This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2020.3011316, IEEE Transactions on Multimedia

SRD: A Tree Structure Based Decoder for Online Handwritten Mathematical Expression Recognition

Jianshu Zhang, Jun Du, Yongxin Yang, Yi-Zhe Song, and Lirong Dai

***Abstract*—Recently, recognition of online handwritten mathematical expression has been greatly improved by employing encoder-decoder based methods. Existing encoder-decoder models use string decoders to generate LaTeX strings for mathematical expression recognition. However, in this paper, we importantly argue that string representations might not be the most natural for mathematical expressions – mathematical expressions are inherently tree structures other than flat strings. For this purpose, we propose a novel sequential relation decoder (SRD) that aims to decode expressions into tree structures for online handwritten mathematical expression recognition. At each step of tree construction, a sub-tree structure composed of a relation node and two symbol nodes is computed based on previous sub-tree structures. This is the first work that builds a tree structure based decoder for encoder-decoder based mathematical expression recognition. Compared with string decoders, a decoder that better understands tree structures is crucial for mathematical expression recognition as it brings a more reasonable learning objective and improves overall generalization ability. We demonstrate how the proposed SRD outperforms state-of-the-art string decoders through a set of experiments on CROHME database, which is currently the largest benchmark for online handwritten mathematical expression recognition.**

***Index Terms*—Handwritten Mathematical Expression Recognition, Tree Structure, Decoder, Attention**

I. Introduction

Handwritten mathematical expressions are common in our daily life as they are indispensable for describing problems or theories in math, physics and many other fields. Unlike English or other languages which usually writes in one direction without further internal structures, mathematical expression is a more complicated two-dimensional language with internal tree structures (see in Figure 1(a)). Therefore, mathematical expression recognition becomes a challenging problem as complexity of the internal tree structures can often be enormous [1], [2]. Achieving success in this problem could, in turn, accelerate progress in machine recognition of other tree-structured languages, like chemical formula and flow chart.

Mathematical expression recognition comprises two major problems [3]: symbol recognition and structural analysis. Traditional methods usually first segment an expression into several math symbols and recognize them. These recognized math symbols together with the possible relations among them construct a math forest. Then structural analysis searches for the most likely math tree (see in Figure 1(b)) directly from the math forest. During structural analysis, using a pre-defined math grammar helps improve the searching accuracy as it can make sure the output tree structure is always consistent with the grammar of math language.

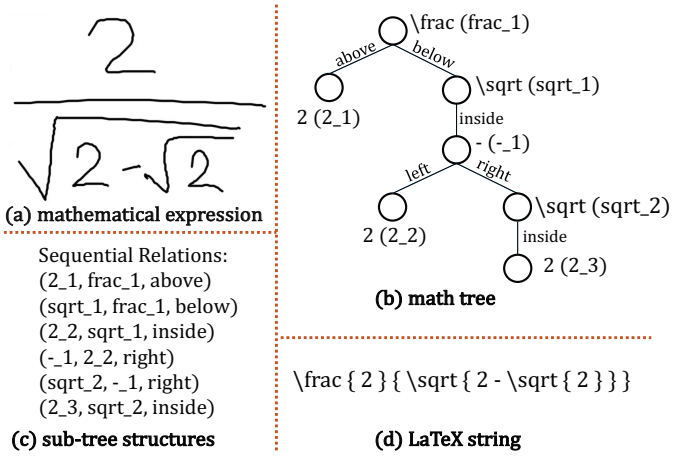

Fig. 1. (a) An example of handwritten mathematical expression; (b) Math tree structure of the expression; (c) Sub-tree structures of the expression; (d) LaTeX string of the expression.

Recently, some researches propose to convert the mathematical expression recognition problem from tree generation into string generation because (i) a mathematical expression can be considered as correctly recognized when its corresponding LaTeX string (see in Figure 1(d)) has been generated and (ii) basically, string generation is easier to be implemented than tree generation. Following the success in sequence to sequence learning [4], [5] and image to sequence learning [6], [7], researchers started to use the attention based encoder-decoder model to generate LaTeX string for mathematical expression recognition [8], [9], [10]. Compared with traditional methods, the encoder-decoder based LaTeX string generators have substantially improved the expression recognition performance because they are end-to-end trainable and can be free of symbol segmentation.

In this study, unlike previous works that employ LaTeX string decoders, we introduce a novel sequential relation decoder (SRD) which is a tree-structured decoder for mathematical expression recognition. The SRD is a more natural decoder for mathematical expression recognition as mathematical expressions are inherently represented as tree structures other than flat strings. As shown in Figure 1(c), the tree structure of

Jianshu Zhang, Jun Du and Lirong Dai were with the National Engineering Laboratory for Speech and Language Information Processing, University of Science and Technology of China, Hefei, Anhui, P. R. China. e-mail: xysszjs@mail.ustc.edu.cn, jundu@ustc.edu.cn, lrdai@ustc.edu.cn. Yongxin Yang and Yi-Zhe Song were with the University of Surrey, UK. e-mail: yongxin.yang@surrey.ac.uk, y.song@surrey.ac.uk.

mathematical expression can be decomposed into several sub-tree structures. The proposed SRD generates these sub-tree structures step by step. At each step, each sub-tree structure composed of a relation node and two symbol nodes is computed based on previous sub-tree structures. More specifically, the two symbol nodes are named as primary symbol node and related symbol node respectively. The primary symbol node represents the symbol leading the decoding procedure and the related symbol node represents the symbol that has a relation with the primary symbol node. Note that, the one-dimensional language recognition can be seen as a special case of tree-structured language recognition for SRD, because in one-dimensional language recognition the current related symbol node will always be the previous primary symbol node. We employ two spatial attention models that learn to locate the two math symbols and then recognize them. After determining the two symbol nodes, the relation node describing the relation between two symbols can be classified by feeding the spatial information of two symbols into a fully-connected layer. The final complete mathematical tree structure is constructed by organizing the output sub-tree structures.

Compared with LaTeX string decoders, the SRD has three distinctive advantages: (i) it brings a more reasonable learning objective to train the whole encoder-decoder model for mathematical expression recognition; (ii) it improves overall generalization ability of the recognition model as it understands the sub-structures of a mathematical expression rather than remembers an entire LaTeX string; (iii) string decoders are data-driven, they do not employ a math grammar, therefore in bad situations, string decoders will output some strings that totally disobey the math grammar, while SRD ensures that the output will always follow the tree-based structures, which helps alleviate the bad situations. Besides, following the superiority of string decoders, the proposed SRD can also be jointly trained with the whole encoder-decoder framework.

We summarize the main contributions of this study as:

- We introduce a novel tree structure based decoder for mathematical expression recognition, which can be applied for other tree-structured languages.
- The novel SRD significantly outperforms the string decoder, indicating the effectiveness of a tree structure based decoder for mathematical expression recognition.
- We show how the proposed SRD decodes the tree structure step by step and demonstrate its advantages by experimental analysis.

The source codes of SRD are publicly available[1].

The rest of this paper is organized as follows: Section II introduces the related works. Section III describes the proposed framework of the whole recognition system. Section IV introduces the implementation of the training and testing procedures. Section V reports the experimental results, and Section VI presents concluding remarks.

II. Related Works

In this section, we describe the previous work on handwritten mathematical expression recognition, including both traditional grammar based approaches and recent encoder-decoder based approaches. We also describe the recent work on neural network based tree structure modeling.

A. Traditional Methods for Mathematical Expression Recognition

Mathematical expression recognition consists of two major problems: symbol recognition and structural analysis. The two problems can be solved sequentially or globally.

1) Sequential Approaches: Sequential approaches [11] implemented symbol recognition and structural analysis separately. They first segmented the whole mathematical expression into several instances and then recognized the segmented instances into math symbols. Based on the best symbol segmentation and symbol recognition results, the analysis of two-dimensional structures was performed to find the most likely math tree. In sequential approaches, symbol recognition can not use contextual information of the whole expression, while the contextual information could be helpful for recognizing ambiguous isolated symbols, especially for handwritten symbols. Besides, the symbol segmentation and recognition errors will be subsequently inherited by structural analysis.

