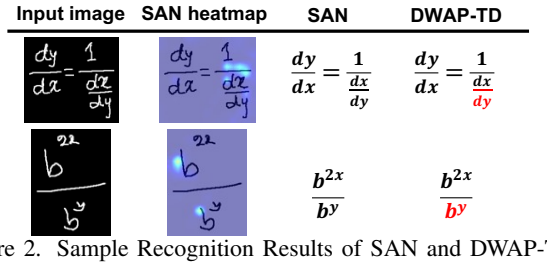

Figure 2. Sample Recognition Results of SAN and DWAP-TD. The SAN heatmaps indicate the model is focused on different components. The characters in red refer to the lost component during the prediction.

longer lengths and more complicated structures, making it useful to promote more robust algorithms toward real-world applications.

2. Related Work

HMER is a critical branches of document analysis and recognition that can be applied to recognize answer sheets of mathematics and digitize numerous kinds of scientific literature. Compared with conventional handwriting text recognition, HMER is a more challenging task due to ambiguities coming from handwriting style, the two-dimensional structures complexity, and irregular scales of handwritten mathematics symbols. Therefore, HMER can be divided into three major challenging tasks [5, 22, 36]: grouping strokes of the same symbol by segmentation, recognizing the symbols, and grammar-guided symbols structural relationship analysis to generate a mathematical expression. Traditional HMER methods tried to solve these challenges sequentially and globally.

The sequential methods [1,4,5,10,12,13,28,30,37] first segment input expression into mathematical symbols, classify each symbol separately, and then the structural relationship analysis recognize the mathematical expression. These methods employed classification techniques such as HMM [1,10,13,30], Elastic Matching [4,28], Support Vector Machines [12], and tree transformations [37]. On the other hand, the global approaches [2,3,17] apply a comprehensive strategy to learn mathematical symbols and their structural relationship analysis, while segmenting the symbols implicitly. These methods handle HMER as a global optimization of mathematical expression segmentation, symbol recognition, and structure of the expression identification based on the symbol recognition results.

Recent deep learning-based HMER methods can be roughly classified into the sequence-to-sequence approach and the tree-structure approach. Most HMER methods extensively adopt the sequence-to-sequence approach. The authors in [8, 15, 16, 23, 27, 29, 39, 40, 40, 43, 46] proposed an attention-based sequence-to-sequence model to convert the handwritten mathematical expression images into rep-

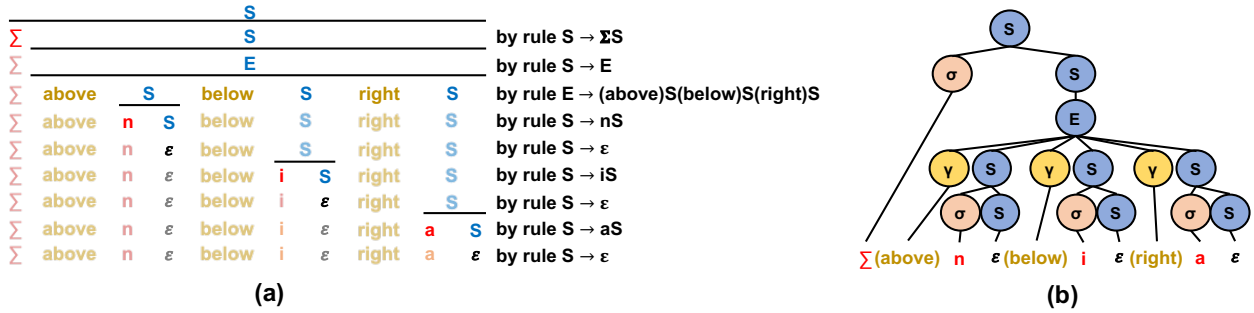

Figure 3. (a) A possible parsing procedure of $\sum_{i}^{n} a$ and (b) the parse tree. In the figure, the strings refer to non-terminal symbols in blue, terminal symbols in red, relations in yellow, and empty in grey.

resentational markup language LaTeX. Recently, Wu *et al.* [31] designed a graph-to-graph(G2G) model that explores the HMEs structural relationship of the input formula and output markup, which significantly improve the performance. The sequence-to-sequence approach improved the HMER performance. However, it lacks syntactic relationship awareness due to its inherited 1D nature, which causes inevitable prediction errors when dealing with 2D HMEs.

The tree-structure approach employs a tree-structured decoder to model the parent-child relationship of mathematical expression trees explicitly. To recognize online HME, Zhang *et al.* [44] proposed tree-structured based on BLSTM, while Zhang *et al.* [41] proposed tree-structured based on sequential relation decoder (SRD). On the other hand, to recognize offline HEMs, Zhang *et al.* [42] proposed a tree-structured decoder that attempts to consider the grammatical information by decomposing the target syntactic structure tree into a sub-tree sequence, where each sub-tree has a parent-child relationship. Generally, a tree-structured decoder exhibits better robustness compared to a string decoder. However, the existing tree-structured decoders have two limitations. 1) The lack of holistic representation of grammatical information in deep network feature learning. 2) The methods theoretically attempt to consider the syntactic information; however, they couldn't escape from the vicious circle of the sequence-to-sequence string decoding scheme in practice.

