# Using BLSTM for interpretation of 2-D languages

## Case of handwritten mathematical expressions

Ting Zhang, Harold Mouchère, Christian Viard-Gaudin

Dans Document numérique 2016/2-3 Vol. 19 , pages 135 à 157
Éditions JLE

**ISSN 1279-5127**
**ISBN 9782746247956**
**Date de mise en ligne : 10/01/2017**

# Using BLSTM for interpretation of 2-D languages

## Case of handwritten mathematical expressions

**Ting Zhang, Harold Mouchère, Christian Viard-gaudin**

*LUNAM/IRCCyN/IVC - UMR CNRS 6597*
*Université de Nantes*
*Nantes, France*
*ting.zhang1@etu.univ-nantes.fr*
*{harold.mouchere,christian.viard-gaudin}@univ-nantes.fr*

ABSTRACT. *In this work, we study how to extend the capability of BLSTM networks with CTC to process data which are not only text strings but graphical two-dimensional languages such as handwritten mathematical expressions. An online math expression is a sequence of strokes which is later labeled by BLSTM network. Besides normal math symbols, we introduce 6 additional specific labels assigned to each of the different possible spatial relationships that exist between sub-expressions. The output of BLSTM network with CTC is a sequence of labels. Our aim is to build a two-dimensional (2-D) expression from this sequence of labels. CTC technology is a good choice for sequence transcription tasks but does not provide the alignment between the inputs and the target labels. In our case, we need the labels of strokes in the building process. A local CTC is proposed to solve this problem. As a result, BLSTM network is able to perform at the same time the symbol recognition task, the segmentation task and the relationship recognition task, which is a new perspective for the mathematical expression domain.*

RÉSUMÉ. *Nous proposons une extension de l'utilisation classique des réseaux de type BLSTM pour leur permettre de traiter des données provenant de langages graphiques bidimensionnels tels que les formules mathématiques manuscrites. La solution proposée repose sur un parcours respectant l'ordre temporel des traits. Il en résulte une séquence alternant les étiquettes de symboles et les étiquettes des relations spatiales. Dans le cas des expressions purement linéaires (1-D), nous utilisons l'étiquette « Right » pour permettre la segmentation entre les symboles. Pour une extension au cas des expressions véritablement bidimensionnelles (2-D), nous utilisons autant de nouvelles étiquettes qu'il y a de relations spatiales différentes entre les sous-expressions. Les BLSTM sont appris en utilisant la stratégie CTC que nous avons adaptée pour fournir un étiquetage aligné avec les traits de l'encre. Il en résulte que les réseaux BLSTM*

Document numérique – n° 2-3/2016, 135-157

*permettent de résoudre à la fois la tâche de reconnaissance de symboles et celle de segmentation. Une telle approche est nouvelle dans le domaine de la reconnaissance des expressions mathématiques.*

KEYWORDS: *handwritten mathematical expression, online handwriting, recurrent neural network, BLSTM, local CTC.*

MOTS-CLÉS : *reconnaissance d'expressions mathématiques, écriture manuscrite, réseau récurrent, BLSTM, local CTC.*

DOI:10.3166/DN.19.2-3.135-157 



## 1. Introduction

Online Handwritten Mathematical Expressions (MEs) recognition has been an active research field for years, especially after the rapid development of touch screen devices. There are two main sources of handwritten ME: offline (images of manuscript documents or video of blackboards) and online. We focus on online documents which come from sensitive screens (interactive whiteboards, smaller touch screens or tablets, etc.). An online document consists of one or more strokes which are sequences of points sampled from the trajectory of the writing tool between a pen-down and a pen-up. On-line handwritten ME recognition involves the automatic interpretation of these temporal sequence of 2-D points into structured sets of symbols. To realize these, both a large set of symbol classes and symbol arrangements in a two-dimensional layout need to be recognized. ME recognition still exhibits a big challenge to researchers.

This research domain has been boosted by the Competition on Recognition of Handwritten Mathematical Expressions (CROHME) (Mouchère *et al*., 2016), which began as part of the International Conference on Document Analysis and Recognition (ICDAR) in 2011. It provides a platform for researchers to test their methods and compare them, and then facilitate the progress in this field. It attracts increasing participation of research groups from all over the world. In this work, the provided data and tools will be used and results will be compared to participants.

Generally, ME recognition involves three interdependent tasks (Zanibbi, Blostein, 2012): (1) Symbol segmentation, which consists in grouping strokes that belong to the same symbol; (2) symbol recognition, the task of labeling the symbol to assign each of them a symbol class; (3) structural analysis, its goal is to identify spatial relations between symbols and with the help of a grammar to produce a mathematical interpretation. These three problems can be solved sequentially or jointly. In the first case, the errors from the previous stage will be propagated to the next stage. The alternative solutions run these three tasks concurrently and aim at making a final decision of the formula using global information (Awal *et al*., 2014; Álvaro *et al*., 2014b; 2016). These three problems are highly interdependent by nature. For example, human beings recognize symbols with the help of structure, and vice versa. Thus, the integrated

solutions seem more reasonable and indeed exhibit better performances than the sequential ones.