2) Global Approaches: Global approaches [12], [13] optimized symbol recognition and structural analysis simultaneously, which seem to be more appropriate than sequential approaches as symbol recognition and structural analysis can both utilized global contextual information. The recent LaTeX string decoder based methods and the proposed SRD all belong to global approaches because the encoder and the decoder are optimized in a joint way. However, traditional global approaches are computationally more expensive because the probabilities for symbol segmentation are exponentially expanded. Therefore, effective search strategies must be executed [14].

3) Structural Analysis Methods: As for symbol recognition, traditional methods usually employed convolutional neural network (CNN) [15] and recurrent neural network (RNN) [16] as symbol classifier. While for analysis of math tree structures, many approaches have been investigated, like maximum spanning tree [17], two-dimensional HMM [18], matching based approaches [19] and grammar based approaches [11], [13], [20], [21]. Among these, the grammar based methods seem to be more dependable. For example, the stochastic two-dimensional context-free grammars have performed well in the two systems [11], [13], while [11] and [13] both achieved the first place on CROHME competition, which is the largest competition of online handwritten mathematical expression recognition.

B. LaTeX String Decoder for Mathematical Expression Recognition

Recently, attention based encoder-decoder models have been extensively applied to many applications including machine translation [22], speech recognition [5], [23], image captioning [24], [25] and video processing [26], which implies researchers that LaTeX string decoder based mathematical

[1]https://github.com/JianshuZhang/SRD

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2020.3011316, IEEE Transactions on Multimedia

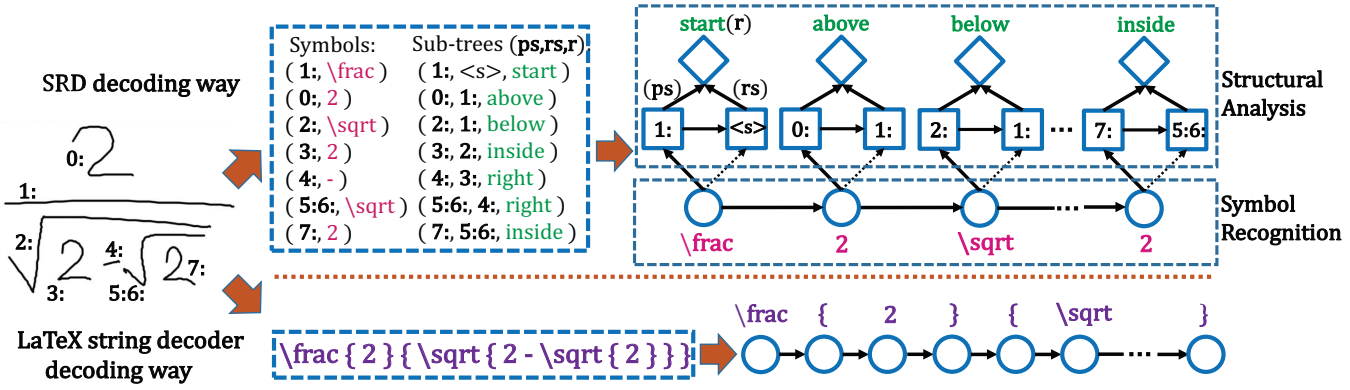

Fig. 2. Comparison between the decoding procedure of SRD and the decoding procedure of LaTeX string decoder. In the top-left box, below "Symbols" we show each math symbol and its alignment to strokes, below "Sub-trees" we show each sub-tree structure composed of primary symbol, related symbol and relation $(\mathbf{ps}, \mathbf{rs}, \mathbf{r})$. In the top-right box, SRD generates the sequential sub-tree structures step by step with parent node denoting spatial relationship, left child node denoting primary symbol and right child node denoting related symbol. In the box below the top-right box, SRD generates the sequential math symbols step by step.

expression recognition can also be a good application. Current LaTeX string decoder based approaches can be divided into offline recognition approaches and online recognition approaches.

1) Offline Recognition: For offline recognition, the input mathematical expressions are stored in images. In [8], a model named WAP was introduced, which first proposed to use LaTeX string decoder for offline handwritten mathematical expression recognition. Compared with traditional methods, WAP achieved significant improvements both on recognition rate and efficiency, indicating the advantage of end-to-end trainable model. Then, in [27], a DenseNet encoder and multi-scale attention mechanism were proposed for further improving the WAP model. Besides, [10] proposed a model named WYGIWYS with a coarse-to-fine attention for mathematical expression recognition and proved that the string decoder can be applied to other two-dimensional markup languages. Also, [28] proposed an adversarial learning strategy and introduced a PAL model, which achieved currently the best published results for offline mathematical expression recognition.

2) Online Recognition: In the data acquisition of online handwriting, the pen-tip movements (xy-coordinates) are automatically stored as sequential data. Clearly, the pen-tip movements can be transformed into image-like representations, then researchers can employ the previous offline approaches for online handwritten mathematical expression recognition. While [9] proposed a TAP model that utilized RNN to replace CNN to be the encoder so that the model can better use the dynamic information of handwriting traces, the extension of TAP [29] made use of the complementarity between dynamic traces and static images and achieved currently the best published results for online handwritten mathematical expression recognition. In this paper, we implement the proposed SRD for online handwritten mathematical expression recognition, which means the input is sequential traces.

C. Researches on Tree Structure Modeling

In [30], the first tree-structured RNN was proposed, named Tree-LSTM. The Tree-LSTM aimed to append the tree-structure into LSTM nodes to improve the semantic representations of LSTM features, especially for those tree-structured modeling tasks, like syntactic parsing and many other natural language processing tasks. Then, in [31], the researchers employed the Tree-LSTM to improve the encoding of online handwriting mathematical expression since mathematical expression is a typical tree-structured language.

Although prior works have recognized the importance of modeling the tree structure of an object, they mainly focused on the problem of encoding the tree structure, but rarely focused on the more important problem of decoding the tree structure. Tree decoder is harder to be implemented than tree encoder because it is difficult to perform recursive modeling during step-by-step decoding. Although some tree decoders have been proposed, they usually utilized task-specific structure and constrained definition to decrease the difficulty of tree-structured modeling, therefore can not be extented for other tasks. For example, in [32], a well-performed tree decoder for tree-structured program translation was proposed. But it was constructed based on program knowledge and can only deal with binary tree structure, therefore can not be applied on mathematical expression recognition. In this paper, we utilize the properties of handwriting recognition problem (stroke information) to overcome the difficulties of tree generation and for the first time, we build a tree structure based decoder for mathematical expression recognition.

III. Proposed Method

In this section, we first illustrate the main architecture of the proposed SRD and compare it to the LaTeX string decoder. Then, we elaborate the overall system for online handwritten mathematical expression recognition, including both the encoder and the proposed SRD. In Section III-A, we introduce the RNN encoder which extracts high-level features from raw handwriting input. In Section III-B, we introduce the SRD which is devised to address symbol recognition and structural analysis simultaneously, the training loss of symbol recognition and structural analysis are both given in detail.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2020.3011316, IEEE Transactions on Multimedia

Unlike LaTeX string decoders, SRD aims to generate a complete mathematical expression tree. As shown in the top-left box in Figure 2, the math tree can be decomposed into a sequence of math symbols and a sequence of sub-tree structures, called label graph style [33]. Generating the two sequences turns into the problem of symbol recognition and structural analysis respectively, as illustrated in the top-right two boxes in Figure 2. Here, each sub-tree structure is composed of a primary symbol node, a related symbol node and a relation node, denoted as $(\mathbf{ps}, \mathbf{rs}, \mathbf{r})$. The primary symbol node and the related symbol node describe the absolute spatial positions of primary math symbol and related math symbol respectively, and the relation node describes the spatial relationship between the two math symbols. Note that, we propose to use absolute spatial positions of math symbols to be symbol nodes because one math symbol (e.g., the symbol "2" in Figure 1) might appear multiple times in one mathematical expression tree which brings confusion, but their absolute spatial positions can distinguish them. For online recognition, the absolute spatial positions are stroke indices of input traces (e.g. "0:" denotes the first stroke of handwriting input, "1:" denotes the second stroke), which is also the distinctive property of handwriting recognition problems.