The difficulty of HMER mostly relies on considering syntactic relationship complexity rather than symbol recognition. Thus, we proposed a novel neural network architecture equipped with grammar rules that efficiently divides a syntax tree into different components to alleviate errors caused by tree structure ambiguity. The proposed method learns the grammatical relationship between the structures based on the grammar rules and navigates the syntax tree to create components according to the structural relationship. The key differences between our work and the existing tree-structured approaches are that (1) SAN integrates syntax constraints representation into deep neural network feature learning. (2) Our method is a holistic syntax-aware approach that accurately identifies and locates components of MEs with complex structural relationships. As shown in Fig. 2, SAN locates and recognizes the given MEs correctly while DWAP-TD misses a component of MEs. (3) SAN minimizes high computational costs by parsing one component to another instead of the individual parent and child nodes.

3. Syntax-Aware Network

Formally, a SAN can be defined as a 7-tuple $G = (N, \Sigma, R, S, \Gamma, C, D)$, consisting of a finite set of non-terminal symbols (N), a finite set of terminal symbols (Σ), a finite set of production rules (R), a start symbol (S), a finite set of relations (Γ), an encoder (C) and a decoder (D). We design the grammar rules with two constraints: 1) It follows the standard reading orders: left-to-right, top-to-down. 2) The spatial relations between adjacent symbols are used. For a pair of adjacent symbols of HME, there are nine possible relations in total (left, right, above, below, low left, low right, upper left, upper right, inside). Due to constraint 1), we have removed "left" and "low left", and the rest 7 relations have been sustained to deal with all the cases of MEs in our implementation. Even though a ME may correspond to different LaTeX sequences, the syntax trees generated by our grammar rules are the same due to the two constraints.

The SAN transforms an image into a parse tree, of which the leaf nodes are terminal symbols or relations, and others are non-terminal symbols. There are two non-terminal symbols S and E, where S is the start symbol and servers as the tree's root. The terminal set Σ contains all symbols that might be used in a LaTeX expression sequence.

The R production rule can be used to construct the parse tree. The production rules are with the form of $\alpha \rightarrow \beta$, where $\alpha \in N$, $\beta \in (\Gamma \cup N \cup \Sigma)^*$ and the asterisk represents the Kleene star operation. Hence, a parent node (α) can be split into a list of children nodes (β) containing terminal symbols, non-terminal symbols and relations.

R has two production rules. The first rule is an S produce 1) an arbitrary terminal symbol followed by an S on the right, 2) an E or, 3) an empty string indicated by ϵ:

$$S \rightarrow \sigma S|E|\epsilon, \tag{1}$$

where $\sigma \in \Sigma$ is an arbitrary terminal symbol and "|" separates alternatives. The second rule is an E produces a string for each type of relation and then concatenates them. The string can be the relation followed by an S, or an empty string:

$$E \rightarrow [((\gamma_1)S|\epsilon), \ldots, ((\gamma_7)S|\epsilon)], \tag{2}$$

where $\gamma_i \in \Gamma$ is the i^{th} type of relation in Γ, and $[\cdot]$ is the concatenation operator.

Fig. 3 illustrates the possible parsing procedure of an expression with the production rules. To understand those rules intuitively, regard S as an expression and E an extendable structure. Assume that an expression may contain multiple extendable structures, whereas each extendable structure can be extended to multiple expressions with spatial relations. Moreover, the production rule is associated with a probability conditioned on the input image and the context state of the parent node. Specifically, the conditional probability is defined as

$$p(\alpha \rightarrow \beta|c(\alpha), X) = D_{\alpha\rightarrow\beta}(c(\alpha), E(X)), \tag{3}$$

where X is the input image, $E(X)$ is the output of encoder, $c(\alpha)$ is the context state of α (will be detailed in Sec. 3.2) and $D_{\alpha\rightarrow\beta}(\cdot)$ is the output of decoder that corresponds to the production rule.

As illustrated in Algorithm 1, given the SAN parameters and an input image, a tree traverse is implemented with a stack. Specifically, the implemented stack can guarantee the training process according to the traversal order on a syntax tree. Similarly, the prediction process is also implemented by stacking step by step. The encoder takes an input image and down-samples it. Then based on the grammar rules, identify an expression and its extendable structures; meanwhile, the decoder calculates and selects the production rule with the highest probability. Consequently, generate new expressions with extendable structures and update the parse tree of the image in the LaTeX sequence. Once the parse tree is found, the recognition result can be acquired by traversing the tree in preorder. The remaining parts of this section focus on encoder, decoder, attention mechanism, and parameter learning.

3.1. Encoder

We use the densely connected convolutional network (DenseNet) [11] as the encoder. The DenseNet is an FCN that connects all networks in a feed-forward style and reinforces feature propagation and reuse by guaranteeing maximum information flow between layers in the network. Consequently, the FCN can handle an image of arbitrary size, making it appropriate for HMER since the sizes of handwritten images are usually in random sizes.

As a result, the encoder takes a gray-scale image X, with size of $1 \times H \times W$, where H and W are height and width respectively, and returns a $C \times H/\zeta \times W/\zeta$ matrix, where C is the channel number and ζ is the down-sampling factor. Then the encoding is represented as $E(X) = [e_1, \ldots, e_L]$, where $L = H/\zeta \times W/\zeta$ and $e_i \in \mathbb{R}^C$. Each element of $E(X)$ is related to a local region of the image. In our implementation, C is set as 684 and ζ is set as 16.