Current solutions process each task separately but need to consider context. Context is included to classical classifier (MLP, SVM, etc.) with specific features or language model (grammar). Bidirectional Long Short-Term Memory (BLSTM) network naturally takes this context into account because it can access the contextual information from both the future and the past in an unlimited range. The advanced recurrent neural network BLSTM has proved to outperform other classifiers in tasks like handwritten text recognition (Graves *et al*., 2009), handwritten math symbol recognition (Álvaro *et al*., 2013; 2014a), printed text recognition (Ray *et al*., 2015) and keyword spotting (Frinken *et al*., 2014). However, the move from text recognition to mathematical expression recognition is far from being straightforward since ME have a 2 dimensional structure. In this study, we explored ME recognition using BLSTM models, not simply as a symbol classifier exploiting a pre-segmented fragment as proposed in (Álvaro *et al*., 2014a), but as a system able to label a global sequence, solving at the same time the symbol segmentation, their recognition and the spatial relation recognition tasks.

It is possible to describe a ME using a Stroke Label Graph (SLG) in which nodes represent strokes, whereas edges encode either segmentation information or layout information. The detailed description of this structure will be found in Section 3.1. Given an handwritten expression which is a sequence of strokes, we first adopt the strong sequence classifier BLSTM with a local CTC (connectionist temporal classification) output layer to label the sequence. The next step is to build the SLG from the outputted sequence of labels. During the building process, the alignment between labels and strokes is required but this information is not available from classical CTC. Thus we propose a local CTC mapping to extend the capability of BLSTM networks. The proposed solution works well on linear expressions and part of 2-D expressions. For some 2-D expressions, it fails because there exists a limitation in our method. We give more explanations in the following content.

The remainder of the paper is organized as follows. Section 2 introduces BLSTM architecture and CTC technology briefly. Section 3 describes the representation of ME, including the concept of primitive label graph and how to build it. Section 4 presents the detailed implementation of the entire proposed solution. Features (local and contextual) for on-line ME are showed in Section 5. In Section 6, we report the experiments and results. Finally, conclusions and future works are presented.

## 2. Related works—BLSTM and CTC

### *2.1. BLSTM*

Recurrent neural networks (RNNs) can use contextual information and therefore are suitable for sequence labeling (Graves *et al*., 2012). In order to visualize the RNN and understand how it operates, we unfolded it along the input sequence and illustrated

a part of the unfolded one in Figure 1. Here, each node represents a layer of network

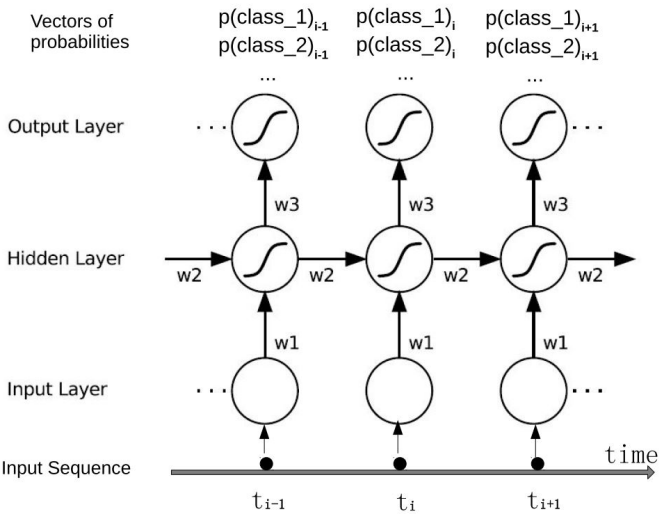



*Figure 1. An unfolded single-directional recurrent network*

units at a single time-step. The input at a single time-step is a feature vector; with regard to the whole process, the input is a time sequence of feature vectors. The output at a single time-step is a probability vector of which each element is the probability of belonging to some class; the overall output is a sequence of probability vectors. As can be seen, the output at each step depends on both the current input and the information from the previous time-step. The same weights (w1, w2, w3) are reused at every time-step. Training a RNN requires the ground-truth for each time-step to compute the error. During recognition, the network only outputs local classifications, if needed, some form of post-processing can be performed.