A. Encoder

In this work, we aim to deal with the online recognition problem, which means the input is sequential handwriting traces. Therefore, we employ a RNN encoder to encode the sequential input. To alleviate the vanishing and exploding gradient problems of simple RNN [34], we employ Gated Recurrent Units (GRU) [35] to be the encoder, which is an improved version of simple RNN.

According to [36], we first normalize the raw input traces to address the issue of non-uniform sampling by different writing speed and size variations of the coordinates on different portable devices. We then extract an 8-dimensional feature vector for each point:

$$[x_i, y_i, \Delta x_i, \Delta y_i, \Delta' x_i, \Delta' y_i, strokeFlag1, strokeFlag2] \quad (1)$$

where x_i and y_i are xy-coordinates of the pen movements, $\Delta x_i = x_{i+1} - x_i$, $\Delta y_i = y_{i+1} - y_i$, $\Delta' x_i = x_{i+2} - x_i$, $\Delta' y_i = y_{i+2} - y_i$ and the last two terms are flags indicating the status of the pen, i.e., $[1, 0]$ means pen-down while $[0, 1]$ means pen-up. After the processing, a sequence of 8-dimensional feature vectors is then considered as the input to be fed into the encoder. To both utilize the history and future context information, we employ bidirectional GRU other than unidirectional GRU. We implement the bidirectional GRU by passing the input vectors through two GRU layers that run in opposite directions and concatenating their hidden states.

In this paper, to let the proposed SRD fairly comparable with the LaTeX string decoder based encoder-decoder models, we employ the same bidirectional GRU encoder architecture as introduced in TAP model [9]. The implementation details of the encoder can be seen in Section IV-A.

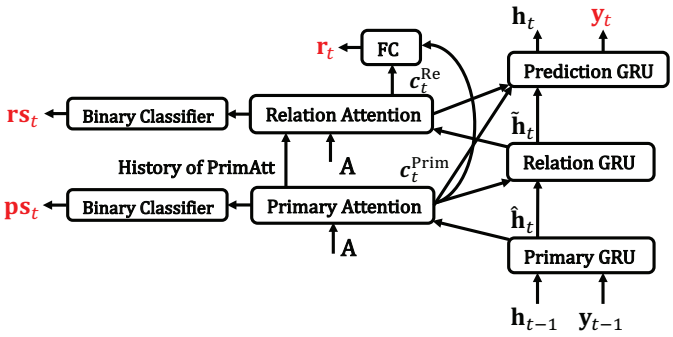

Fig. 3. We denote the output of SRD in color red, where $\mathbf{y}_t$ computes the symbol recognition loss, $\mathbf{ps}_t$, $\mathbf{rs}_t$ and $\mathbf{r}_t$ computes the structural analysis loss. As for the input of SRD, $\mathbf{h}_{t-1}$ denotes the previous prediction GRU state, $\mathbf{y}_{t-1}$ denotes the output symbol of previous step, $\mathbf{A}$ denotes the features extracted by encoder. $\hat{\mathbf{h}}_t$ denotes primary GRU state, $\tilde{\mathbf{h}}_t$ denotes the relation GRU state, $\mathbf{c}_t^{\mathrm{Prim}}$ denotes the context vector computed by primary attention, $\mathbf{c}_t^{\mathrm{Re}}$ denotes the context vector computed by relation attention.

B. Sequential Relation Decoder

Assuming the encoder output is an annotation sequence $\mathbf{A}$ with length L: $\mathbf{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_L\}, \mathbf{a}_i \in \mathbb{R}^D$, the SRD begins to generate a math tree structure using $\mathbf{A}$ to be the context input. Figure 3 illustrates the schematic representation of SRD.

1) Symbol Recognition Loss: We employ a primary GRU (PrimGRU), a primary attention model (PrimAtt) and a prediction GRU (PredGRU) to deal with the symbol recognition. The symbol sequence is denoted as $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_T\}, \mathbf{y}_t \in \mathbb{R}^K$, K is the number of math symbols in the symbol vocabulary. Note that, the order of the sub-tree sequence is determined based on the order of primary symbols, while the order of primary symbols is determined by traversing MathML (math markup language) structure of the expression following a depth-first order. Therefore, we utilize GRUs to be the basic components of SRD as we expect the recurrent units can learn this implicit order. Given input $\mathbf{x}_t$ and previous state $\mathbf{h}_{t-1}$, the GRU state $\mathbf{h}_t$ is computed by:

$$\mathbf{h}_t = \mathrm{GRU}\left(\mathbf{x}_t, \mathbf{h}_{t-1}\right) \quad (2)$$

To recognize the current output primary symbol $\mathbf{y}_t$, not the entire input handwriting traces is necessary to provide the useful information. Only a subset of input traces should mainly contribute to the computation of current output primary symbol. Therefore, we employ a primary attention model to attempt to find the alignment between output symbols and input handwriting traces. However, to learn the alignment between current output $\mathbf{y}_t$ and input handwriting traces, we only have the previous output symbol $\mathbf{y}_{t-1}$ and previous decoder hidden state $\mathbf{h}_{t-1}$. To alleviate the mismatch issue, we employ a primary GRU to compute the prediction of current primary hidden state $\hat{\mathbf{h}}_t$ given by the previous output symbol $\mathbf{y}_{t-1}$ and previous decoder state $\mathbf{h}_{t-1}$:

$$\hat{\mathbf{h}}_t = \mathrm{PrimGRU}\left(\mathbf{y}_{t-1}, \mathbf{h}_{t-1}\right). \quad (3)$$

Here, $\hat{\mathbf{h}}_t$ represents the information of current output primary symbol and $\mathbf{A}$ represents the high-level features of input handwriting traces, we then compute the alignment for primary

symbols by employing a primary attention model to generate attention probabilities based on $\hat{\mathbf{h}}_t$ and $\mathbf{A}$. The attention probabilities can be seen as the alignment because the annotation sequence $\mathbf{A}$ with higher attention probabilities will contribute more for current output primary symbol. We first compute the energy between $\hat{\mathbf{h}}_t$ and $\mathbf{A}$ as follows:

$$\hat{\mathbf{F}} = \mathbf{Q}_{\text{PrimAtt}} * \sum_{l=1}^{t-1} \boldsymbol{\alpha}_l^{\text{Prim}} \tag{4}$$

$$e_{ti}^{\text{Prim}} = \boldsymbol{\nu}_{\text{PrimAtt}}^{\text{T}} \tanh(\mathbf{W}_{\text{PrimAtt}}\hat{\mathbf{h}}_t + \mathbf{U}_{\text{PrimAtt}}\mathbf{a}_i + \hat{\mathbf{U}}_f\hat{\mathbf{f}}_i) \tag{5}$$

where $*$ denotes a convolution operation, $\sum_{l=1}^{t-1} \boldsymbol{\alpha}_l^{\text{Prim}}$ denotes the sum of past primary attention probabilities, e_{ti}^{Prim} denotes the output energy, $\hat{\mathbf{f}}_i$ denotes the elements of $\hat{\mathbf{F}}$. The $\hat{\mathbf{F}}$ is called coverage vector, we compute it by feeding the past primary attention into a convolution layer $\mathbf{Q}_{\text{PrimAtt}}$ so that $\hat{\mathbf{F}}$ can help alleviate the problem of standard attention mechanism, namely, lack of history information [37]. Let q denotes the number of output channels of convolution layer $\mathbf{Q}_{\text{PrimAtt}}$, n denotes the dimension of primary GRU and n' denotes the dimension of primary attention, $\boldsymbol{\nu}_{\text{PrimAtt}}^{\text{T}} \in \mathbb{R}^{n'}$, $\mathbf{W}_{\text{PrimAtt}} \in \mathbb{R}^{n' \times n}$, $\mathbf{U}_{\text{PrimAtt}} \in \mathbb{R}^{n' \times D}$ and $\hat{\mathbf{U}}_f \in \mathbb{R}^{n' \times q}$.