Algorithm 1 Inference process of SAN

Input: SAN parameters; The input image
Output: The parse tree of the image
Encode the image
Push S and its context state onto the top of the stack
while the stack is not empty **do**
1. Pop a non-terminal symbol or relation together with its context state from the top of the stack
2. Use the decoder to calculate the conditional probabilities of the production rules
3. Select the production rule with the highest conditional probability
4. Push each newly produced non-terminal symbol or relation in the rule with the context state onto the top of the stack
5. Update the parse tree with the selected production rule
end while
return the parse tree

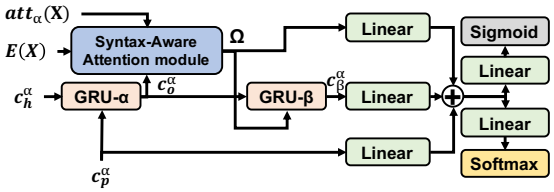

Figure 4. Syntax-Aware Decoder: Consisting of GRU-α, GRU-β, and the Syntax-Aware Attention Module.

3.2. Syntax-Aware Decoder

As illustrated in Fig. 4, the decoder mainly consists of two Gated Recurrent Units (GRU) cells [7] (indicated by GRU-α, GRU-β) and a syntax-aware attention module. Given the context state of a non-terminal symbol α and the encoding vector $E(X)$, the decoder returns the probability of each production rule that begins with α.

We use a historical state and a partner state to describe the context state of the current parsing non-terminal symbol. The historical state c_h^α is used to keep track of the information of how the non-terminal symbol α is produced. In addition, the word embedding of the latest generated terminal symbol or relation is used as the partner state c_p^α of the non-terminal symbol to capture the short-term contextual information. The first GRU-α takes c_p^α as the input vector and c_h^α as the hidden vector, and outputs a new hid-

den vector c_o^α:

$$c_o^\alpha = \mathrm{GRU}(c_p^\alpha, c_h^\alpha). \tag{4}$$

Then the attention module calculates a compact visual feature

$$\Omega = \mathrm{Att}(E(X), c_o^\alpha, att_\alpha(X)), \tag{5}$$

where $\mathrm{Att}(\cdot)$ is the attention function and $att_\alpha(X)$ is the syntax-aware attention vector which is detailed in Sec 3.3. The second GRU-β takes Ω as the input vector and c_o^α as the hidden vector, and outputs a new hidden vector c_β^α:

$$c_\beta^\alpha = \mathrm{GRU}(\Omega, c_o^\alpha). \tag{6}$$

We then aggregate c_p^α, c_β^α and Ω to predict two branches of probabilities:

$$\begin{aligned} &p_{symbol}(\alpha \to \beta | c(\alpha), X) = \\ &\quad \mathrm{softmax}(W_s(W_p c_p^\alpha + W_g c_\beta^\alpha + W_t \Omega)) \end{aligned} \tag{7}$$

$$\begin{aligned} &p_{relation}(\alpha \to \beta | c(\alpha), X) = \\ &\quad \mathrm{sigmoid}(W_r(W_p c_p^\alpha + W_g c_\beta^\alpha + W_t \Omega)) \end{aligned} \tag{8}$$

where W_s, W_p, W_g, W_t and W_r are learnable parameters. $p_{symbol}(\alpha \to \beta | c(\alpha), X)$ is a probability vector with $|\Sigma|+2$ dimensions. Note that there are three scenarios 1) $|\Sigma|$ dimensions for predicting terminal symbols, 2) one dimension for predicting E, and 3) one dimension for predicting an empty string. For the first scenario, if the prediction of a terminal symbol (σ) has the highest probability, then apply the rule of $S \to \sigma S$ to update the parse tree, use the word embedding of σ as the partner state of the newly generated S, and c_β^α as its historical state. For the second scenario, if the prediction of E has the highest probability, then apply $p_{relation}(\alpha \to \beta | c(\alpha), X)$ to predict the probability of each relation. The relations with probabilities higher than 0.5 are sustained, and others are thrown away. For each remaining relation, we use the embedding of the relationship as the partner state of the following S, and use c_β^α as the historical state. Moreover, there is no need to consider parsing an E, because it has already been acquired from the relation branch. For the third scenario, if the empty string has the highest probability, we update the parse tree with the rule of $S \to \epsilon$.

3.3. Syntax-Aware Attention Module

Instead of using the entire image feature for decoding, the attention module calculates a compact visual feature with the attention mechanism. We first compute a normalized weight for each local region of the image, and then use the weighted average to aggregate the local features. We use the image encoding $E(X)$, the hidden state c_o^α and a syntax-aware attention vector $att_\alpha(X)$ to calculate the normalized weight vector:

$$\begin{aligned} \xi_\alpha = \mathrm{softmax}(W_w(\tanh(W_o c_o^\alpha + \\ W_\alpha att_\alpha(X) + W_e E(x)))), \end{aligned} \tag{9}$$

where W_w, W_o, W_α and W_e are learnable parameters, and ξ_α is a vector of length L. The compact visual feature

$$\mathrm{Att}(E(X), c_o^\alpha, att_\alpha(X)) = E(X)\xi_\alpha \tag{10}$$

is computed by the matrix product. Unlike [42, 43], which used a coverage vector based on the sum of all past attention probabilities, SAN does not keep track of all past attention probabilities. The attention drift problem happens because the attention probabilities of the numerator give no information for parsing the denominator but appear as the noise. Instead, we sum up the past attention probabilities along the path from the root of the parse tree to the current parsing node but not all past attention probabilities. Therefore, we calculate the syntax-aware attention vector as follows:

$$att_\alpha(X) = \sum_i \xi_i, \quad i \in path_\alpha. \tag{11}$$

The syntax-aware attention vector can be efficiently traced by storing it as intermediate information with the stack.

