TDv2: A Novel Tree-Structured Decoder for Offline Mathematical Expression Recognition

Changjie Wu[1], Jun Du[1*], Yunqing Li[1], Jianshu Zhang[2], Chen Yang[1], Bo Ren[3], Yiqing Hu[3]

[1] NEL-SLIP Lab, University of Science and Technology of China
[2] iFLYTEK Research
[3] Tencent Youtu

wucj@mail.ustc.edu.cn, jundu@ustc.edu.cn, lyq123@mail.ustc.edu.cn, jszhang6@iflytek.com, yc24@mail.ustc.edu.cn, timren@tencent.com, hooverhu@tencent.com

Abstract

In recent years, tree decoders become more popular than LaTeX string decoders in the field of handwritten mathematical expression recognition (HMER) as they can capture the hierarchical tree structure of mathematical expressions. However previous tree decoders converted the tree structure labels into a fixed and ordered sequence, which could not make full use of the diversified expression of tree labels. In this study, we propose a novel tree decoder (TDv2) to fully utilize the tree structure labels. Compared with previous tree decoders, this new model does not require a fixed priority for different branches of a node during training and inference, which can effectively improve the model generalization capability. The input and output of the model make full use of the tree structure label, so that there is no need to find the parent node in the decoding process, which simplifies the decoding process and adds a priori information to help predict the node. We verified the effectiveness of each part of the model through comprehensive ablation experiments and attention visualization analysis. On the authoritative CROHME 14/16/19 datasets, our method achieves the state-of-the-art results.

Introduction

Handwritten mathematical expression recognition (HMER) has attracted more and more attention in academia and industry due to its huge application value. Different from text line recognition (Shi, Bai, and Yao 2016; Cheng et al. 2017; Li et al. 2019), handwritten mathematical expressions usually have complex structures, which means that HMER not only needs to accurately recognize each handwritten symbol in various forms, but also needs to analyze the corresponding relationship between the symbols. This complex spatial structure not only makes the process more complicated, but also puts forward higher requirements on the generalization of HMER algorithms.

Many recent handwritten mathematical expression recognition works use end-to-end algorithms, which can perform symbol recognition and structural analysis in a single pipeline. These works are mainly based on the encoder-decoder framework to convert pictures or a series of trajectory points into LaTeX strings (Deng et al. 2017; Zhang et al. 2017a; Zhang, Du, and Dai 2019). They do not independently model the spatial relationship of mathematical expressions, so the generalization of the model is often poor. When recognizing mathematical expressions with more complex structures, the performance of these methods is unsatisfactory (Zhang et al. 2020).

Besides, tree-structured decoder (TD) (Zhang et al. 2020) describes mathematical expressions with a tree structure label and uses different modules to separately predict the node category and the spatial relationship between nodes. This separate modeling of spatial relationships can effectively improve the generalization of the model to complex structures. But unfortunately, in the process of training, they converted the tree structure label into a fixed and ordered sequence, which could not make full use of the diversified expression brought by the tree structure label. On the other hand, in the decoding process of this model, a complex attention memory module is needed to find the parent node of the current node, which makes the decoding process more complicated and inefficient.

In this article, we propose a novel tree decoder (TDv2) that can fully utilize the tree structure tags of mathematical expressions to obtain more generalization and higher accuracy. The overall architecture of TDv2 is based on the encoder-decoder model, which is shown in Figure 1. The encoder uses a convolutional neural network (CNN) to extract high-dimensional features of offline mathematical expression pictures. The decoder includes a node classification module and a branch prediction module to predict the symbol category of the current node and branches of the current node separately. After getting the node information, we put it into the stack table to build a tree of mathematical expression. In the inference stage, the stack table is also responsible for automatically popping the parent node and the spatial relationship between the current node and the parent node as the input of the decoder. For HMER, we also propose two novel and effective improvements: adding "thinking" labels and pixel-level auxiliary classification loss. We add a "thinking" label to let the attention mechanism observe first, and make better preparations for finding child nodes, so that the attention mechanism can focus on the child nodes more accurately. In addition, in order to better optimize the encoder to obtain a stronger ability of extracting features, we directly

*Corresponding author.

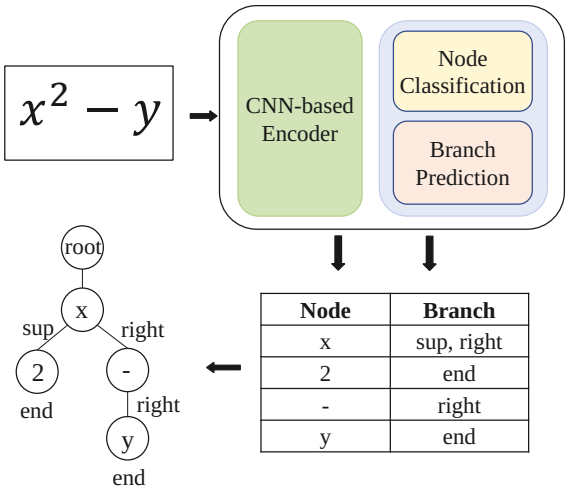

Figure 1: Overal architecture of TDv2.

classify each position feature vector in the feature map obtained by the encoder, and then calculate the probability of the current child node category through the attention mechanism.

The main contributions of this article are as follows:

- Through the tree structure label of the mathematical expression, we propose a novel tree decoder (TDv2) with only a node classification module and a branch prediction module to simplify the decoding process and improve the generalization.
- For HMER, we also present two novel improvements: adding "thinking" labels and pixel-level auxiliary classification loss, which effectively boost the performance of TDv2.
- We verify the effectiveness of each part of the proposed method through ablation experiments and attention visualization analysis, and achieve very competitive results on the CROHME 14/16/19 datasets.
- The source codes of our proposed TDv2 are available at https://github.com/yqingli123/TDv2.git.