We then obtain the primary attention probabilities $\alpha_{ti}^{\text{Prim}}$ by feeding e_{ti}^{Prim} into a softmax function. A primary context vector $\mathbf{c}_t^{\text{Prim}}$ which includes the information of only useful parts of handwriting input to describe the primary math symbol is computed by weighted summation of all annotation vectors:

$$\alpha_{ti}^{\text{Prim}} = \frac{\exp(e_{ti}^{\text{Prim}})}{\sum_{k=1}^{L} \exp(e_{tk}^{\text{Prim}})} \qquad \mathbf{c}_t^{\text{Prim}} = \sum_{i=1}^{L} \alpha_{ti}^{\text{Prim}} \mathbf{a}_i \tag{6}$$

The probability of each predicted symbol is computed by the primary context vector $\mathbf{c}_t^{\text{Prim}}$, current decoder state $\mathbf{h}_t$ and one-hot vector of previous output symbol $\mathbf{y}_{t-1}$ using the following equation:

$$P^{\text{Prim}}(\mathbf{y}_t) = g\left(\mathbf{W}_o h\left(\mathbf{E}\mathbf{y}_{t-1} + \mathbf{W}_h\mathbf{h}_t + \mathbf{W}_c\mathbf{c}_t^{\text{Prim}}\right)\right) \tag{7}$$

where g denotes a softmax activation function over all the math symbols in the symbol vocabulary, h denotes a maxout activation function and $\mathbf{E}$ denotes the embedding matrix. The current decoder state $\mathbf{h}_t$ is computed by PredGRU and the computation details will be explained in the next section. Let m denotes the dimension of embedding, $\mathbf{W}_o \in \mathbb{R}^{K \times \frac{m}{2}}$, $\mathbf{W}_h \in \mathbb{R}^{m \times n}$ and $\mathbf{W}_c \in \mathbb{R}^{m \times D}$.

The training loss of symbol recognition (the highlighted $\mathbf{y}_t$ in Figure 3) is computed as:

$$\mathcal{L}_{\text{Rec}} = -\sum_{t=1}^{T} \log P^{\text{Prim}}(w_t) \tag{8}$$

where w_t represents the ground-truth symbol at time step t.

2) Structural Analysis Loss: We employ a relation GRU (ReGRU), a relation attention model (ReAtt) and the primary attention model (PrimAtt) to deal with the structural analysis. We parse the mathematical expression tree by generating the sequential sub-tree structures (shown in Figure 2). Each sub-tree structure comprises a primary symbol node, a related symbol node and a relation node, where symbol nodes are denoted by the absolute spatial positions of the corresponding symbols and the relation node is denoted by the spatial relationship between the two symbol nodes.

We propose to use the output of primary attention model to compute the spatial position of primary symbol and use the output of relation attention model to compute the spatial position of related symbol. Similar to the computation of primary attention model, we compute the relation attention model as:

$$\tilde{\mathbf{h}}_t = \text{ReGRU}\left(\mathbf{c}_t^{\text{Prim}}, \hat{\mathbf{h}}_t\right) \tag{9}$$

$$\tilde{\mathbf{F}} = \mathbf{Q}_{\text{ReAtt}} * \sum_{l=1}^{t} \boldsymbol{\alpha}_l^{\text{Prim}} \tag{10}$$

$$e_{ti}^{\text{Re}} = \boldsymbol{\nu}_{\text{ReAtt}}^{\text{T}} \tanh(\mathbf{W}_{\text{ReAtt}}\tilde{\mathbf{h}}_t + \mathbf{U}_{\text{ReAtt}}\mathbf{a}_i + \tilde{\mathbf{U}}_f\tilde{\mathbf{f}}_i) \tag{11}$$

$$\alpha_{ti}^{\text{Re}} = \frac{\exp(e_{ti}^{\text{Re}})}{\sum_{k=1}^{L} \exp(e_{tk}^{\text{Re}})} \qquad \mathbf{c}_t^{\text{Re}} = \sum_{i=1}^{L} \alpha_{ti}^{\text{Re}} \mathbf{a}_i \tag{12}$$

Here, $\tilde{\mathbf{h}}_t$ is the output hidden state of relation GRU, it is computed by using the primary context vector $\mathbf{c}_t^{\text{Prim}}$ and the output state of PrimGRU $\hat{\mathbf{h}}_t$ as we believe the semantic information of related symbol should be extracted from the knowledge of primary symbol. We compute $\tilde{\mathbf{F}}$ by feeding the summation of primary attention probabilities $\sum_{l=1}^{t} \boldsymbol{\alpha}_l^{\text{Prim}}$ into a convolution layer $\mathbf{Q}_{\text{ReAtt}}$. We use $\tilde{\mathbf{F}}$ to tell the relation attention model that the related symbol must have been seen in the primary attention model. α_{ti}^{Re} is the attention probabilities of the related math symbol and $\mathbf{c}_t^{\text{Re}}$ is a related context vector which includes the information of only useful parts of handwriting traces to describe the related math symbol. The number of output channels of $\mathbf{Q}_{\text{ReAtt}}$ is set to q, the dimension of relation GRU is set to n and the dimension of relation attention is also set to n', $\boldsymbol{\nu}_{\text{ReAtt}}^{\text{T}} \in \mathbb{R}^{n'}$, $\mathbf{W}_{\text{ReAtt}} \in \mathbb{R}^{n' \times n}$, $\mathbf{U}_{\text{ReAtt}} \in \mathbb{R}^{n' \times D}$ and $\tilde{\mathbf{U}}_f \in \mathbb{R}^{n' \times q}$.

After having the primary context vector $\mathbf{c}_t^{\text{Prim}}$ and the related context vector $\mathbf{c}_t^{\text{Re}}$, we can recognize the relation node by feeding $\mathbf{c}_t^{\text{Prim}}$ and $\mathbf{c}_t^{\text{Re}}$ into a fully-connected layer as context vectors contain the spatial information of the two symbol nodes. We denote relation sequence as $\mathbf{R} = \{\mathbf{r}_1, \ldots, \mathbf{r}_T\}, \mathbf{r}_t \in \mathbb{R}^V$, V is the number of relations in the relation vocabulary. The probability of each $\mathbf{r}_t$ is then computed as follows:

$$P^{\text{Re}}(\mathbf{r}_t) = g\left(\mathbf{W}_{rc1}\mathbf{c}_t^{\text{Prim}} + \mathbf{W}_{rc2}\mathbf{c}_t^{\text{Re}}\right) \tag{13}$$

where g denotes a softmax activation function over all the math relations in the relation vocabulary.

The training loss of relation node (the highlighted $\mathbf{r}_t$ in Figure 3) is computed as:

$$\mathcal{L}_{\text{Re}} = -\sum_{t=1}^{T} \log P^{\text{Re}}(v_t) \tag{14}$$

where v_t represents the ground-truth relation at time step t.

As for the training loss of primary symbol node, we can get an alignment distance vector d_{ti}^{Prim} which denotes distance between primary symbol t and input point i (xy-coordinate of handwriting traces) by using the input of computing primary attention energy:

$$\mathbf{d}_{ti}^{\text{Prim}} = \tanh(\mathbf{W}_{\text{PrimAtt}}\hat{\mathbf{h}}_t + \mathbf{U}_{\text{PrimAtt}}\mathbf{a}_i + \hat{\mathbf{U}}_f\hat{\mathbf{f}}_i) \tag{15}$$

$$G_{ti}^{\text{Prim}} = \sigma(\boldsymbol{u}_{\text{PrimAli}}^{\text{T}}\mathbf{d}_{ti}^{\text{Prim}}) \tag{16}$$

Note that, $\mathbf{W}_{\text{PrimAtt}}$, $\mathbf{U}_{\text{PrimAtt}}$ and $\hat{\mathbf{U}}_f$ are the same parameters used in Eq. (5), $\boldsymbol{u}_{\text{PrimAtt}}^{\text{T}} \in \mathbb{R}^{n'}$. The alignment distance vector

$\mathbf{d}_{ti}^{\text{Prim}}$ is then subject to a binary classification loss (cross-entropy) to obtain the alignment prediction G_{ti}^{Prim}.