An attention self-regularization strategy is used to correct the attention. We use an additional reversed decoder to predict the parent node of each given child node. The reversed decoder shares the same structure with the original one but operates the data in reverse. Thus we have two normalized weight vectors for predicting each non-terminal node β, the forward ξ_α and the reversed $\hat{\xi}_\eta$, where α is the parent of β and β is the parent of η. We use a Kullback-Leibler (KL) divergence to regularize them

$$\mathcal{L}_{reg} = -\sum_\beta \hat{\xi}_\eta \log \frac{\hat{\xi}_\eta}{\xi_\alpha}. \tag{12}$$

The reversed decoder is jointly trained with SAN but is omitted during inference to skip additional inference time.

3.4. Parameter Learning

SAN is trained end-to-end under a multi-task setting, that aims to minimize the sum of symbol loss ($\mathcal{L}_{symbol}$), relation loss ($\mathcal{L}_{relation}$), reversed symbol loss ($\mathcal{L}_{symbol}^{rev}$) and attention self-regularization loss ($\mathcal{L}_{reg}$).

We use the teacher forcing strategy for accelerating convergence. The ground-truth parse tree is obtained for each training image by parsing the LaTeX sequence with the depth-first-search algorithm. Thus, a list of parent-children samples is acquired from the parse tree. Afterward, deal with each sample sequentially according to the preorder of the tree until the entire tree is processed. Likewise, we use the reversed decoder to predict the parent node of each given child node. Then using the ground-truth labels, we calculate the symbol, reversed symbol and the relation losses. The attention self-regularization loss can be calculated by

Table 1. Statistical comparison of the HME100K and CROHME2019 dataset. "Max Length" and "Avg Length" mean the maximum length and average length of the mathematical expression.

Dataset	Train Size	Test Size	Symbol Class No.	Max Length	Avg Length	Writer No.
HME100K	74,502	24,607	245	184	17.62	∼10K
CROHME2019	8,836	1,199	101	96	15.79	∼100

(12). Thus we can optimize the parameters by minimizing the following objective function with backpropgation

$$\mathcal{L} = \mathcal{L}_{symbol} + \mathcal{L}_{relation} + \mathcal{L}^{rev}_{symbol} + \mathcal{L}_{reg}. \quad (13)$$

4. The HME100K Dataset

This section presents the new dataset HME100K, consisting of 74,502 images for training and 24,607 images for testing with 245 symbol classes, as shown in Table 1. Compared to CRHOME 2019 [19] dataset, the data size is increased tenfold. The data were collected from tens of thousands of writers who wrote the MEs on papers and uploaded them to an internet application.

As tens of thousands of writers participate in writing MEs, the variety of writing styles creates unique features to our MEs dataset. Consequently, our dataset HME100K becomes more authentic and realistic with variations in color, blur, complicated background, twist, illumination, longer length, and complicated structure. Samples images from the dataset are shown in Fig. 5(b-h). Furthermore, the maximum character length in HME100K is 184, which is almost twice longer than CROHME. HME100K also has a longer average sequence length than CROHME. For additional information about the HME100K dataset, please refer to the supplementary materials.

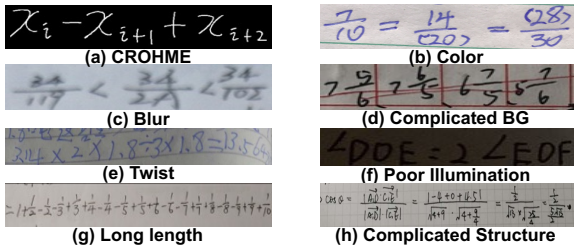

Figure 5. Sample images from (a) the CROHME dataset and (b-h) the HME100K dataset.

5. Experiments

We evaluate our method performance on two CROHME benchmark datasets and the proposed HME100K and make the comparison with the state-of-the-art methods.

5.1. Datasets

We use the Competition on Recognition of Online Handwritten Mathematical Expressions (CROHME) benchmark, which currently is the most widely used public dataset for handwritten mathematical expression to train and validate the HMER model. We convert the handwritten stroke trajectory information in the InkML files into image format for the training and test sets. Sample of CROHME images is shown in (Fig. 5(a)).

The CROHME training set consists of 8,836 expressions, including 101 math symbol classes, while the test set number of expressions are different according to the year they were published. We evaluate our model on CROHME 2014 test set [20], CROHME 2016 test set [21] and CROHME 2019 test set [19] contain 986, 1147 and 1199 expressions respectively.