Related Work

Traditional Methods

Symbol recognition and structural analysis are two major problems for mathematical expression recognition, while can be solved sequentially or globally. Sequential approaches (Zanibbi, Blostein, and Cordy 2002; Álvaro, Sánchez, and Benedí 2014) first segmented the whole mathematical expression into several instances and then recognized them into math symbols. The structural analysis was performed to find the most likely math tree based on symbol segmentation and recognition results. In contrast, global approaches (Awal, Mouchère, and Viard-Gaudin 2014; Álvaro, Sánchez, and Benedí 2016) optimized symbol recognition and structural analysis simultaneously, which can utilize the global contextual information from symbol recognition and structural analysis.

Latex Modeling

In LaTeX modeling methods, the input can be static image or dynamic traces and the output is LaTex sequence, corresponding to offline recognition and online recognition respectively. Both can be modeling based on encoder-decoder framework, which has been applied to many sequence-to-sequence tasks (Cho et al. 2014; Chan et al. 2016; Cho, Courville, and Bengio 2015; Huang et al. 2016).

For offline methods, the encoder is usually based on VGG (Simonyan and Zisserman 2014) or DenseNet(Huang et al. 2017) for feature extraction. WYGIWYS (Deng et al. 2017) was proposed with coarse-to-fine attention mechanism for machine-printed mathematical expression recognition, while Zhang (Zhang et al. 2017b; Zhang, Du, and Dai 2018a) proposed more robust model structures which perform better in HMER. Besides, adversarial learning strategy was (Wu et al. 2018, 2020; Le 2020) adopted to help learning more semantic invariant feature. For online methods, some works (Le and Nakagawa 2017; Zhang, Du, and Dai 2018b) utilize RNN-based encoder to extract feature from dynamic formula strokes. Hong (Hong et al. 2019) employed residual connection in BiRNN to improve feature extraction. Besides, multi-modal learning was also introduced into HMER with both online and offline information for better encoding (Wang et al. 2019) and decoding (Wang et al. 2021). There are also approaches which apply data augmentation (Le and Nakagawa 2017; Le, Indurkhya, and Nakagawa 2019; Li et al. 2020).

Tree Structure Modeling

There are some studies using neural network to encode or decode a tree in various tasks, such as machine translation (Eriguchi, Hashimoto, and Tsuruoka 2016) and semantic parsing (Dong and Lapata 2016). Generally, the research of tree structure encoder (Tai, Socher, and Manning 2015; Zhu, Sobhani, and Guo 2015; Socher et al. 2011) has achieved good performance. However, the design of the tree-structured decoder is more difficult and complicated. Some models (Rabinovich, Stern, and Klein 2017; Parisotto et al. 2017) need to rely on specific prior knowledge of specific tasks, thus they are task-specific tree decoder and difficult to generalize to other tasks. Meanwhile, by utilizing the structural knowledge, general tree decoders have also been investigated in several studies (Chen, Liu, and Song 2018; Chakraborty, Allamanis, and Ray 2018). Recently, Zhang (Zhang et al. 2020) proposed a novel tree-structured decoder (TD) which produced a parent node, a child node and their relationship simultaneously at each step. It abandons the constraint of complex prior knowledge and can easily generalize to other tasks.

As described above, with the help of structured models, fully mining the structural information of formulas is the key to enhancing generalization. Graph-to-graph model (Wu et al. 2021) introduced to online HMER has also proved that. Therefore, in this paper, we fully tap the diversity of tree structure label and simplify TD for better generalization.

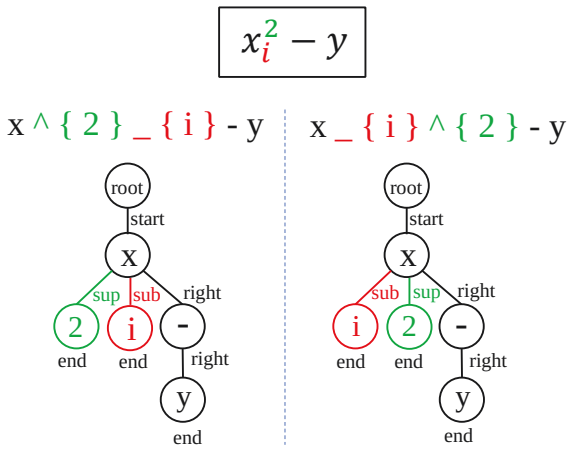

Figure 2: Multiple descriptions of a mathematical expression

Methodology

In this section, we first introduce the diverse tree structure labels of mathematical expressions in Subsection 3.1. Next, we show the main framework of the proposed TDv2. In Subsection 3.2, we briefly introduce the encoder. In Subsection 3.3, we introduce the two main modules in the decoder in detail. Finally, we introduce two novel improvements for HMER in Subsection 3.4 and Subsection 3.5.

Diverse Tree Structure Label

As shown in the Figure 2, mathematical expressions are actually data in a tree structure. Each node in the tree represents a mathematical symbol, and the path between nodes represents the spatial relationship between nodes, such as subscripts, superscripts, etc. What we need to pay attention to is that some nodes in the tree may have multiple spatial relationships (branches), but these spatial relationships are not in strict order. For example, the left and right descriptions in the Figure 2 all represent the same mathematical expression $x_i^2 - y$. In order to prevent different annotations from causing ambiguity, the previous methods need to strictly ensure that the spatial relationship of the same formula is described in a fixed order. In TDv2, the priority of spatial relations can be specified or randomly disrupted. This means that when a node has multiple branches, the model can choose any path freely, regardless of their order. We believe that this choice of out-of-order decoding paths can not only allow more patterns in the training phase, but also improve the generalization by weakening contexual dependence. In each epoch of training, we randomly shuffle the branches, and convert the shuffled tree structure label into node list and branch list recording node categories and braches separately according to the depth-first traversal principle. Taking the formula $x_i^2 - y$ as an example, if its tree structure is the left-hand situation in Figure 2, we can get the node list: [x, 2, i, -, y] and the branch list: [(sup, sub, right), (end), (end), (right), (end)]. With these two lists, we can generate the input list [(root, start), (x, sup), (x, sub), (x, right), (-, right)] of the decoder to help predict nodes. This process transfers a tree structure into ordered list for latter modeling.