The training loss of primary symbol node (the highlighted $\mathbf{ps}_t$ in Figure 3) is computed as:

$$\mathcal{L}_{\text{PrimAli}} = -\sum_{t=1}^{T}\sum_{i=1}^{L}[\bar{G}_{ti}^{\text{Prim}}\log(G_{ti}^{\text{Prim}}) + (1-\bar{G}_{ti}^{\text{Prim}})\log(1-G_{ti}^{\text{Prim}})] \tag{17}$$

where $\bar{G}_{ti}^{\text{Prim}}$ denotes the ground-truth of primary symbol node. $\bar{G}_{ti}^{\text{Prim}}$ is 1 if point i belongs to the primary symbol node t, otherwise 0.

As for the training loss of related symbol node, we can also get the alignment prediction G_{ti}^{Re} by using the input of computing relation attention energy:

$$\mathbf{d}_{ti}^{\text{Re}} = \tanh(\mathbf{W}_{\text{ReAtt}}\tilde{\mathbf{h}}_t + \mathbf{U}_{\text{ReAtt}}\mathbf{a}_i + \tilde{\mathbf{U}}_f\tilde{\mathbf{f}}_i) \tag{18}$$

$$G_{ti}^{\text{Re}} = \sigma(\boldsymbol{u}_{\text{ReAli}}^{\text{T}}\mathbf{d}_{ti}^{\text{Re}}) \tag{19}$$

where $\mathbf{W}_{\text{ReAtt}}$, $\mathbf{U}_{\text{ReAtt}}$ and $\tilde{\mathbf{U}}_f$ are the same parameters used in Eq. (11), $\boldsymbol{u}_{\text{ReAtt}}^{\text{T}} \in \mathbb{R}^{n'}$. and we compute the training loss of related symbol node (the highlighted $\mathbf{rs}_t$ in Figure 3) as:

$$\mathcal{L}_{\text{ReAli}} = -\sum_{t=1}^{T}\sum_{i=1}^{L}[\bar{G}_{ti}^{\text{Re}}\log(G_{ti}^{\text{Re}}) + (1-\bar{G}_{ti}^{\text{Re}})\log(1-G_{ti}^{\text{Re}})] \tag{20}$$

where $\bar{G}_{ti}^{\text{Re}}$ denotes the ground-truth of related symbol node. $\bar{G}_{ti}^{\text{Re}}$ is 1 if point i belongs to the related symbol node t, otherwise 0.

The final training loss of structural analysis is the summation of training loss of relation node, primary symbol node and related symbol node because the whole tree structure is composed of these three parts:

$$\mathcal{L}_{\text{Struc}} = \mathcal{L}_{\text{Re}} + \mathcal{L}_{\text{PrimAli}} + \mathcal{L}_{\text{ReAli}} \tag{21}$$

We finally utilize the prediction GRU (shown in Figure 3) to compute the current decoder output state $\mathbf{h}_t$ for symbol recognition (Eq. (7)) and for next decoding step:

$$\mathbf{c}_t = \text{Concat}(\mathbf{c}_t^{\text{Prim}}, \mathbf{c}_t^{\text{Re}}) \tag{22}$$

$$\mathbf{h}_t = \text{PredGRU}\left(\mathbf{c}_t, \tilde{\mathbf{h}}_t\right) \tag{23}$$

where $\mathbf{c}_t$ is the concatenation of the two context vectors $\mathbf{c}_t^{\text{Prim}}$ and $\mathbf{c}_t^{\text{Re}}$.

IV. Implementation Details

A. Training

The training objective of SRD is to minimize the symbol recognition loss (Eq. (8)) and structural analysis loss (Eq. (21)) simultaneously. The objective function is shown as follows:

$$O = \lambda_1\mathcal{L}_{\text{Rec}} + \lambda_2\mathcal{L}_{\text{Struc}} \tag{24}$$

In our experiments, we set $\lambda_1 = \lambda_2 = 1$ as we believe symbol recognition and structural analysis are equally important.

To be fairly comparable with state-of-the-art LaTeX string decoder based methods, we use the same encoder architecture employed in TAP model [9], [29]. The encoder consists of 4 bidirectional GRU layers. Each layer has 256 forward and 256 backward GRU units. The pooling over time is applied to the top 2 bidirectional GRU layers.

As for the proposed SRD, PrimGRU, ReGRU and PredGRU all employ unidirectional GRU layers with each layer containing 256 forward GRU units. The primary attention dimension and the relation attention dimension are both set to 512. The kernel size of convolution filter $\mathbf{Q}_{\text{PrimAtt}}$ is set to (121×1) and the number of output channels is 256. The kernel size of convolution filter $\mathbf{Q}_{\text{ReAtt}}$ is set to (121×1) and the number of output channels is 256. The embedding dimension is set to 256.

We utilize the adadelta algorithm [38] with $\varepsilon = 10^{-8}$ for optimization. The experiments are all implemented with Theano 1.0 [39] and an NVIDIA Tesla M40 24G GPU.

B. Testing

In the testing stage, we aim to generate a sequence of math symbols $\mathbf{y}$ for symbol recognition and a sequence of sub-tree structures $(\mathbf{ps}, \mathbf{rs}, \mathbf{r})$ for structural analysis.

Firstly, at each decoding step, the SRD outputs the most likely math symbol $\mathbf{y}$ given the input handwriting traces:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \log P^{\text{Prim}}(\mathbf{y}|\mathbf{x}) \tag{25}$$

The decoding procedure ends when the symbol generator outputs the end-of-decoding token "</s>".

Secondly, we get the absolute spatial positions of primary symbol node $\mathbf{ps}$ by processing $G_{ti}^{\text{Prim}}, 1 \le i \le L$ at decoding step t. For online handwritten mathematical expression recognition, the absolute spatial positions of symbol node are the stroke indices of input traces. To get the primary symbol node, considering that one stroke can only be aligned to one primary math symbol, we average G_{ti}^{Prim} of all points for every stroke and obtain $\widehat{G}_{tn}^{\text{Prim}}, 1 \le n \le N_{\text{stroke}}$, where N_{stroke} is the total number of strokes. The n-th stroke is aligned to $\widehat{t}$-th primary symbol node, where $\widehat{t} = \arg\max_t \widehat{G}_{tn}^{\text{Prim}}$.

Thirdly, to get the related symbol node $\mathbf{rs}$, at each decoding step, we compare G_{ti}^{Re} with alignments of previous determined primary symbol nodes G_{ti}^{Prim} and choose the primary symbol node which has the most likely alignments as G_{ti}^{Re} to be the current related symbol node.

Finally, after determining the primary symbol node and related symbol node, we feed the primary context vector and related context vector into the relation prediction fully-connected layer to output the most likely math relation $\mathbf{r}$:

$$\hat{\mathbf{r}} = \arg\max_{\mathbf{r}} \log P^{\text{Re}}(\mathbf{r}|\mathbf{x}) \tag{26}$$

C. Dataset

The experiments are conducted on CROHME competition dataset [40], which is currently the most widely used public dataset for online handwritten mathematical expression recognition. The CROHME training set contains 8836 handwritten mathematical expressions. There are totally 101 math symbol classes and 6 math relations (above, below, right, inside, superscript, subscript). Most researches evaluate their proposed methods on CROHME 2014 [41] test set, which contains

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2020.3011316, IEEE Transactions on Multimedia

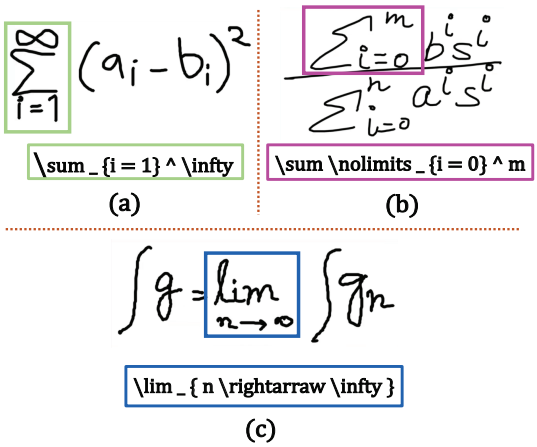

Fig. 4. (a) upper limit and lower limit are on the above and below of "$\sum$"; (b) upper limit and lower limit are on the superscript and subscript of "$\sum$"; (c) condition is on the below of "lim".