An important benefit of RNNs is their ability to use contextual information. Unfortunately, for standard RNN architectures, the range of context that can be accessed is quite limited. Long short-term memory (LSTM) (Hochreiter, Schmidhuber, 1997) has the ability to access long range of context. An LSTM network is similar to a standard RNN, except that the summation units in the hidden layer are replaced by memory blocks, as shown in Figure 2. Each block contains one or more self-connected memory cells and three multiplicative units (the input, output and forget gates). The three gates collect activation from inside and outside the block and control the activation of the cell via multiplications. The input and output gates multiply the input and output of the cell while the forget gate multiplies the cell's previous state. The only output from the block to the rest of the network emanates from the output gate multiplication.

Standard RNNs process sequences in temporal order, as presented in Figure 1, which means they can only access the past context, ignoring the information from the future. However, future context is vital for recognition task likewise. An elegant solution to this problem is bidirectional recurrent neural networks (BRNNs) (Schuster,

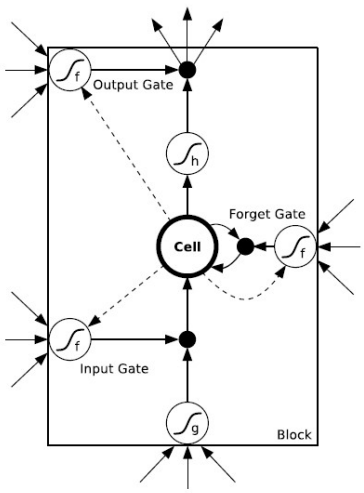



*Figure 2. LSTM memory block with one cell, figure extracted from (Graves* et al.*, 2012)*

Paliwal, 1997), which presents every training sequence forward and backward to two separate recurrent hidden layers. Using LSTM as the network architecture in a BRNN yields bidirectional LSTM (Graves, Schmidhuber, 2005). Combining the advantages of BRNN and LSTM, BLSTM offers access to long range context in two directions. In the nature of things, this advanced architecture outperformed other models in several tasks (Graves *et al*., 2009; Álvaro *et al*., 2013; 2014a).

### *2.2. CTC*

RNNs have the ability to access contextual information which is very important in sequence recognition. However, if we want to use RNNs for sequence labelling, there exists at least one difficulty: a target function must be defined (Bourlard, Morgan, 2012). In other words, neural networks require separate training targets for each time-step. CTC was specifically designed for sequence labelling problems where the alignment between the inputs and the target labels is unknown. CTC achieves this by allowing the network to make label predictions at any point in the input sequence, so long as the overall sequence of character labels is correct. We introduce CTC briefly here; for detailed description, refer to A. Graves' original paper (Graves *et al*., 2012).

For a labelling task where the labels are drawn from an alphabet $A$ ($|A| = N$), CTC consists of a softmax output layer with one more unit than there are labels in $A$. Defining the extended alphabet $A' = A \cup [blank]$, $p_{tj}$ is the probability of outputting the $j^{\text{th}}$ label at the $t^{\text{th}}$ time step given the input sequence $X$ of length $T$, where $j$ is

from 1 to $N+1$ and $t$ is from 1 to $T$. Let $A'^T$ denote the set of sequences over $A'$ with length $T$ and any sequence $\pi \in A'^T$ is referred to as a path. Then, assuming the output probabilities at each time-step to be independent of those at other time-steps, the probability of outputting a sequence $\pi$ would be:

$$p(\pi|X) = \prod_{t=1}^{T} p_{t\pi_t} \tag{1}$$

To get the possible labellings of $X$, a many-to-one function $F : A'^T \rightarrow A^{\leq T}$ is defined from the set of paths $\{\pi\}$ onto the set of labellings $\{l\}, l \in A^{\leq T}$. The specific method is to first remove the repeated labels and then the blanks $(-)$ from the paths. For example $F(cc--aaa-rr-) = F(c---aa--rrr) = car$. Since the paths are mutually exclusive, the probability of a labelling sequence $l \in A^{\leq T}$ can be calculated by summing the probabilities of all the paths mapped onto it by $F$ :

$$p(l|X) = \sum_{\pi \in F^{-1}(l)} p(\pi|X) \tag{2}$$

As the number of paths grows exponentially with the length of the input sequence, the calculation of Eqn. 2 seems to be problematic. A variant of the forward-backward algorithm was proposed to compute $p(l|X)$ efficiently.