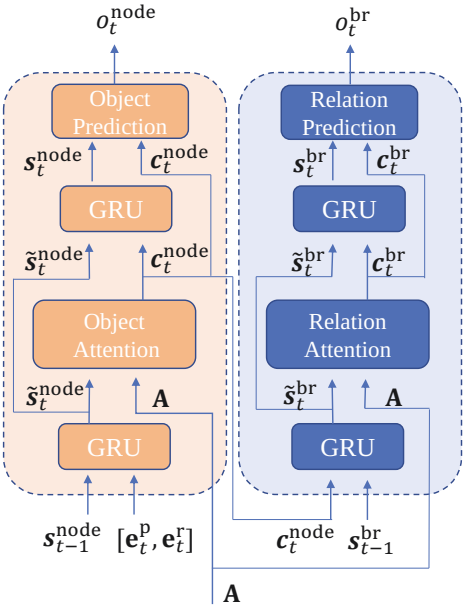

Figure 3: Illustration of tree decoder with two modules. Left part in 'orange' represents node classification module. Right part in 'blue' represents branch prediction module.

Encoder

The encoder used in this model is a densely connected network (DenseNet). Since the focus of this work is to propose a new tree decoder, some other classic convolutional neural networks are also recommended, and there will be no further discussion on the encoder. The input picture of the encoder is $\mathbf{I} \in \mathbb{R}^{H \times W \times C}$, H, W, C are the height, width and number of channels. We define the high-dimensional features of the output of the last convolutional layer as $\mathbf{A}$:

$$\mathbf{A} = \text{CNN}(\mathbf{I}), \mathbf{A} \in \mathbb{R}^{H' \times W' \times D} \quad (1)$$

$\mathbf{A}$ can be regarded as a collection of elements $\mathbf{a}_i \in \mathbb{R}^D$, where $\mathbf{a}_i$ is the feature vector corresponding to the local area of the image.

Decoder

In each decoding step of the decoder, the tree decoder needs to predict the current child node information, including the category of the child node and the branches of the child node. The branch of a node represents the spatial relationship between the node and its child nodes. We can gradually build a mathematical tree through node categories and branches. As shown in Figure 3, in order to decouple node classification and spatial relationship prediction, we designed two modules in the decoder: node classification module and branch prediction module.

Node Classification Module. As shown in Figure 3, the node classification module mainly includes two gated recurrent units (GRU) (Chung et al. 2014), an attention mecha-

nism (Zhang et al. 2017a) and a classifier. In the node classification module, we first use two embedding layers to obtain the high-dimensional feature vectors $\mathbf{e}_t^{\mathrm{p}}$ and $\mathbf{e}_t^{\mathrm{r}}$ of the parent node p_t and the spatial relationship r_t. Then, the previous hidden state of the node decoder $\mathbf{s}_{t-1}^{\mathrm{node}}$ is taken as the previous hidden state of the $\mathrm{GRU}_1^{\mathrm{node}}$ layer. The parent node embedding $\mathbf{e}_t^{\mathrm{p}}$ and the spatial relationship node embedding $\mathbf{e}_t^{\mathrm{r}}$ are stacked together as the input of $\mathrm{GRU}_1^{\mathrm{node}}$. In this way we can get the current hidden state $\widetilde{\mathbf{s}}_t^{\mathrm{node}}$ of $\mathrm{GRU}_1^{\mathrm{node}}$.

$$\mathbf{e}_t^{\mathrm{p}} = \mathrm{Emd}_{\mathrm{node}}\left(p_t\right) \tag{2}$$

$$\mathbf{e}_t^{\mathrm{r}} = \mathrm{Emd}_{\mathrm{re}}\left(r_t\right) \tag{3}$$

$$\widetilde{\mathbf{s}}_t^{\mathrm{node}} = \mathrm{GRU}_1^{\mathrm{node}}\left(\left[\mathbf{e}_t^{\mathrm{p}}, \mathbf{e}_t^{\mathrm{r}}\right], \mathbf{s}_{t-1}^{\mathrm{node}}\right) \tag{4}$$

Then, a node attention module $f_{\mathrm{att}}^{\mathrm{node}}$ is used to calculate the attention probability $\boldsymbol{\alpha}_t^{\mathrm{node}}$ on the feature map $\mathbf{A}$. The node context vector $\mathbf{c}_t^{\mathrm{node}}$ is obtained by performing weighted summation on $\mathbf{A}$. We use $\widetilde{\mathbf{s}}_t^{\mathrm{node}}$ as the query, and $\mathbf{A}$ as key and value.

$$\boldsymbol{\alpha}_t^{\mathrm{node}} = f_{\mathrm{att}}^{\mathrm{node}}\left(\mathbf{A}, \widetilde{\mathbf{s}}_t^{\mathrm{node}}\right) \tag{5}$$

$$\mathbf{c}_t^{\mathrm{node}} = \sum_i \alpha_{ti}^{\mathrm{node}} \mathbf{a}_i \tag{6}$$

The function $f_{\mathrm{att}}^{\mathrm{node}}$ is designed as follows:

$$\mathbf{F}^{\mathrm{node}} = \mathbf{Q}^{\mathrm{node}} * \sum_{l=1}^{t-1} \boldsymbol{\alpha}_l^{\mathrm{node}} \tag{7}$$

$$e_{ti}^{\mathrm{node}} = \boldsymbol{V}_{\mathrm{node}}^{\mathrm{T}} \tanh\left(\mathbf{W}_{\mathrm{att}}^{\mathrm{node}} \tilde{\mathbf{s}}_t^{\mathrm{node}} + \mathbf{U}_{\mathrm{att}}^{\mathrm{node}} \mathbf{a}_i + \hat{\mathbf{U}}_{\mathrm{F}}^{\mathrm{node}} \mathbf{f}_i^{\mathrm{node}}\right) \tag{8}$$

$$\alpha_{ti}^{\mathrm{node}} = \frac{\exp\left(e_{ti}^{\mathrm{node}}\right)}{\sum_k \exp\left(e_{tk}^{\mathrm{node}}\right)} \tag{9}$$

where $\alpha_{ti}^{\mathrm{node}}$ denotes the node attention probability of i-th element at step t, e_{ti}^{node} denotes the output energy of i-th step. $\mathbf{f}_i^{\mathrm{node}}$ denotes the i-th element of $\mathbf{F}^{\mathrm{node}}$, which represents the history attention for avoiding over-parsing or under-parsing problem. $\mathbf{Q}^{\mathrm{node}}$, $\boldsymbol{V}_{\mathrm{node}}^{\mathrm{T}}$, $\mathbf{W}_{\mathrm{att}}^{\mathrm{node}}$, $\mathbf{U}_{\mathrm{att}}^{\mathrm{node}}$, $\hat{\mathbf{U}}_{\mathrm{F}}^{\mathrm{node}}$ are all learnable parameters.