5.2. Implementation Details

The proposed model SAN is implemented in PyTorch. All experiments are conducted on a single Nvidia Tesla V100 with 32GB RAM, and the batch size is set at 8. The hidden state sizes of the two GRUs are set at 256. The dimensions of word embedding and relation embedding are set at 256. The Adadelta optimizer [38] is used during the training process, in which ρ is set at 0.95 and ϵ is set at 10^{-6}. The learning rate starts from 0 and monotonously increases to 1 at the end of the first epoch. After that the learning rate decays to 0 following the cosine schedules [45]. Following most previous works, no data augmentation is applied during training for the fair comparisons.

5.3. Evaluation Protocol

Recognition Protocol. Expression recognition rate (ExpRate) is the widely used recognition protocol for mathematical expression recognition, defined as the percentage of predicted mathematical expressions accurately matching the ground truth. ExpRate ≤ 1 and ≤ 2 indicate the expression recognition rates are tolerable at most one or two symbol-level errors.

Structure Recognition Protocol. Expression Structure Prediction Rate (ESPR) is used as the structure recognition protocol. ESPR is calculated by the percent of MEs whose structure is recognized correctly irrespective of symbol labels.

5.4. Comparisons with State-of-the-arts

In this subsection, we measure our proposed method on CROHME 2014, CROHME 2016 and CROHME 2019 datasets and compare the performance with other state-of-

Table 2. Expression Recognition Rate (ExpRate) and Expression Structure Prediction Rate (ESPR) performance of our model and other state-of-the-art methods on CROHME 2014 and CROHME 2016. All results are reported as a percentage (%). Our model achieves the best performance on all public datasets. ⋆ indicates the methods that used data augmentation during the training process.

Method	CROHME 2014			CROHME 2016			CROHME 2019		
	ExpRate	≤1	≤2	ExpRate	≤1	≤2	ExpRate	≤1	≤2
WYGIWYS [8]	36.4	-	-	-	-	-	-	-	-
WAP [43]	40.4	56.1	59.9	44.6	57.1	61.6	-	-	-
PAL [32]	39.7	56.8	65.1	-	-	-	-	-	-
PAL-v2 [33]	48.9	64.5	69.8	49.6	64.1	70.3	-	-	-
PGS [16]	48.8	66.1	73.9	36.3	-	-	-	-	-
TAP [40]	48.5	63.3	67.3	44.8	59.7	62.8	-	-	-
DLA [15]	49.9	-	-	47.3	-	-	-	-	-
DWAP [39]	50.1	-	-	47.5	-	-	-	-	-
DWAP-MSA [39]	52.8	68.1	72.0	50.1	63.8	67.4	47.7	59.5	63.3
DWAP-TD [42]	49.1	64.2	67.8	48.5	62.3	65.3	51.4	66.1	69.1
WS-WAP [27]	53.7	-	-	52.0	64.3	70.1	-	-	-
MAN [29]	54.1	68.8	72.2	50.6	64.8	67.1	-	-	-
RBR [27]	53.4	65.2	70.3	52.1	63.2	69.4	53.1	63.9	68.5
DWAP + CTC [23]	51.0	-	-	50.0	-	-	-	-	-
BTTR [46]	54.0	66.0	70.3	52.3	63.9	68.6	53.0	66.0	69.1
Li *et al.* [18]⋆	56.6	69.1	75.3	54.6	69.3	73.8	-	-	-
Ding *et al.* [9]⋆	58.7	-	-	57.7	70.0	76.4	61.4	75.2	80.2
SAN	**56.2**	**72.6**	**79.2**	**53.6**	**69.6**	**76.8**	**53.5**	**69.3**	**70.1**
SAN⋆	**63.1**	**75.8**	**82.0**	**61.5**	**73.3**	**81.4**	**62.1**	**74.5**	**81.0**

the-art methods. Most of the previous methods do not use data augmentation, so we mainly focus on the results produced without data augmentation. Note that our method is not with the beam search process to obtain additional performance improvements.

Evaluation on CROHME dataset As shown in table 2, our method achieves state-of-the-art performance on all CROHME datasets. SAN outperforms MAN [29] by 3.1%, BTTR [46] by 1.3% and BTTR [46] by 0.5% on CRHOME 2014, 2016 and 2019, respectively. In addition to ExpRate, SAN outperforms other state-of-the-art methods by a larger margin in ESPR. Thus, the SAN achievement demonstrates that incorporating syntax information into HMER neural network is effective and efficient.

Table 3. Performance of our model versus DWAP, DWAP-TD and BTTR on Easy (E.), Moderate (M.) and Hard (H.) HME100K test subsets. Inference speed reported as FPS means frames per second. The last column shows the parameter numbers of each model. The number in bold font corresponds to the best performance, and the second-best result is shown with an underline. Our model achieves the best performance on the HME100K dataset.