Consider a modified label sequence $l'$ with blanks added to the beginning and the end of $l$, and inserted between every pair of consecutive labels. Suppose that the length of $l$ is $U$, apparently the length of $l'$ is $U' = 2U + 1$. For a labelling $l$, let the forward variable $\alpha(t, u)$ denote the summed probability of all length $t$ paths that are mapped by $F$ onto the length $u/2$ prefix of $l$, and let the set $V(t, u) = \{\pi \in A'^t : F(\pi) = l_{1:u/2}, \pi_t = l'_u\}$ be the set of all length $t$ paths that are mapped by $F$ onto the length $u/2$ prefix of $l$, where $u$ is from 1 to $U'$ and $u/2$ is rounded down to an integer value. Thus:

$$\alpha(t, u) = \sum_{\pi \in V(t,u)} \prod_{i=1}^{t} p_{i\pi_i} \tag{3}$$

As can be seen, the forward variables at time $t$ can be calculated recursively from those at time $t-1$.

$$\alpha(t, u) = p_{tl'_u} \sum_{i=f(u)}^{u} \alpha(t-1, i) \tag{4}$$

where

$$f(u) = \begin{cases} u-1 & \text{if } l'_u = \text{blank or } l'_{u-2} = l'_u \\ u-2 & \text{otherwise} \end{cases} \tag{5}$$

Note that

$$\alpha(t, u) = 0, \forall u < U' - 2(T-t) - 1 \tag{6}$$

because these variables correspond to states for which there are not enough time steps left to complete the sequence. Given the above formulation, the probability of $l$ can

be expressed as the sum of the forward variables with and without the final blank at time $T$.

$$p(l|X) = \alpha(T, U') + \alpha(T, U' - 1) \tag{7}$$

The backward variable can be calculated in a similar way, and will not be covered here.

The CTC loss function $L(S)$ is defined as the negative log probability of correctly labelling all the training examples in some training set $S$:

$$L(S) = -\ln \prod_{(X,l)\in S} p(l|X) \tag{8}$$

BLSTM networks can be trained to minimize the differentiable loss function $L(S)$ using any gradient-based optimization algorithm. The basic idea is to find the derivative of the loss function with respect to each of the network weights, then adjust the weights in the direction of the negative gradient.

## 3. The Representation of ME

An input mathematical expression consists of one or more strokes. The sequence of strokes in an expression can be described as $S = (s_1, ..., s_n)$. For $i < j$, we assume $s_i$ has been entered before $s_j$. In effect, $S$ is the path (different from the path mentioned in Section 2.2) following time order in a Stroke Label Graph (SLG) which can model a ME. In this section, we first introduce how to represent an expression with SLG and then how to build SLG from a path $S$ outputted by a recognizer.

### *3.1. Stroke Label Graph*

Structures can be depicted at three different levels: symbolic, object and primitive (Zanibbi *et al.*, 2013). In the case of handwritten ME, the corresponding levels are expression, symbol and stroke. A symbol can consist of one or more strokes. Grouping strokes that belong to the same symbol is noted as **symbol segmentation**.

**Symbol Relation Tree (SRT)** It is possible to describe a ME at the symbol level using a SRT which represents spatial relationships between symbols. In SRT, nodes represent symbols, while labels on the edges indicate the relationships between symbols. As the relations between the symbols are inherited from the math semantics, the graph built with the symbols and their relations is a tree. For example, in Figure 3a, the leftmost symbol '-' on the base line is the root of the tree with symbol 'a' above and symbol 'c' below it. In Figure 3b, the symbol 'a' is the root; the symbol '+' is on the right of 'a'.

**Stroke Label Graph (SLG)** If we go down at the stroke level, a SLG can be derived from the SRT. Specially, the symbol node having several strokes is split into

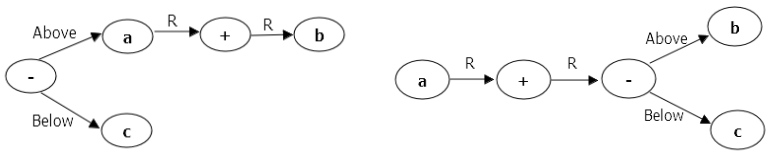



*Figure 3. The symbol relation tree (SRT) for (a) $\frac{a+b}{c}$ and (b) $a + \frac{b}{c}$. 'R' is for left-right relationship*

several stroke nodes; the tree structure is transformed into the graphe structure. In SLG, nodes represent strokes, while labels on the edges encode either segmentation information or symbol relationships. Relationships are defined at the level of symbols, implying that all strokes (nodes) belonging to one symbol have the same input and output edges. Consider the simple expression '2 + 2' written using four strokes (two strokes for '+') in Figure 4a. The corresponding SRT and SLG are shown in Figure 4b and Figure 4c respectively. As Figure 4c illustrates, nodes of SLG are labeled with the class of the corresponding symbol to which the stroke belongs. A dashed edge corresponds to segmentation information; it indicates that a pair of strokes belongs to the same symbol. In this case, the edge label is the same as the common symbol label. On the other hand, the non-dashed edges define spatial relationships between nodes and are labeled with one of the different possible relationships between symbols. As a consequence, all strokes belonging to the same symbol are fully connected, nodes and edges sharing the same symbol label; when two symbols are in relation, all strokes from the source symbol are connected to all strokes from the target symbol by edges sharing the same relationship label.