Then, we use $\mathbf{c}_t^{\mathrm{node}}$ and $\widetilde{\mathbf{s}}_t^{\mathrm{node}}$ as the input of $\mathrm{GRU}_2^{\mathrm{node}}$ to calculate the hidden state $\mathbf{s}_t^{\mathrm{node}}$ of the node prediction module:

$$\mathbf{s}_t^{\mathrm{node}} = \mathrm{GRU}_2^{\mathrm{node}}\left(\mathbf{c}_t^{\mathrm{node}}, \widetilde{\mathbf{s}}_t^{\mathrm{node}}\right) \tag{10}$$

Finally, the probability of each predicted node $\mathbf{o}_t^{\mathrm{node}}$ is computed from the concatenation of parent node $\mathbf{e}_t^{\mathrm{p}}$, parent relation $\mathbf{e}_t^{\mathrm{r}}$, node hidden state $\mathbf{s}_t^{\mathrm{node}}$ and child context vector $\mathbf{c}_t^{\mathrm{node}}$:

$$\mathbf{h}_t^{\mathrm{node}} = \mathrm{maxout}\left(\mathbf{W}_1^{\mathrm{node}}\left[\mathbf{e}_t^{\mathrm{p}}, \mathbf{e}_t^{\mathrm{r}}, \mathbf{s}_t^{\mathrm{node}}, \mathbf{c}_t^{\mathrm{node}}\right]\right) \tag{11}$$

$$\mathbf{o}_t^{\mathrm{node}} = \mathrm{softmax}\left(\mathbf{W}_2^{\mathrm{node}} \mathbf{h}_t^{\mathrm{node}}\right) \tag{12}$$

where $\mathbf{W}_1^{\mathrm{node}}$, $\mathbf{W}_2^{\mathrm{node}}$ are learnable parameters.

We use the cross-entropy function to calculate the classification loss:

$$\mathcal{L}_{\mathrm{node}} = -\sum_t \log \mathbf{o}_t^{\mathrm{node}} \cdot \mathbf{n}_t \tag{13}$$

where $\mathbf{n}_t$ denotes the one-hot vector of ground-truth node at time step t.

Branch Prediction Module. The overall structure of the branch prediction module is roughly the same as that of node classification module, except that the spatial attention mechanism is slightly different. In the branch prediction module, we first take the previous hidden state of the branch prediction module $\mathbf{s}_{t-1}^{\mathrm{br}}$ as the previous hidden state of the $\mathrm{GRU}_1^{\mathrm{br}}$ layer. The context vector $\mathbf{c}_t^{\mathrm{node}}$ of the node works as input of $\mathrm{GRU}_1^{\mathrm{br}}$, and we can get the hidden state $\widetilde{\mathbf{s}}_t^{\mathrm{br}}$ of the $\mathrm{GRU}_1^{\mathrm{br}}$ layer:

$$\widetilde{\mathbf{s}}_t^{\mathrm{br}} = \mathrm{GRU}_1^{\mathrm{br}}\left(\mathbf{c}_t^{\mathrm{node}}, \mathbf{s}_{t-1}^{\mathrm{br}}\right) \tag{14}$$

Then, a branch attention module $f_{\mathrm{att}}^{\mathrm{br}}$ is used to calculate the attention probability $\boldsymbol{\alpha}_t^{\mathrm{br}}$ on the feature map $\mathbf{A}$. The branch context vector $\mathbf{c}_t^{\mathrm{br}}$ is obtained by performing weighted summation on $\mathbf{A}$. We use $\widetilde{\mathbf{s}}_t^{\mathrm{br}}$ as the query, and the $\mathbf{A}$ as key and value.

$$\boldsymbol{\alpha}_t^{\mathrm{br}} = f_{\mathrm{att}}^{\mathrm{br}}\left(\mathbf{A}, \widetilde{\mathbf{s}}_t^{\mathrm{br}}\right) \tag{15}$$

$$\mathbf{c}_t^{\mathrm{br}} = \sum_i \alpha_{ti}^{\mathrm{br}} \mathbf{a}_i \tag{16}$$

The function $f_{\mathrm{att}}^{\mathrm{br}}$ is designed as follows:

$$e_{ti}^{\mathrm{br}} = \mathbf{V}_{\mathrm{br}}^{\mathrm{T}} \tanh\left(\mathbf{W}_{\mathrm{att}}^{\mathrm{br}} \tilde{\mathbf{s}}_t^{\mathrm{br}} + \mathbf{U}_{\mathrm{att}}^{\mathrm{br}} \mathbf{a}_i\right) \tag{17}$$

$$\alpha_{ti}^{\mathrm{br}} = \frac{\exp\left(e_{ti}^{\mathrm{br}}\right)}{\sum_k \exp\left(e_{tk}^{\mathrm{br}}\right)} \tag{18}$$

where $\alpha_{ti}^{\mathrm{br}}$ denotes the branch attention probability of i-th element at step t, e_{ti}^{br} denotes the output energy of branch module at i-th step.