HME100K	E.	M.	H.	Total	FPS	P.N.
Image size	7721	10450	6436	24607	-	-
DWAP [39]	75.1	62.2	45.4	61.9	<u>23.3</u>	4.8M
DWAP-TD [42]	76.2	63.2	45.4	62.6	6.9	8.0M
BTTR [46]	<u>77.6</u>	<u>65.3</u>	<u>46.0</u>	<u>64.1</u>	3.9	6.5M
SAN	**79.2**	**67.6**	**51.5**	**67.1**	**23.9**	8.9M

5.5. Evaluation on HME100K

Dataset Division. For mathematical expressions, the structural complexity (S.C.) [44] and character length (C.L.) significantly affect the model performance. With this in mind, we carefully allocate our test dataset into three subsets by difficulty. The allocation criteria is as follow:

$$\begin{cases} \textbf{Easy}, & S.C. \in [0,1] \ \& \ C.L. \in [1,10) \\ \textbf{Moderate}, & S.C. \in [0,1] \ \& \ C.L. \in [10,20) \\ \textbf{Hard}, & otherwise \end{cases}$$

Comparisons with previous methods. In this subsection, we compare our proposed method with DWAP [39], DWAP-TD [42] and BTTR [46] on HME100K dataset. In order to be consistent with the reported inference process, DWAP-TD [42] and BTTR [46] use beam search, while DWAP [39] doesn't. Specifically, as shown in table 3, our method outperforms BTTR [46] by 1.6% on easy subset. However, as the difficulty of the test subset increases, the leading margin of our method increases to 5.5% on the hard subset. The measurements of performances on different test subsets prove the superior robustness and structural awareness of our method.

In addition to the recognition accuracy, we evaluate the inference speed of our model against DWAP [39] and DWAP-TD [42] as summarised in table 3. We measure the inference speeds in frame per second (FPS) on an HME100K test set via an Nvidia Tesla V100. Impressively, SAN is 3.5 times faster than DWAP-TD [42], while 2.6% faster than DWAP [39], which indicates the efficiency of our method in minimizing computational costs.

5.6. Ablation Study

In this subsection, we perform ablation studies to analyze the impact of grammar syntax and syntax-aware attention modules. We trained all models from scratch and evaluated their performance on three datasets. SAN is the default model, while SAN-GS has the syntax-aware decoder but

Table 4. Ablation studies on CROHME datasets and HME100K datasets. The effect of recognition performance with regard to the two basic parameter settings: grammar syntax (G.S.) and syntax-aware attention (S.A. Attention). (✓) denotes the module existence while (✗) indicates the module absence.

Model	G.S.	SA. Attention	ExpRate	Dataset
Baseline	✗	✗	49.1	CROHME 2014
SAN_GS	✓	✗	55.3	
SAN	✓	✓	56.2	
Baseline	✗	✗	48.5	CROHME 2016
SAN_GS	✓	✗	52.8	
SAN	✓	✓	53.6	
Baseline	✗	✗	51.4	CROHME 2019
SAN_GS	✓	✗	53.0	
SAN	✓	✓	53.5	
Baseline	✗	✗	62.6	HME100K
SAN_GS	✓	✗	66.5	
SAN	✓	✓	67.1	

adopts coverage attention in the attention module instead of syntax-aware attention. The results are summarized in Table 4.

Impact of Grammar Syntaxes. Table 4 shows the integration of grammar syntaxes to baseline improves the recognition performance ExpRate by 6.2% on CROHME 2014, 4.3% on CROHME 2016, 1.6% on CROHME 2019, and 3.9% on HME100K. Hence integrating grammatical constraints into the baseline model achieved improvements on all datasets, and this proofs the significance of grammar syntaxes incorporation to the decoder.

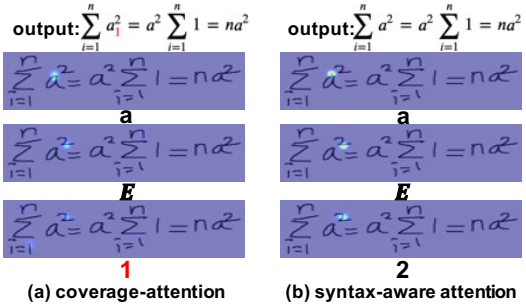

Figure 6. Examples of (a) Converge-Attention and (b) Syntax-Aware Attention. The current recognized symbol is printed below each image. "E" indicates the non-terminal symbol (Eq.(1)).

Impact of Syntax-Aware Attention. As it is illustrated in Table 4, incorporating syntax-aware attention on top of grammar syntaxes increases the recognition performance rate ExpRate by 7.1% on CROHME 2014, 5.1% on CROHME 2016, 2.1% on CROHME 2019, and 4.5% on HME100K from the baseline.

Furthermore, as shown in Fig. 6, we compare the coverage attention and syntax-aware attention through a qualitative example. The images extracted from the prediction steps of coverage-attention and syntax-aware attention models illustrate how each model focuses on the target at each step. As shown in (Fig. 6(a)), the coverage attention model wrongly focuses on the region of the symbol "1", predicting a redundant sub-tree that doesn't exist in the predicted position. In contrast, the syntax-aware attentions model predicts an appropriate location that coincides with human intuitions (Fig. 6(b)).

6. Limitation

As it is illustrated in Fig. 7, distorted and sticking components of ME can cause failure to SAN prediction, which leads to the under/over translation. Fig. 7 gives four typical examples on CROHME (a and b) and HME100K (c and d) dataset.

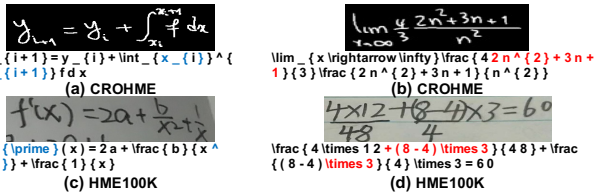

Figure 7. Limitation sample examples with the ground truth and recognition results of SAN. Characters in red color are mispredictions.