The spatial relationships as defined in the CROHME competition are: $Right$, $Above$, $Below$, $Inside$ (for square root), $Superscript$, $Subscript$. For the case of nth-Roots, like $\sqrt[3]{x}$, we define that the symbol 3 is $Above$ the square root and $x$ is $Inside$ the square root. The limits of an integral and summation are designated as $Above$ or $Superscript$ and $Below$ or $Subscript$ depending on the actual position of the bounds. In addition, the label '_' is used to denote that there is no relation between two symbols.

Since CROHME 2013, SLG has been used to represent mathematical expressions (Mouchère *et al.*, 2016). SLG is the official format to represent the ground-truth of handwritten math expressions and also for the recognition outputs. Thus, it allows detailed error analyses on stroke, symbol and expression levels. In order to be comparable to the ground truth SLG and allow error analyses on any level, our recognition system aims to generate SLG from the input. It means that we need a label decision for each stroke and each stroke pair used in a symbol relation.

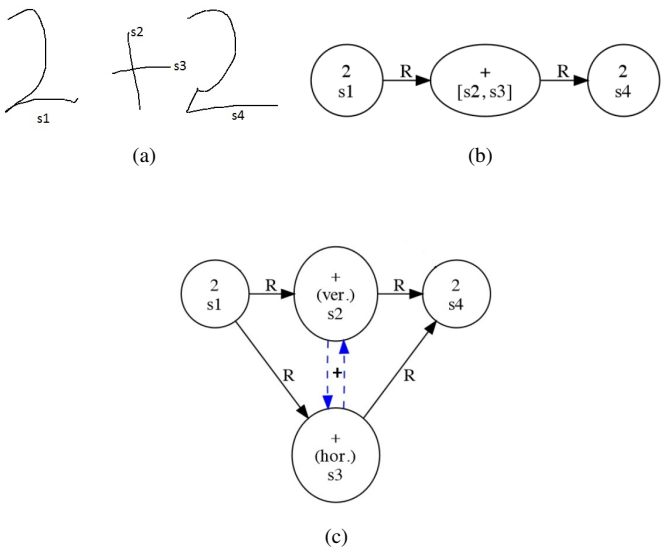



*Figure 4. (a) '2 + 2' written with four strokes; (b) the symbol relation tree of '2 + 2'; (c) the SLG of '2 + 2'. The four strokes are indicated as s1, s2, s3, s4 in writing order. (ver.) and (hor.) are added to differentiate the vertical and the horizontal strokes for '+'. 'R' is for left-right relationship*

### *3.2. Build SLG from* $S$

SLG can model ME distinctly which means once a correct SLG is built from $S$, a math formula is fully recognized. A path in SLG can be defined as $\Phi_i = (n_0, n_1, n_2, ..., n_e)$, where $n_0$ is the starting node and $n_e$ is the end node. The set of nodes of $\Phi_i$ is $n(\Phi_i) = \{n_0, n_1, n_2, ..., n_e\}$ and the set of edges of $\Phi_i$ is $e(\Phi_i) = \{n_0 \rightarrow n_1, n_1 \rightarrow n_2, ..., n_{e-1} \rightarrow n_e\}$, where $n_i \rightarrow n_{i+1}$ denotes the edge from $n_i$ to $n_{i+1}$. As mentioned before, the sequence of strokes in an expression can be described as $S = (s_1, ..., s_n)$ which is exactly the path following stroke writing order (called $time$ path, $\Phi_t$) in SLG. Still taking '2 + 2' for example, the $time$ path is presented with red color in Figure 5a. We propose to build SLG from $S$ by adding the edges which can be deduced from the labeled path to obtain a coherent SLG as depicted on Figure 5c. This time sequence is used since it is the most intuitive and it is readily available. However, it does not always allow a complete construction of the SLG. Two successful examples are given below to illustrate this point.