Then, we use $\mathbf{c}_t^{\mathrm{br}}$ and $\widetilde{\mathbf{s}}_t^{\mathrm{br}}$ as the input of $\mathrm{GRU}_2^{\mathrm{br}}$ to calculate the hidden state $\mathbf{s}_t^{\mathrm{br}}$ of the branch prediction module:

$$\mathbf{s}_t^{\mathrm{br}} = \mathrm{GRU}_2^{\mathrm{br}}\left(\mathbf{c}_t^{\mathrm{br}}, \widetilde{\mathbf{s}}_t^{\mathrm{br}}\right) \tag{19}$$

The embedding $\mathbf{e}_t^{\mathrm{node}}$ of the current node, the branch prediction module's hidden state $\mathbf{s}_t^{\mathrm{br}}$ and branch context vector $\mathbf{c}_t^{\mathrm{br}}$ are connected as the input of full-connection layer to calculate the prediction probability of the current branch $\mathbf{o}_t^{\mathrm{br}}$.

$$\mathbf{e}_t^{\mathrm{node}} = \mathrm{Emd}_{\mathrm{node}}(\mathbf{o}_t^{\mathrm{node}}) \tag{20}$$

$$\mathbf{h}_t^{\mathrm{br}} = \mathrm{maxout}\left(\mathbf{W}_1^{\mathrm{br}}\left[\mathbf{e}_t^{\mathrm{node}}, \mathbf{s}_t^{\mathrm{br}}, \mathbf{c}_t^{\mathrm{br}}\right]\right) \tag{21}$$

$$\mathbf{o}_t^{\mathrm{br}} = \mathrm{softmax}\left(\mathbf{W}_2^{\mathrm{br}} \mathbf{h}_t^{\mathrm{br}}\right) \tag{22}$$

We use the binary cross entropy function to compute the classification loss:

$$\mathcal{L}_{\mathrm{br}} = -\sum_t \log \mathbf{o}_t^{\mathrm{br}} \cdot \mathbf{r}_t \tag{23}$$

where $\mathbf{r}_t$ denotes the ground-truth branch label at time step t.

Adding "thinking" Label

When the spatial positional relationship between the parent node and the child node is too far away, the attention mechanism needs to span a far position to focus on the child node. We hope to add a "thinking" label before looking for the child node, so that the attention mechanism can observe and think first. The thinking label makes the attention mechanism pay attention to the position closer to the

child node first, and then pay attention to the child node. Therefore, the model becomes more accurate in predicting nodes and branches. From the perspective of tree structure, each node is the starting node of a subtree. Taking mathematical expressions as an example, we treat each subtree as a new mathematical expression. At this point, the thinking label can be understood as the root node of each mathematical formula. Each time the current node is predicted, a root symbol is predicted first, which indicates the beginning of the next subexpression. When encountering mathematical expressions with more nested structures, each nested subexpression will be treated as a new mathematical expression by the model, so the generalization ability of the model can be improved.

Pixel-level Auxiliary Classification Loss

In order to better optimize the encoder to obtain a stronger ability to extract features, we add a pixel-assisted classification loss. We directly classify each position feature vector $\mathbf{a}_i$ in the feature map $\mathbf{A}$ obtained by the encoder. Using the pixel-level feature $\mathbf{a}_i$, we can obtain the pixel-level classification probability $\mathbf{p}_i^{\text{pixel}}$ through two fully connected layers and two activation layers:

$$\mathbf{h}_i^{\text{pixel}} = \text{maxout}\left(\mathbf{W}_1^{\text{pixel}}\mathbf{a}_i\right) \tag{24}$$

$$\mathbf{p}_i^{\text{pixel}} = \text{softmax}\left(\mathbf{W}_2^{\text{pixel}}\mathbf{h}_i^{\text{pixel}}\right) \tag{25}$$

At each decoding step t, the attention probability $\boldsymbol{\alpha}_t^{\text{node}}$ is used to perform a weighted summation of the pixel-level classification probability $\mathbf{p}_i^{\text{pixel}}$ to obtain the classification probability $\mathbf{o}_t^{\text{pixel}}$ at step t:

$$\mathbf{o}_t^{\text{pixel}} = \sum_i \alpha_{ti}^{\text{node}}\mathbf{p}_i^{\text{pixel}} \tag{26}$$

Same as the node classification module, we use the cross-entropy function to calculate the auxiliary classification loss:

$$\mathcal{L}_{\text{pixel}} = -\sum_t \log \mathbf{o}_t^{\text{pixel}} \cdot \mathbf{n}_t \tag{27}$$

Where $\mathbf{n}_t$ denotes the one-hot vector of ground-truth node at time step t.

The gradient backpropagation produced by this loss skips the decoder. This is a shortcut to better optimize the encoder. We only use the auxiliary classification loss in the training phase, and do not use the auxiliary classification probability in the inference phase.

Implementation Details

Training

TDv2 has three losses in the training process: node classification loss $\mathcal{L}_{\text{node}}$, pixel-level auxiliary classification loss $\mathcal{L}_{\text{pixel}}$ and branch classification loss of $\mathcal{L}_{\text{br}}$. These three losses are minimized under end-to-end training, and we set the optimization goal of the model as:

$$L = \lambda_1\mathcal{L}_{\text{node}} + \lambda_2\mathcal{L}_{\text{br}} + \lambda_3\mathcal{L}_{\text{pixel}} \tag{28}$$

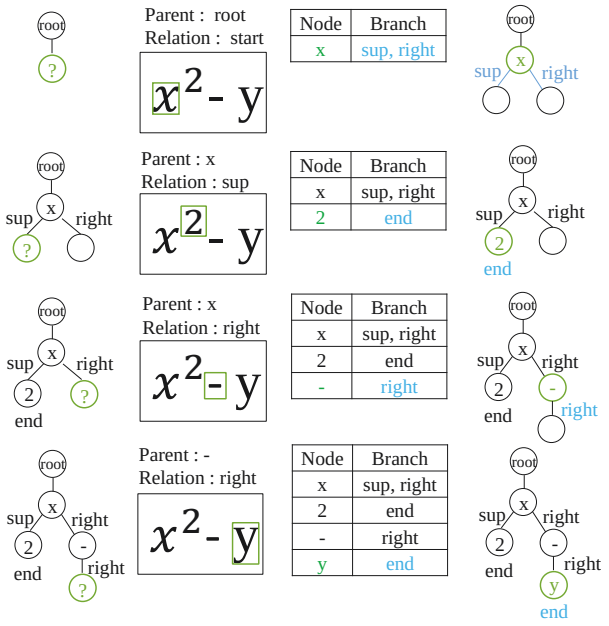

Figure 4: The tree construction process with the stack changing in inference stage. Node in green denotes current decoding node. Branch in blue denotes current decoding branch.