986 handwritten mathematical expressions. We also test the generalization capability of SRD on CROHME 2016 [42] and CROHME 2019 [43] test set, which are newly collected and labeled by the competition organizers. There are totally 1147 expressions on CROHME 2016 set and 1199 expressions on CROHME 2019 set with the symbol classes and relation classes unchanged.

V. Experiments

In this section, we will show the effectiveness of the proposed SRD for online handwritten mathematical expression recognition by answering the three questions.

- Q1 Compared with LaTeX string decoders, does SRD have properties that are especially valuable for mathematical expression recognition?
- Q2 Can SRD outperform LaTeX string decoders and other state-of-the-art methods?
- Q3 How does the SRD generate the math tree structures step by step?

A. Valuable Properties of SRD (Q1)

1) Robustness of relation prediction: Firstly, we show a property of SRD that it can better distinguish the spatial relationships between math symbols than LaTeX string decoders. We believe so because in SRD, the spatial relationships are classified based on the spatial information of current output math symbols, but in LaTeX string decoders the spatial relationships are converted into LaTeX words that are computed based on the embedding vectors of previous output words. For example, in mathematical expressions, there are some special symbols like "$\sum$" and "$\int$" that will have upper limit and lower limit. As illustrated in Figure 4, the upper limit of "$\sum$" can be on the above direction or superscript direction. In LaTeX language, a word called "nolimits" is used to help distinguish that the upper limit is on the above direction or superscript direction of "$\sum$". But "nolimits" is a rare word in CROHME training LaTeX texts, therefore, during testing the word "nolimits" can never be generated, leading to many incorrectly recognized examples. Yet SRD can successfully distinguish the above and the superscript relation related with "$\sum$" or "$\int$" as SRD depends on spatial information to classify relation. For another example, in CROHME training set, the relations between math symbol "lim" and its condition are always below but are all inaccurately labelled as subscript. However, during testing, the SRD can still recognize the below relation as the below relation has been learned enough when training mathematical expressions having fraction operations, regardless of the errors coming from wrong labelled data.

2) Grammatical output: Secondly, as for LaTeX string decoder, the decoder generates the output string without any restriction. In bad situations, the decoder will output strings that disobey LaTeX grammar. For example, in LaTeX grammar, strings must have a pair of "{" and "}" following math structures like "x $\wedge$ { 2 }", but string decoder will generate ungrammatical strings like "x $\wedge$ { 2" that omits one "}" and "x $\wedge$ { 2 } }" that has one extra "}". These ungrammatical strings can not even be displayed by using a LaTeX tool, like the recognition string of the first mathematical expression in Figure 5. However, as for SRD, the decoder generates the tree structures of mathematical expressions, we then convert the tree structures into LaTeX string under the restriction of LaTeX grammar. This approach helps alleviate the problem of ungrammatical output as we can ensure that a pair of "{" and "}" must appear after math structures during converting.

3) Object-level metric: Thirdly, LaTeX string decoder based approaches evaluate their models on expression level, i.e., the percentage of predicted mathematical expression LaTeX strings matching the ground truth. They do so because the participating systems in all of the CROHME competitions are ranked by expression recognition rates (ExpRate). But, intuitively, it is inappropriate to evaluate an expression recognition system only at the expression level as the expression-level metric varies seriously. The LaTeX string decoder based approaches propose to use word error rate (WER) [44] as the object-level metric. We argue that the WER metric (including substitutions, deletions and insertions) only shows the distance between target string and output string, it can not correctly show the distance between target tree structure and output tree structure. While SRD aims to output the tree structure, and we can get more valuable object-level metrics showing distance between two trees like symbol recognition rate and structure recognition rate as shown in Table I. These tree-structured object-level metrics are official and now well-established metrics of the competition [43].

B. Performance Comparison (Q2)

In Table I, we compare the SRD based encoder-decoder system with the state-of-the-arts. We list the best 3 systems in CROHME 2014 competition [40] using only official dataset and 4 LaTeX string decoder based encoder-decoder systems. We compare systems not only by "ExpRate" but also those with at most three object-level errors ($\leq$ 1, $\leq$ 2, $\leq$ 3). Unlike traditional methods which can output a math tree structure, LaTeX string decoder based systems only output a mathematical LaTeX string. "LaTeX Match" shows the rate

TABLE I
EVALUATION OF MATHEMATICAL EXPRESSION RECOGNITION SYSTEMS ON CROHME 2014 TEST SET (IN %). "SEGMENTS" DENOTES SYMBOL SEGMENTATION, "SEG+CLASS" DENOTES DETECTION OF SYMBOLS WITH CORRECT CLASSIFICATION, "TREE RELS." DENOTES TREE RELATIONS WHICH REQUIRES TWO RELATED SYMBOLS WHICH ARE CORRECTLY SEGMENTED AND IN THE RIGHT RELATIONSHIP, "REC." DENOTES RECALL, "PREC." DENOTES PRECISION, "*" DENOTES THE REPORTED RESULT IS OF AN ENSEMBLE SYSTEM COMBINING 5 MODELS TRAINED WITH DIFFERENT PARAMETERS INITIALIZATION.

System	Segments		Seg+Class		Tree Rels.		ExpRate	$\leq$ 1 error	$\leq$ 2 errors	$\leq$ 3 errors	LaTeX Match
	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.					
I	93.3	90.7	86.6	84.2	84.2	82.0	37.2	44.2	47.3	50.2	–
VI	83.1	85.4	69.7	71.7	66.8	74.8	25.7	33.2	35.9	37.3	–
VII	89.4	86.1	76.5	73.7	71.8	71.7	26.1	33.9	38.5	40.0	–
WYGIWYS[10]	–	–	–	–	–	–	–	–	–	–	35.9
PAL[28]	–	–	–	–	–	–	–	–	–	–	39.7
WAP[8]	–	–	–	–	–	–	–	–	–	–	39.4
TAP[9]	–	–	–	–	–	–	–	–	–	–	46.9
SRD	**95.0**	**95.6**	**87.7**	**88.2**	**89.3**	**90.0**	**48.9**	**57.9**	**62.1**	**64.2**	**50.6**
SRD*	**95.7**	**96.2**	**88.8**	**89.3**	**90.6**	**91.3**	**53.6**	**62.4**	**66.4**	**68.6**	**55.3**

that output LaTeX strings match the corresponding ground-truth LaTeX strings. Note that, although both "ExpRate" and "LaTeX Match" are expression-level metrics, the "ExpRate" is a stricter metric as it considers the symbol segmentation errors while "LaTeX Match" not. As we can see in Table I, previous string decoder based systems like WYGIWYS, PAL, WAP and TAP only publish the "LaTeX Match" results, and they use "LaTeX Match" results to compare with "ExpRate" results of traditional methods. The comparison is unfair. We show through the results of SRD that without considering segmentation errors, "LaTeX Match" results have about 2% improvement compared with "ExpRate" results. Considering this gap between "ExpRate" and "LaTeX Match", the string decoder based encoder-decoder models like WYGIWYS, PAL and WAP do not largely outperform traditional grammar tree methods (e.g., System I).