From Figure 5b, the corresponding 1-D sequence of the $time$ path $\Phi_t$ is $\{s1, s1 \rightarrow s2, s2, s2 \rightarrow s3, s3, s3 \rightarrow s4, s4\}$ labeled as $\{2, R, +, +, +, R, 2\}$. This sequence alternates the node labels $\{2, +, 2\}$ and the edge labels $\{R, +, R\}$. Given the labeled sequence $\{2, R, +, +, +, R, 2\}$, the information that $s2$ and $s3$ belong to the same

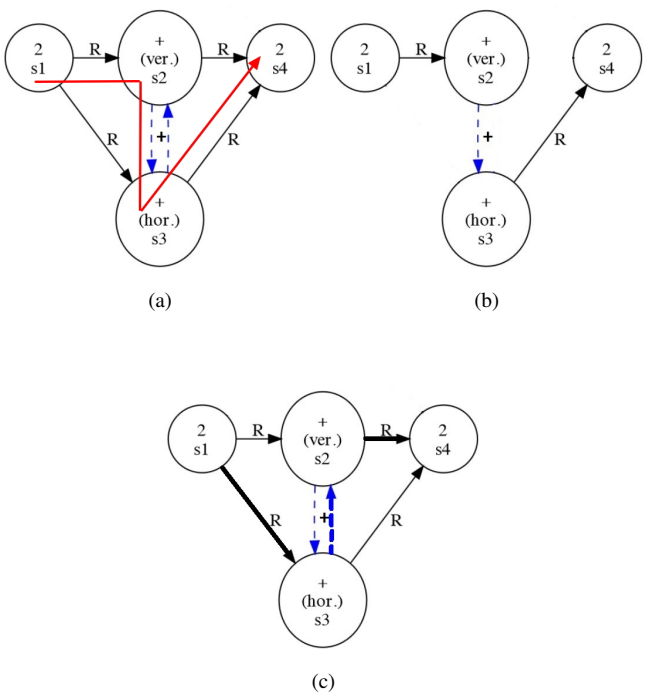



*Figure 5. (a) The time path (red) in SLG; (b) the time path; (c) the built SLG of '2 + 2', added edges are depicted as bold*

symbol $+$ can be derived. According to the rule that all strokes in a symbol have the same input and output edges, the edges from $s1$ to $s3$ and from $s2$ to $s4$ will be added automatically. With the rule that double-direction edge represents segmentation information, the edge from $s3$ to $s2$ will be added automatically. The added edges are shown in bold in Figure 5c. In this case a correct SLG is built from $\Phi_t$. This proposal works well on linear expressions. It can also deal with a part of 2-D expressions. With expression $P^{eo}$ shown in Figure 6a, the sequence of strokes and edges is $\{P, P, P, Superscript, e, R, o\}$. All the spatial relationships are covered in it and naturally a correct SLG can be generated.

However, this kind of sequence fails on a number of 2-D expressions. Figure 6c presents a failed case. According to time order, 2 and $h$ are neighbors but there is no edge between them as can be seen on Figure 6d. In the best case the system can output a sequence is stroke and edge labels $\{r, Superscript, 2, \_, h\}$. The $Right$ relationship existing between $r$ and $h$ drawn with red color in Figure 6d is missing in the previous sequence. It is not possible to build the correct SLG with $\{r, Superscript, 2, \_, h\}$.

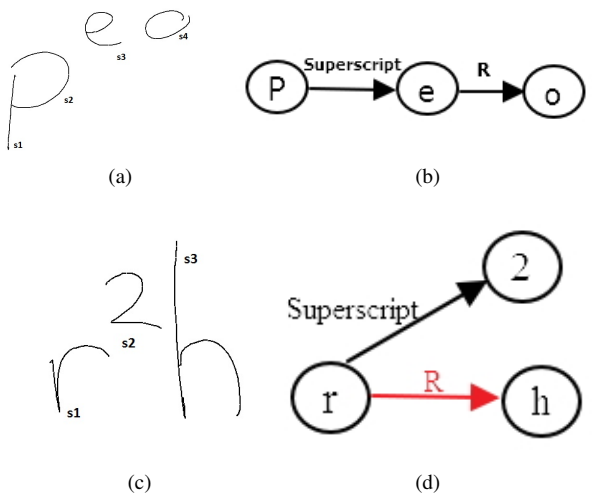



*Figure 6. (a) $P^{eo}$ written with four strokes; (b) the SRT of $P^{eo}$; (c) $r^2h$ written with three strokes; (d) the SRT of $r^2h$, the red edge cannot be generated by the time sequence of strokes*

For the training process, it means missing a training sample for relationship $Right$. For the recognition part, it means this expression can never be recognized fully. Being aware of this limitation, the 1-D time sequence of strokes is used to train the BLSTM and the outputted sequence of labels during recognition will be completed to generate a SLG graph as much as possible.

## 4. Detailed Implementation

An online mathematical expression is a sequence of strokes described as $S = (s_1, ..., s_n)$. In this section, we present how to get the above-mentioned 1-D sequence of labels from S with the BLSTM and local CTC model. CTC layer only outputs the final sequence of labels while the alignment between the inputs and the labels is unknown. BLSTM with CTC model may emit the labels before, after or during the segments (strokes). Furthermore, it tends to glue together successive labels that frequently co-occur (Graves *et al.*, 2012). However, the label of each stroke is required to build SLG, which means the alignment information between a sequence of strokes and a sequence of labels should be provided. Thus, we propose local CTC here, constraining the network to emit the label during the segment (stroke), not before or after. First part is to feed the inputs of the BLSTM with S. Then, we focus on the network training process—local CTC technology. Lastly, the recognition strategies adopted in this paper will be explained in detail.