In our experiment, we set $\lambda_1 = \lambda_2 = 1, \lambda_3 = 0.5$. The node classification module and the branch prediction module are the most important parts of the model, so the weights are set relatively large. The pixel-level auxiliary classification loss only plays an auxiliary role, and does not participate in the prediction in the inference stage, so the weight setting is relatively small. The encoder uses a DenseNet, which is mainly composed of a DenseBlock and a Transition layer. We set the depth of each DenseBlock to 22, the channel of each layer to 24. After each convolutional layer, batch normalization (Ioffe and Szegedy 2015) is used, and the activation function is ReLU (Nair and Hinton 2010). Both the node classification module and branch prediction module adopt 2 GRU layers. The dimension of the GRU unit, the dimension of the word embedding layer and the dimension of the relation embedding layer are set to 256. The dimension of the node attention mechanism and the branch attention mechanism is set to 512. The optimization algorithm is Adadelta (Zeiler 2012) with gradient clipping, and the learning rate is set to 1. We implement TDv2 using PyTorch and conduct experiments on a Nvidia Tesla V100 with 12GB RAM.

Testing

In the inference stage, we use a stack table to record the node information: the symbol category and the branches for building a mathematical expression tree. In each step, we pop parent node and spatial relationship as the input of the current decoding step, and TDv2 only needs to be responsible for predicting current symbol and branches, which are then pushed into stack. The specific construction process is shown in Figure 4. First, we initialize the parent node as

System	Thinking	Pixel	ExpRate		
			14	16	19
E1			49.46	47.32	51.90
E2	✓		53.65	50.87	55.91
E3	✓	✓	53.62	54.16	58.72

Table 1: Results of Ablation Experiment (in %) on CROHME 2014, CROHME 2016 and CROHME 2019.

"root" and the spatial relationship as "start". When the tree decoder decodes the node "x" and the spatial relationship "sup" and "right" in the first step, we store them in the stack table and fill in "x" in the mathematical expression tree and create two branches on the "x" node. As shown in the second line of Figure 4, when we need to decode the superscript node of "x", we can automatically pop up the parent node "x" and the spatial relationship "sup" through the stack table. In essence, it is equivalent to finding its parent node and spatial relationship along the empty node marked with a green question mark in Figure 4. After the node "2" and the branch "end" are decoded, they are stored in the stack table and the mathematical expression tree is further constructed. This step is repeated until the stack table is empty or the nodes of the mathematical expression tree are completely traversed.

Dataset

We verified the proposed model on CROHME benchmark (Mouchere et al. 2016; Mouchère et al. 2016; Mahdavi et al. 2019), which is the most widely used public dataset for HMER. The training set has 8836 mathematical expressions, including 101 mathematical symbol classes and 9 mathematical spatial relations. We define the spatial relationship as: start, leftsup, inside, below, above, subscript, superscript, right and end. CROHME 2014/2016/2019 test sets contain 986/1147/1199 handwritten mathematical expressions respectively. Among them, CROHME 2016 is the most difficult data set, and most methods have achieved lower accuracy on it.

Experiments

Ablation Experiment and Visualization

In order to verify whether the two improvements proposed in Section 3.4 and Section 3.5 can effectively improve the performance of TDv2, we do this set of ablation experiments on the CROHME dataset. According to the idea proposed in Section 3.4, we add a "thinking" label before each real node and denote this system as E2. As shown in Table 1, adding "thinking" labels increases the accuracy of the mathematical expressions of the model on the test sets of CROHME 2014, CROHME 2016 and CROHME 2019 by 4.19%, 3.55% and 4.01% respectively.

In order to more intuitively explain the effect of adding thinking labels, we visualize the attention in the process of node recognition. As shown in Figure 5, when the decoding step is "thinking", the attention mechanism focuses on the middle area between the parent node and the child node. Without thinking about the label, the attention mechanism should directly focus on the child node from the parent node. After adding the thinking label, the model can first find an area closer to the child node, which enables the attention mechanism to more accurately focus on the location of the child node next time. Take the node in the red box in Figure 5 as an example. When the parent node is "0" and the spatial relationship is "sup", the attention mechanism needs to find the child node "frac". After adding the thinking tag, the attention mechanism first focuses on the area between "0" and "frac", and then on "frac".

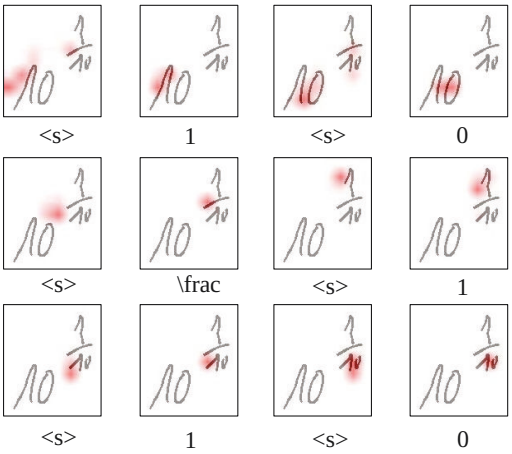

Figure 5: The attention visualization after adding the "thinking" labels. The red parts denote the attention probabilities of current step. "⟨s⟩" denotes the "thinking" label.

Finally, based on the system E2, we add the pixel-level auxiliary classification loss proposed in Section 3.5 and denote the system as E3. As shown in Table 1, the ExpRate of system E3 on the testsets of CROHME 2014, CROHME 2016 and CROHME 2019 increase by -0.03%, 3.29% and 2.81% respectively. Although the ExpRate on the CROHME 2014 testset shows a very small drop, the ExpRate on the more difficult CROHME 2016 testset improves significantly. In general, this set of ablation experiments proves that adding thinking labels and pixel-level auxiliary classification loss can effectively improve the performance of TDv2.