7. Conclusions

This paper has presented an unconventional method for handwritten mathematical expression recognition by combining syntax information and visual representations to make robust predictions. To our best knowledge, the proposed syntax-aware network is the first to effectively incorporate the grammar rules into deep feature learning. Our method not only predicts LaTeX markup results but also directly produces the tree structure output that can precisely describe the component relationship of mathematical expressions. Experiments on the benchmark datasets and the proposed HME100K dataset have validated the effectiveness and efficiency of our method. In our future work, we are interested in extending the proposed method to recognize other complicated structure objects.

8. Acknowledgement

This work was supported by National Key R&D Program of China, under Grant No. 2020AAA0104500 and National Natural Science Foundation of China 61733007.

References

[1] Francisco Alvaro, Joan-Andreu Sánchez, and José-Miguel Benedí. Recognition of on-line handwritten mathematical expressions using 2d stochastic context-free grammars and

hidden markov models. *Pattern Recognition Letters*, 35:58–67, 2014. 1, 2

[2] Francisco Álvaro, Joan-Andreu Sánchez, and José-Miguel Benedí. An integrated grammar-based approach for mathematical expression recognition. *Pattern Recognition*, 51:135–147, 2016. 2

[3] Ahmad-Montaser Awal, Harold Mouchere, and Christian Viard-Gaudin. A global learning approach for an online handwritten mathematical expression recognition system. *Pattern Recognition Letters*, 35:68–77, 2014. 2

[4] Kam-Fai Chan and Dit-Yan Yeung. Elastic structural matching for online handwritten alphanumeric character recognition. In *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No. 98EX170)*, volume 2, pages 1508–1511. IEEE, 1998. 2

[5] Kam-Fai Chan and Dit-Yan Yeung. Mathematical expression recognition: a survey. *International Journal on Document Analysis and Recognition*, 3(1):3–15, 2000. 2

[6] Kam-Fai Chan and Dit-Yan Yeung. Error detection, error correction and performance evaluation in on-line mathematical expression recognition. *Pattern Recognition*, 34(8):1671–1684, 2001. 1

[7] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *Advances in Neural Information Processing Systems Workshop*, 2014. 4

[8] Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M Rush. Image-to-markup generation with coarse-to-fine attention. In *International Conference on Machine Learning*, pages 980–989. PMLR, 2017. 1, 2, 7

[9] Haisong Ding, Kai Chen, and Qiang Huo. An encoder-decoder approach to handwritten mathematical expression recognition with multi-head attention and stacked decoder. In Josep Lladós, Daniel Lopresti, and Seiichi Uchida, editors, *Document Analysis and Recognition – ICDAR 2021*, pages 602–616, Cham, 2021. Springer International Publishing. 7

[10] Lei Hu and Richard Zanibbi. Hmm-based recognition of online handwritten mathematical symbols using segmental k-means initialization and a modified pen-up/down feature. In *2011 International Conference on Document Analysis and Recognition*, pages 457–462. IEEE, 2011. 2

[11] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 4

[12] Birendra Keshari and S Watt. Hybrid mathematical symbol recognition using support vector machines. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 859–863. IEEE, 2007. 2

[13] Andreas Kosmala, Gerhard Rigoll, Stéphane Lavirotte, and Loic Pottier. On-line handwritten formula recognition using hidden markov models and context dependent graph grammars. In *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR'99 (Cat. No. PR00318)*, pages 107–110. IEEE, 1999. 2

[14] Stéphane Lavirotte and Loic Pottier. Mathematical formula recognition using graph grammar. In *Document Recognition V*, volume 3305, pages 44–52. International Society for Optics and Photonics, 1998. 1

[15] Anh Duc 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*, pages 566–567, 2020. 2, 7

[16] Anh Duc Le, Bipin Indurkhya, and Masaki Nakagawa. Pattern generation strategies for improving recognition of handwritten mathematical expressions. *Pattern Recognition Letters*, 128:255–262, 2019. 2, 7

[17] Anh Duc Le and Masaki Nakagawa. A system for recognizing online handwritten mathematical expressions by using improved structural analysis. *International Journal on Document Analysis and Recognition (IJDAR)*, 19(4):305–319, 2016. 2

[18] Zhe Li, Lianwen Jin, Songxuan Lai, and Yecheng 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)*, pages 175–180. IEEE, 2020. 7

[19] Mahshad Mahdavi, Richard Zanibbi, Harold Mouchère, Christian Viard-Gaudin, and Utpal 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)*, pages 1533–1538. IEEE, 2019. 2, 6

[20] Harold Mouchere, Christian Viard-Gaudin, Richard Zanibbi, and Utpal Garain. Icfhr 2014 competition on recognition of on-line handwritten mathematical expressions (crohme 2014). In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 791–796. IEEE, 2014. 2, 6

[21] Harold Mouchère, Christian Viard-Gaudin, Richard Zanibbi, and Utpal Garain. Icfhr2016 crohme: Competition on recognition of online handwritten mathematical expressions. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 607–612. IEEE, 2016. 2, 6

[22] Harold Mouchere, Richard Zanibbi, Utpal Garain, and Christian Viard-Gaudin. Advancing the state of the art for handwritten math recognition: the crohme competitions, 2011–2014. *International Journal on Document Analysis and Recognition (IJDAR)*, 19(2):173–189, 2016. 2