As shown in Table I, TAP is currently the best published string decoder based model for online handwritten mathematical expression recognition. To prove the superiority of SRD compared to string decoder, SRD uses the same encoder architecture and optimization strategy as TAP. Therefore, by comparing TAP and SRD we can clearly see the improvement of SRD. To be fairly comparable with TAP, we also convert the output math tree structure into the LaTeX string, regardless of the segmentation error. By comparing SRD with TAP, it is clear to see, a tree-structured decoder can achieve a significant improvement (from 46.9% to 50.6%, nearly 4% absolutely on single model) for mathematical expression recognition than a string decoder. Also, SRD has a property that it can be evaluated on object level. The comparison between SRD and the best system in CROHME 2014 competition (system I) shows that SRD significantly outperforms traditional methods.

To test the generalization of the improvement of SRD compared with string decoder and more recent grammar tree algorithms, we compare SRD with TAP and competition teams using only official training dataset on CROHME 2016 test set as shown in Table II. We can see the improvement of SRD compared with string decoder is larger (5.3%, from 41.3% to 46.6%) on a harder test set. The team Wiris was awarded the first place on CROHME 2016 competition, but it used a

TABLE II
EVALUATION OF MATHEMATICAL EXPRESSION RECOGNITION SYSTEMS ON CROHME 2016 TEST SET (IN %). "EXP" DENOTES "EXPRATE", "MATCH" DENOTES "LATEX MATCH". "*" DENOTES THE REPORTED RESULT IS OF AN ENSEMBLE SYSTEM COMBINING 5 MODELS TRAINED WITH DIFFERENT PARAMETERS INITIALIZATION.

System	Seg+Class		Tree Rels.		Exp	Match
	Rec.	Prec.	Rec.	Prec.		
Wiris	**90.8**	**91.3**	**92.0**	**92.6**	**49.6**	–
Tokyo	86.1	87.6	84.4	86.0	43.9	–
São Paulo	86.3	88.3	83.5	86.3	33.4	–
Nantes	87.2	82.4	76.3	71.7	13.3	–
TAP	–	–	–	–	–	41.3
SRD	87.4	88.1	88.8	89.8	44.9	**46.6**
SRD*	88.1	88.8	89.1	90.1	48.8	**50.4**

Wikipedia formula corpus, consisting of more than 592,000 formulas, to train a strong language model. While we directly use the SRD system trained on CROHME 2014 training set (only 8,836 formulas) without any further tuning or other language models and also achieve a good performance.

TABLE III
EVALUATION OF MATHEMATICAL EXPRESSION RECOGNITION SYSTEMS ON CROHME 2019 TEST SET (IN %). "EXP" DENOTES "EXPRATE", "MATCH" DENOTES "LATEX MATCH". "*" DENOTES THE REPORTED RESULT IS OF AN ENSEMBLE SYSTEM COMBINING 5 MODELS TRAINED WITH DIFFERENT PARAMETERS INITIALIZATION.

System	Seg+Class		Tree Rels.		Exp	Match
	Rec.	Prec.	Rec.	Prec.		
TAP	–	–	–	–	–	41.7
SRD	**87.4**	**87.3**	**89.0**	**89.1**	**43.9**	**45.9**
SRD*	**88.4**	**88.5**	**90.4**	**90.6**	**48.5**	**50.6**

We also evaluate SRD on CROHME 2019 [43] as listed in Table III. We only compare SRD with TAP using official training dataset to clearly see the improvement of SRD in comparison to string decoder (4.2%, from 41.7% to 45.9%). We do not compare SRD with the competition teams as they all use additional training set or other synthetic data. The

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2020.3011316, IEEE Transactions on Multimedia

"ExpRate" is a little lower on CROHME 2019 because there are many expressions containing strokes which only have one point, increasing the difficulty of symbol segmentation. By watching "LaTeX Match", the proposed SRD still significantly outperforms string decoder.

Finally, by comparing SRD to SRD* (ensemble of 5 SRD models) and traditional methods on all test sets in Table I, II, III, it is also interesting to find that the ensemble SRD* systems only achieve slightly better performance at object-level metric than SRD (usually less than 1%), but it achieves much better performance at expression-level (usually near 5%). Therefore, we believe that the ensemble method [45] can greatly help improve the expression rate.

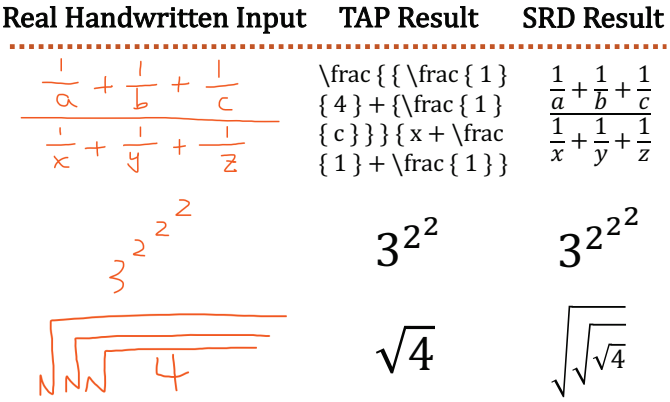

Fig. 5. Comparison of TAP and SRD on real handwritten mathematical expressions with more complex math structures that have never been seen on CROHME training set.

In Figure 5, we compare the recognition results of TAP and SRD on three math formulas with complex structures to further show the generalization capability of SRD on tree structures. For example, the superscript structure has appeared in CROHME training set many times, like formula "x ∧ { 2 }". During real testing, we specifically draw a formula with nested superscript structures, like "x ∧ { 2 ∧ { 2 ∧ { 2 } } }". The nested superscript structure has never been seen during training, but the superscript structure has been learned many times. From the recognition results of TAP and SRD, we can infer that string decoder has no understanding of math tree structure as it fails when dealing with nested structures, it totally relies on the implicit language model embedded in string decoder to generate LaTeX strings. But SRD learns the math tree structures well, as it performs well when dealing with nested fraction structures, nested superscript structures and nested sqrt structures.

C. Visualization of Decoding Procedure (Q3)

Figure 6 shows two examples to illustrate how SRD recognizes the math tree structure. As we can see in the right part of the two examples, SRD generates the sequential subtree structures step by step with parent node denoting spatial relationship, left child node denoting primary symbol and right child node denoting related symbol. The left images show the alignment generated by alignment prediction G_{ti}^{Prim} and G_{ti}^{Re},

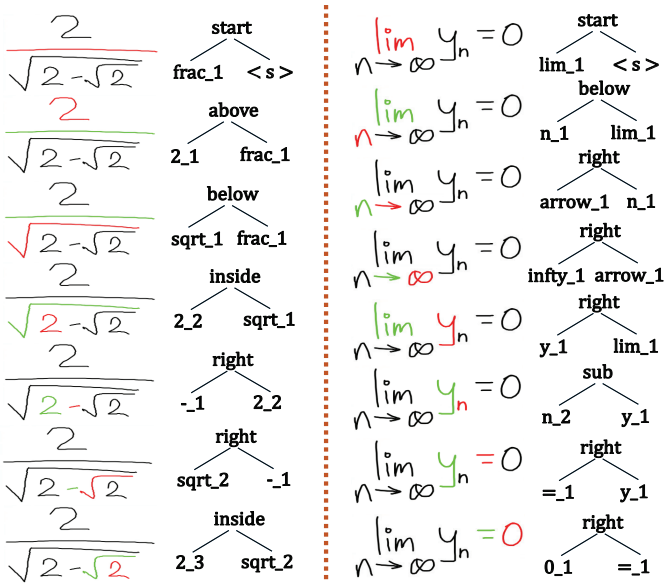

Fig. 6. Two examples showing decoding procedure through stroke based alignment of primary symbol nodes and related symbol nodes.

with the red color representing the alignment of primary math symbols and the green color representing the alignment of related math symbols. The processing method is introduced in Section IV-B.