Evaluating the Generalization of the Model

In order to prove that TDv2 has better generalization than TD (Zhang et al. 2020), we design this set of experiments on the depth of mathematical expressions. As shown in the Figure 6, with depth-first traversal order, we call the number of nodes between the non-root node and its parent node 'distance'. The maximum distance is defined as the depth of the tree. We use printed mathematical expressions with a depth of no more than 4 as the training set and test the generalization of the model on test sets with different depths. As shown in Figure 7, the depth of part of the test set exceeds the depth of the training set, and even exceeds 11. When the depth of

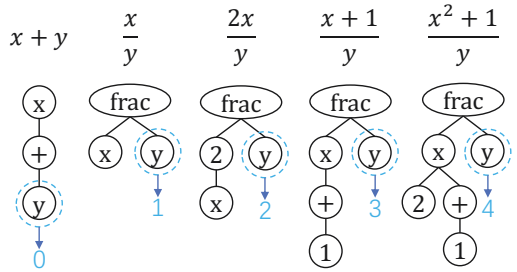

Figure 6: Illustration of math formulas with increased structural depth. Number in blue denotes the depth of current node.

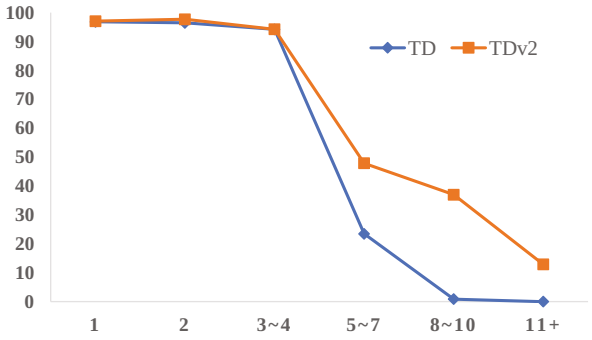

Figure 7: In-depth experiment of mathematical expressions. Note: The abscissa indicates the depth of the test set, and the ordinate indicates the accuracy rate (%).

the mathematical expression in the test set is greater than the depth in the training set, TD can hardly recognize the mathematical expression accurately. However, TDv2 can still identify some of the deep samples. That fully proves that TDv2 has better generalization and can learn the structural information of mathematical expressions more effectively from the relatively simple samples in the training set.

Comparison with State-of-the-Art

We compare this model with other state-of-the-art methods to show the competitive results of our proposed model. For fairness, all our experiments only use offline information in the official CROHME training set without additional data enhancement, beam search strategy or model fusion.

As shown in the Table 2, we select a large number of mainstream algorithms for comparison, including methods with string decoder and tree decoder. Except that the experimental results of our model were measured by the tool (Mouchère et al. 2016) provided by CROHME, all other experimental results are selected from other published papers. Among those, "WAP" (Zhang et al. 2017a), "DenseWAP", "DenseWAP-TD" (Zhang et al. 2020) , "PAL" (Wu et al. 2018), "PAL-2" (Wu et al. 2020) and "DLA" (Le 2020) are very representative and most advanced systems based on deep learning published in recent years. The ExpRate of TDv2 is 53.62% on CROHME 2014, 55.18% on CROHME 2016, and 58.72% on CROHME 2019. Obviously, TDv2 is significantly better than other encoder-decoder models on all three data sets in terms of expression recognition rate. It is particularly worth mentioning that CROHME 2016 has always been the most difficult one of the three CROHME test sets, and the results of most systems on this CROHME 2016 are worse than those on CROHME 2014 and CROHME 2019. TDv2 achieved quite good results on the CROHME 2016 test set, even 1.56% better than that on CROHME 2014. Compared with other methods, TDv2 has achieved a significant improvement even without using data enhancement and beam search strategies. These experimental results fully prove that TDv2 has good generalization and superior performance for HMER.

Dataset	Model	Decoder	ExpRate
CROHME 14	WYGIWYS	SD	36.4
	WAP	SD	40.4
	DenseWAP	SD	43.0
	DenseWAP-TD	TD	49.1
	PAL	SD	39.66
	PAL-2	SD	48.88
	DLA	SD	51.88
	TDv2	**TD**	**53.62**
CROHME 16	WAP	SD	37.1
	DenseWAP	SD	40.1
	DenseWAP-TD	TD	48.5
	PAL-v2	SD	49.61
	DLA	SD	51.53
	TDv2	**TD**	**55.18**
CROHME 19	WAP	SD	37.1
	DenseWAP	SD	41.7
	DenseWAP-TD	TD	51.4
	TDv2	**TD**	**58.72**

Table 2: Evaluation of math formula recognition systems on CROHME 2014, 2016 and 2019 (%). 'SD' denotes methods based on string decoder. 'TD' denotes methods based on tree decoder. 'ExpRate' denotes the metric expression recognition rate.

Conclusion

In this study, we propose a novel tree based decoder which includes a node classification module and a branch prediction module for HMER. Furthermore, we added thinking labels and pixel-level auxiliary classification loss in TDv2 to improve the performance. Through detailed experimental analysis and comparisons with state-of-the-art methods, we clearly demonstrate that TDv2 has good generalization and superior performance for HMER. In tht future, we will implement TDv2 for online handwriting mathematical expression recognition and investigate its application on other tree-structured text recognition.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grants No. 62171427, and Ximalaya Inc.

References

Álvaro, F.; Sánchez, J.-A.; and Benedí, J.-M. 2014. Recognition of on-line handwritten mathematical expressions using 2D stochastic context-free grammars and hidden Markov models. *Pattern Recognition Letters*, 35: 58–67.

Álvaro, F.; Sánchez, J.-A.; and Benedí, J.-M. 2016. An integrated grammar-based approach for mathematical expression recognition. *Pattern Recognition*, 51: 135–147.

Awal, A.-M.; Mouchère, H.; and Viard-Gaudin, C. 2014. A global learning approach for an online handwritten mathematical expression recognition system. *Pattern Recognition Letters*, 35: 68–77.

Chakraborty, S.; Allamanis, M.; and Ray, B. 2018. Tree2tree neural translation model for learning source code changes. *arXiv preprint arXiv:1810.00314*.

Chan, W.; Jaitly, N.; Le, Q.; and Vinyals, O. 2016. 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)*, 4960–4964. IEEE.

Chen, X.; Liu, C.; and Song, D. 2018. Tree-to-tree neural networks for program translation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2552–2562.