### *4.1. BLSTM Inputs*

To feed the inputs of the BLSTM, it is important to scan the points belonging to the strokes themselves (on-paper points) as well as the points separating one stroke from the next one (in-air points). We expect that the visible strokes will be labeled with corresponding symbol labels and that the non-visible strokes connecting two visible strokes will be assigned with one of the possible edge labels (could be relationship label, symbol label or '_'). Thus, besides re-sampling points from visible strokes, we also re-sample points from the straight line which links two visible strokes, as can be seen in Figure 7. In the rest of this paper, $strokeD$ and $strokeU$ are used to indicate a re-sampled pen-down stroke and a re-sampled pen-up stroke for convenience.

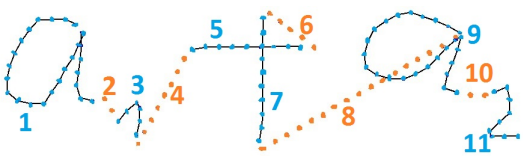



*Figure 7. The illustration of on-paper points (blue) and in-air points (red) in $time$ path, $a_1 + a_2$ written with 6 strokes*

Given each expression, we first re-sampled points both from visible strokes and between two successive visible strokes according the time order. 1-D unlabeled sequence can be described as $\{strokeD_1, strokeU_2, strokeD_3, strokeU_4, ..., strokeD_K\}$ with $K$ being the number of re-sampled strokes. Note that if $s$ is the number of visible strokes in this path, $K = 2 * s - 1$. Each stroke ($strokeD$ or $strokeU$) consists of one or more points. At a time-step, the input provided to the BLSTM is the feature vector extracted from one point. Without CTC output layer, the ground-truth of every point is required for BLSTM training process. With CTC layer, only the target labels of the whole sequence is needed, the presegmented training data is not required. In this paper, a local CTC technology is proposed and the ground-truth of each stroke is required. The label of $strokeD_i$ should be assigned with the label of the corresponding node in SLG; the label of $strokeU_i$ should be assigned with the label of the corresponding edge in SLG. If no corresponding edge exists, the label $NoRelation$ will be defined as '_'.

### *4.2. Training Process—local CTC*

Frame-wise training of RNNs requires separate training targets for every segment or timestep in the input sequence. Even though presegmented training data is available, it is known that BLSTM and CTC stage have better performance when a "blank" label is introduced during training (Bluche *et al.*, 2015), so that better decision can be made only at some point in the input sequence. Of course doing so, precise segmentation of the input sequence is not possible. As the label of each stroke is required to build a SLG, we should make decisions on stroke ($strokeD$ or $strokeU$) level instead of sequence level (as classical CTC) or point level during the recognition process.

Thus, a correspondingly stroke level training method allowing the usage of blank label under the constraint of labeling each stroke should be developed. That is why local CTC is proposed here.

For each stroke, label sequences should follow the state diagram given in Figure 8. For example, suppose character $c$ is written with one stroke and 3 points are resampled from the stroke. The possible labels of these points can be $ccc$, $cc-$, $c--$, $---$ $c$, $-cc$ and $-c-$. More generally, the number of possible label sequences is $n*(n+1)/2$ ($n$ is the number of points), which is actually 6 with the proposed example.

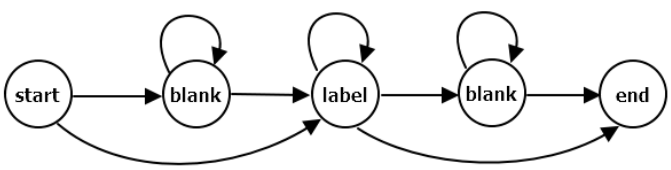



*Figure 8. The possible sequences of point labels in one stroke*

In Section 2.2, we introduce CTC technology proposed by Graves. The forward variable $\alpha(t,u)$ denotes the summed probability of all length $t$ paths that are mapped by $F$ onto the length $u/2$ prefix of $l$. In our case, $\alpha(t,u)$ can be computed recursively according to Eqn. 9:

$$\alpha(t,u) = p_{tl'_u} \sum_{i=f_{local}(u)}^{u} \alpha(t-1,i) \tag{9}$$

where

$$f_{local}(u) = \begin{cases} u-1 & \text{if } l'_u = \text{blank} \\ u-2 & \text{otherwise} \end{cases} \tag{10}$$