The alignments of primary math node and related math node and the analysis of math tree structure correspond well to human intuition. More specifically, take the right mathematical expression as an example, when SRD outputs the primary math symbol "y", it successfully detects that the related symbol of "y" is "lim" and generates the "right" spatial relationship. While for LaTeX string decoders, when a string decoder outputs the primary math symbol "y", it mainly depends on the previous symbol in LaTeX string, i.e. "∞", which is less reasonable than SRD does.

VI. Conclusion and Future Work

In this study we propose a novel sequential relation decoder which is the first tree structure based decoder for mathematical expression recognition. Through detailed comparisons with state-of-the-art string decoders and experimental analysis, we clearly demonstrate the advantages of sequential relation decoder for tree-structured language researches. As for future work, we will implement the sequential relation decoder for offline mathematical expression recognition and investigate its application on other tree-structured languages.

Acknowledgment

This work was supported in part by the National Key R&D Program of China under Grant 2017YFB1002202, the National Natural Science Foundation of China under Grants 61671422 and U1613211, the Fundamental Research Funds for the Central Universities, the MOE-Microsoft Key Laboratory of USTC, and the iFLYTEK-Surrey Joint Research Centre on Artificial Intelligence.

REFERENCES

[1] R. Zanibbi, D. Blostein, and J. R. Cordy, "Recognizing mathematical expressions using tree transformation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 11, pp. 1455–1467, 2002.

[2] A. Belaid and J.-P. Haton, "A syntactic approach for handwritten mathematical formula recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 1, pp. 105–111, 1984.

[3] K.-F. Chan and D.-Y. Yeung, "Mathematical expression recognition: a survey," *International Journal on Document Analysis and Recognition*, vol. 3, no. 1, pp. 3–15, 2000.

[4] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[5] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *International Conference on Acoustics, Speech and Signal Processing*, 2016, pp. 4945–4949.

[6] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3156–3164.

[7] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International Conference on Machine Learning*, 2015, pp. 2048–2057.

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

[9] J. Zhang, J. Du, and L. Dai, "A gru-based encoder-decoder approach with attention for online handwritten mathematical expression recognition," in *International Conference on Document Analysis and Recognition*, vol. 1, 2017, pp. 902–907.

[10] Y. Deng, A. Kanervisto, J. Ling, and A. M. Rush, "Image-to-markup generation with coarse-to-fine attention," in *International Conference on Machine Learning*, 2017, pp. 980–989.

[11] F. Álvaro, J.-A. Sánchez, and J.-M. Benedí, "Recognition of on-line handwritten mathematical expressions using 2d stochastic context-free grammars and hidden markov models," *Pattern Recognition Letters*, vol. 35, pp. 58–67, 2014.

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

[13] F. Álvaro, J.-A. Sánchez, and J.-M. Benedí, "An integrated grammar-based approach for mathematical expression recognition," *Pattern Recognition*, vol. 51, pp. 135–147, 2016.

[14] T. H. Rhee and J. H. Kim, "Efficient search strategy in structural analysis for handwritten mathematical expression recognition," *Pattern Recognition*, vol. 42, no. 12, pp. 3192–3201, 2009.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[16] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 6645–6649.

[17] L. Hu and R. Zanibbi, "Mst-based visual parsing of online handwritten mathematical expressions," in *International Conference on Frontiers in Handwriting Recognition*, 2016, pp. 337–342.

[18] A. Kosmala and G. Rigoll, "On-line handwritten formula recognition using statistical methods," in *International Conference on Pattern Recognition*, vol. 2, 1998, pp. 1306–1308.

[19] N. S. Hirata and F. D. Julca-Aguilar, "Matching based ground-truth annotation for online handwritten mathematical expressions," *Pattern Recognition*, vol. 48, no. 3, pp. 837–848, 2015.

[20] K.-F. Chan and D.-Y. Yeung, "Error detection, error correction and performance evaluation in on-line mathematical expression recognition," *Pattern Recognition*, vol. 34, no. 8, pp. 1671–1684, 2001.

[21] S. MacLean and G. Labahn, "A new approach for recognizing handwritten mathematics using relational grammars and fuzzy sets," *International Journal on Document Analysis and Recognition*, vol. 16, no. 2, pp. 139–163, 2013.

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

[23] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *International Conference on Acoustics, Speech and Signal Processing*, 2016, pp. 4960–4964.

[24] K. Cho, A. Courville, and Y. Bengio, "Describing multimedia content using attention-based encoder-decoder networks," *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 1875–1886, 2015.

[25] L. Li, S. Tang, Y. Zhang, L. Deng, and Q. Tian, "Gla: Global–local attention for image description," *IEEE Transactions on Multimedia*, vol. 20, no. 3, pp. 726–737, 2017.

[26] N. Zhao, H. Zhang, R. Hong, M. Wang, and T.-S. Chua, "Videowhisper: Toward discriminative unsupervised video feature learning with attention-based recurrent neural networks," *IEEE Transactions on Multimedia*, vol. 19, no. 9, pp. 2080–2092, 2017.

[27] J. Zhang, J. Du, and L. Dai, "Multi-scale attention with dense encoder for handwritten mathematical expression recognition," in *International Conference on Pattern Recognition*, 2018, pp. 2245–2250.

[28] J.-W. Wu, F. Yin, Y.-M. Zhang, X.-Y. Zhang, and C.-L. Liu, "Image-to-markup generation via paried adversarial learning," in *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2018.

[29] J. Zhang, J. Du, and L. Dai, "Track, attend and parse (tap): An end-to-end framework for online handwritten mathematical expression recognition," *IEEE Transactions on Multimedia*, vol. 21, no. 1, pp. 221–233, 2019.

[30] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, vol. 1, 2015, pp. 1556–1566.

[31] T. Zhang, H. Mouchère, and C. Viard-Gaudin, "Tree-based blstm for mathematical expression recognition," in *International Conference on Document Analysis and Recognition*, 2017, pp. 914–919.

[32] X. Chen, C. Liu, and D. Song, "Tree-to-tree neural networks for program translation," in *Advances in Neural Information Processing Systems*, 2018.

[33] R. Zanibbi, H. Mouchere, and C. Viard-Gaudin, "Evaluating structural pattern recognition for handwritten math via primitive label graphs," in *Document Recognition and Retrieval*, 2013.

[34] Y. Bengio, P. Simard, P. Frasconi *et al.*, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[35] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[36] X.-Y. Zhang, F. Yin, Y.-M. Zhang, C.-L. Liu, and Y. Bengio, "Drawing and recognizing chinese characters with recurrent neural network," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[37] Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li, "Modeling coverage for neural machine translation," *arXiv preprint arXiv:1601.04811*, 2016.

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

[39] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: A cpu and gpu math compiler in python," in *Proc. 9th Python in Science Conf*, 2010, pp. 1–7.

[40] H. Mouchère, R. Zanibbi, U. Garain, and C. Viard-Gaudin, "Advancing the state of the art for handwritten math recognition: the crohme competitions, 2011–2014," *International Journal on Document Analysis and Recognition*, vol. 19, no. 2, pp. 173–189, 2016.

[41] H. Mouchere, C. Viard-Gaudin, R. Zanibbi, and U. Garain, "Icfhr 2014 competition on recognition of on-line handwritten mathematical expressions (crohme 2014)," in *International Conference on Frontiers in Handwriting Recognition*, 2014, pp. 791–796.

[42] H. Mouchère, C. Viard-Gaudin, R. Zanibbi, and U. Garain, "ICFHR2016 CROHME: Competition on recognition of online handwritten mathematical expressions," in *International Conference on Frontiers in Handwriting Recognition*, 2016, pp. 607–612.

[43] M. Mahdavi, R. Zanibbi, H. Mouchère, C. Viard-Gaudin, and U. Garain, "Icdar 2019 crohme + tfd: Competition on recognition of handwritten mathematical expressions and typeset formula detection," 2019.

[44] D. Klakow and J. Peters, "Testing the correlation of word error rate and perplexity," *Speech Communication*, vol. 38, no. 1-2, pp. 19–28, 2002.

[45] T. G. Dietterich, "Ensemble methods in machine learning," in *International workshop on multiple classifier systems*, 2000, pp. 1–15.