Cheng, Z.; Bai, F.; Xu, Y.; Zheng, G.; Pu, S.; and Zhou, S. 2017. Focusing attention: Towards accurate text recognition in natural images. In *Proceedings of the IEEE international conference on computer vision*, 5076–5084.

Cho, K.; Courville, A.; and Bengio, Y. 2015. Describing multimedia content using attention-based encoder-decoder networks. *IEEE Transactions on Multimedia*, 17(11): 1875–1886.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Deng, Y.; Kanervisto, A.; Ling, J.; and Rush, A. M. 2017. Image-to-markup generation with coarse-to-fine attention. In *International Conference on Machine Learning*, 980–989. PMLR.

Dong, L.; and Lapata, M. 2016. Language to logical form with neural attention. *arXiv preprint arXiv:1601.01280*.

Eriguchi, A.; Hashimoto, K.; and Tsuruoka, Y. 2016. Tree-to-sequence attentional neural machine translation. *arXiv preprint arXiv:1603.06075*.

Hong, Z.; You, N.; Tan, J.; and Bi, N. 2019. Residual birnn based seq2seq model with transition probability matrix for online handwritten mathematical expression recognition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, 635–640. IEEE.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708.

Huang, P.-Y.; Liu, F.; Shiang, S.-R.; Oh, J.; and Dyer, C. 2016. Attention-based multimodal neural machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, 639–645.

Ioffe, S.; and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, 448–456. PMLR.

Le, A. D. 2020. 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*, 566–567.

Le, A. D.; Indurkhya, B.; and Nakagawa, M. 2019. Pattern generation strategies for improving recognition of handwritten mathematical expressions. *Pattern Recognition Letters*, 128: 255–262.

Le, A. D.; and Nakagawa, M. 2017. 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)*, volume 1, 1056–1061. IEEE.

Li, H.; Wang, P.; Shen, C.; and Zhang, G. 2019. Show, attend and read: A simple and strong baseline for irregular text recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 8610–8617.

Li, Z.; Jin, L.; Lai, S.; and Zhu, Y. 2020. Improving Attention-Based Handwritten Mathematical Expression Recognition with Scale Augmentation and Drop Attention. In *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 175–180. IEEE.

Mahdavi, M.; Zanibbi, R.; Mouchere, H.; Viard-Gaudin, C.; and Garain, U. 2019. 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)*, 1533–1538. IEEE.

Mouchère, H.; Viard-Gaudin, C.; Zanibbi, R.; and Garain, U. 2016. ICFHR2016 CROHME: Competition on recognition of online handwritten mathematical expressions. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 607–612. IEEE.

Mouchere, H.; Zanibbi, R.; Garain, U.; and Viard-Gaudin, C. 2016. 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.

Nair, V.; and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 807–814.

Parisotto, E.; Mohamed, A.-r.; Singh, R.; Li, L.; Zhou, D.; and Kohli, P. 2017. Neuro-Symbolic Program Synthesis. In *ICLR (Poster)*.

Rabinovich, M.; Stern, M.; and Klein, D. 2017. Abstract Syntax Networks for Code Generation and Semantic Parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1139–1149.

Shi, B.; Bai, X.; and Yao, C. 2016. 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.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Socher, R.; Lin, C. C.-Y.; Ng, A. Y.; and Manning, C. D. 2011. Parsing natural scenes and natural language with recursive neural networks. In *ICML*.

Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Wang, J.; Du, J.; Zhang, J.; Wang, B.; and Ren, B. 2021. Stroke constrained attention network for online handwritten mathematical expression recognition. *Pattern Recognition*, 108047.

Wang, J.; Du, J.; Zhang, J.; and Wang, Z.-R. 2019. Multi-modal attention network for handwritten mathematical expression recognition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, 1181–1186. IEEE.

Wu, J.-W.; Yin, F.; Zhang, Y.-M.; Zhang, X.-Y.; and Liu, C.-L. 2018. Image-to-markup generation via paired adversarial learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 18–34. Springer.

Wu, J.-W.; Yin, F.; Zhang, Y.-M.; Zhang, X.-Y.; and Liu, C.-L. 2020. Handwritten mathematical expression recognition via paired adversarial learning. *International Journal of Computer Vision*, 1–16.

Wu, J.-W.; Yin, F.; Zhang, Y.-M.; Zhang, X.-Y.; and Liu, C.-L. 2021. Graph-to-Graph: Towards Accurate and Interpretable Online Handwritten Mathematical Expression Recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 2925–2933.

Zanibbi, R.; Blostein, D.; and Cordy, J. R. 2002. Recognizing mathematical expressions using tree transformation. *IEEE Transactions on pattern analysis and machine intelligence*, 24(11): 1455–1467.

Zeiler, M. D. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhang, J.; Du, J.; and Dai, L. 2018a. Multi-scale attention with dense encoder for handwritten mathematical expression recognition. In *2018 24th International Conference on Pattern Recognition (ICPR)*, 2245–2250. IEEE.

Zhang, J.; Du, J.; and Dai, L. 2018b. Track, attend, and parse (tap): An end-to-end framework for online handwritten mathematical expression recognition. *IEEE Transactions on Multimedia*, 21(1): 221–233.

Zhang, J.; Du, J.; and Dai, L. 2019. Track, Attend, and Parse (TAP): An End-to-End Framework for Online Handwritten Mathematical Expression Recognition. *IEEE Transactions on Multimedia*, 21(1): 221–233.

Zhang, J.; Du, J.; Yang, Y.; Song, Y.-Z.; Wei, S.; and Dai, L. 2020. A Tree-Structured Decoder for Image-to-Markup Generation. In *International Conference on Machine Learning*, 11076–11085. PMLR.

Zhang, J.; Du, J.; Zhang, S.; Liu, D.; Hu, Y.; Hu, J.; Wei, S.; and Dai, L. 2017a. Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition. *Pattern Recognition*, 71: 196–206.

Zhang, J.; Du, J.; Zhang, S.; Liu, D.; Hu, Y.; Hu, J.; Wei, S.; and Dai, L. 2017b. Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition. *Pattern Recognition*, 71: 196–206.

Zhu, X.; Sobhani, P.; and Guo, H. 2015. Long short-term memory over tree structures. *arXiv preprint arXiv:1503.04881*.