[23] Cuong Tuan Nguyen, Hung Tuan Nguyen, Kei Morizumi, and Masaki Nakagawa. Temporal classification constraint for improving handwritten mathematical expression recognition. In *International Conference on Document Analysis and Recognition*, pages 113–125. Springer, 2021. 2, 7

[24] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(11):2298–2304, 2017. 1

[25] Baoguang Shi, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. Robust scene text recognition with automatic rectification. In *Proceedings of the IEEE conference on*

computer vision and pattern recognition, pages 4168–4176, 2016. 1

[26] Baoguang Shi, Mingkun Yang, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. Aster: An attentional scene text recognizer with flexible rectification. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2035–2048, 2018. 1

[27] Thanh-Nghia Truong, Cuong Tuan Nguyen, Khanh Minh Phan, and Masaki 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)*, pages 181–186. IEEE, 2020. 2, 7

[28] Ba-Quy Vuong, Yulan He, and Siu Cheung Hui. Towards a web-based progressive handwriting recognition environment for mathematical problem solving. *Expert Systems with Applications*, 37(1):886–893, 2010. 2

[29] Jiaming Wang, Jun Du, Jianshu Zhang, and Zi-Rui Wang. Multi-modal attention network for handwritten mathematical expression recognition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1181–1186. IEEE, 2019. 2, 7

[30] H-J Winkler. Hmm-based handwritten symbol recognition using on-line and off-line features. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 6, pages 3438–3441. IEEE, 1996. 2

[31] Jin-Wen Wu, Fei Yin, Y Zhang, Xu-Yao Zhang, and Cheng-Lin Liu. Graph-to-graph: towards accurate and interpretable online handwritten mathematical expression recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2925–2933, 2021. 3

[32] Jin-Wen Wu, Fei Yin, Yan-Ming Zhang, Xu-Yao Zhang, and Cheng-Lin Liu. Image-to-markup generation via paired adversarial learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 18–34. Springer, 2018. 7

[33] Jin-Wen Wu, Fei Yin, Yan-Ming Zhang, Xu-Yao Zhang, and Cheng-Lin Liu. Handwritten mathematical expression recognition via paired adversarial learning. *International Journal of Computer Vision*, pages 1–16, 2020. 7

[34] Ryo Yamamoto, Shinji Sako, Takuya Nishimoto, and Shigeki Sagayama. On-line recognition of handwritten mathematical expressions based on stroke-based stochastic context-free grammar. In *Tenth international workshop on frontiers in handwriting recognition*. Suvisoft, 2006. 1

[35] Mingkun Yang, Yushuo Guan, Minghui Liao, Xin He, Kaigui Bian, Song Bai, Cong Yao, and Xiang Bai. Symmetry-constrained rectification network for scene text recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9147–9156, 2019. 1

[36] Richard Zanibbi and Dorothea Blostein. Recognition and retrieval of mathematical expressions. *International Journal on Document Analysis and Recognition (IJDAR)*, 15(4):331–357, 2012. 2

[37] Richard Zanibbi, Dorothea Blostein, and James R. Cordy. Recognizing mathematical expressions using tree transformation. *IEEE Transactions on pattern analysis and machine intelligence*, 24(11):1455–1467, 2002. 2

[38] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. 6

[39] Jianshu Zhang, Jun Du, and Lirong Dai. Multi-scale attention with dense encoder for handwritten mathematical expression recognition. In *2018 24th international conference on pattern recognition (ICPR)*, pages 2245–2250. IEEE, 2018. 1, 2, 7

[40] Jianshu Zhang, Jun Du, and Lirong Dai. Track, attend, and parse (tap): An end-to-end framework for online handwritten mathematical expression recognition. *IEEE Transactions on Multimedia*, 21(1):221–233, 2018. 2, 7

[41] Jianshu Zhang, Jun Du, Yongxin Yang, Yi-Zhe Song, and Lirong Dai. Srd: A tree structure based decoder for online handwritten mathematical expression recognition. *IEEE Transactions on Multimedia*, 2020. 3

[42] Jianshu Zhang, Jun Du, Yongxin Yang, Yi-Zhe Song, Si Wei, and Lirong Dai. A tree-structured decoder for image-to-markup generation. In *International Conference on Machine Learning*, pages 11076–11085. PMLR, 2020. 1, 3, 5, 7

[43] Jianshu Zhang, Jun Du, Shiliang Zhang, Dan Liu, Yulong Hu, Jinshui Hu, Si Wei, and Lirong Dai. Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition. *Pattern Recognition*, 71:196–206, 2017. 1, 2, 5, 7

[44] Ting Zhang, Harold Mouchere, and Christian Viard-Gaudin. Tree-based blstm for mathematical expression recognition. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 914–919. IEEE, 2017. 3, 7

[45] Zhi Zhang, Tong He, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of freebies for training object detection neural networks. *arXiv preprint arXiv:1902.04103*, 2019. 6

[46] Wenqi Zhao, Liangcai Gao, Zuoyu Yan, Shuai Peng, Lin Du, and Ziyin Zhang. Handwritten mathematical expression recognition with bidirectionally trained transformer. In *International Conference on Document Analysis and Recognition*, pages 570–584. Springer, 2021. 2, 7