In the original Eqn. 5, the value $u-1$ was also assigned when $l'_{u-2} = l'_u$, enabling the transition from $\alpha(t-1,u-2)$ to $\alpha(t,u)$. This is the case when there are two repeated successive symbols in the final labelling. With regard to the corresponding paths, there exists at least one blank between these two symbols. Otherwise, only one of these two symbols can be obtained in the final labelling. In our case, as one label will be selected for each stroke, the above-mentioned limitation can be ignored. Given the input sequence $X$ with $U$ strokes, assuming that one stroke has not to be segmented as it belongs to at most one symbol, its labeling $l$ will be a sequence of $U$ symbols. Correspondingly, the size of the modified label sequence $l'$ is $2*U+1$. Suppose that the input at time $t$ belongs to $i^{\text{th}}$ stroke ($i$ from 1 to $U$), then we have

$$\alpha(t,u) = 0, \forall u / u < (2*i-1), u > (2*i+1) \tag{11}$$

which means the only possible arrival positions for time $t$ are $l'_{2*i-1}$, $l'_{2*i}$, $l'_{2*i+1}$. Figure 9 demonstrates the local CTC forward-backward algorithm using the example '2a' which is written with 2 visible strokes. The corresponding label sequences $l$ and $l'$

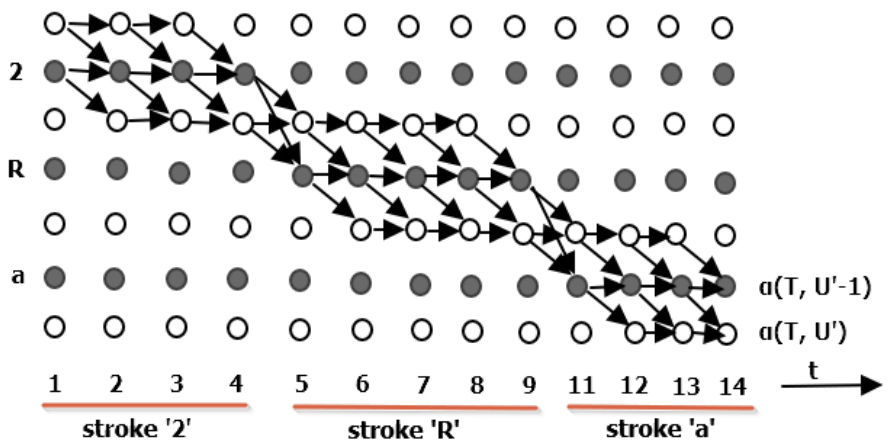



*Figure 9. Local CTC forward-backward algorithm. Black circles represent labels and white circles represent blanks. Arrows signify allowed transitions. Forward variables are updated in the direction of the arrows, and backward variables are updated in the reverse direction*

of it are '2Ra' and '-2-R-a-' respectively (R is for $Right$ relationship). We re-sampled 4 points for pen-down stroke '2', 5 points for pen-up stroke 'R' and 4 points for pen-down stroke 'a'. From this figure, we can see each part located on one stroke is exactly the CTC forward-backward algorithm. That is why the output layer adopted in this paper is called local CTC.

The backward variable can be modified in a similar way and we do not introduce the detail here.

### *4.3. Recognition Strategies*

Once the network is trained, we would ideally label some unknown input sequence $X$ by choosing the most probable labelling $I^*$:

$$I^* = \underset{l}{argmax}\ p(l|X) \tag{12}$$

Since local CTC is already adopted in the training process in this paper, naturally recognition should be performed on stroke ($strokeD$ and $strokeU$) level. As explained in section 3.2 to build the Label Graph, we need to assign one single label

to each stroke. At that stage, for each point or time step, the network outputs the probabilities of this point belonging to different classes. Hence, a pooling strategy is required to go from the point level to the stroke level. We propose two kinds of decoding methods: maximum decoding and local CTC decoding, both based on stroke level.

**Maximum decoding** With the same method taken in (Graves *et al.*, 2012) for isolated handwritten digits recognition using a multidimensional RNN with LSTM hidden layers, we first calculate the cumulative probabilities over the entire stroke. For stroke $k$, let $o^k = \{p_{ij}^k\}$, where $p_{ij}^k$ is the probability of outputting the $j^{\text{th}}$ label at the $i^{\text{th}}$ point. Suppose that we have $N$ classes of labels (including blank), then $j$ is from 1 to $N$; $|s_k|$ points are re-sampled for stroke $k$, then $i$ is from 1 to $|s_k|$. Thus, the cumulative probability of outputting the $j^{\text{th}}$ label for stroke $k$ can be computed as

$$p_j^k = \sum_{i=1}^{|s_k|} p_{ij}^k \tag{13}$$

Then we choose for stroke $k$ the label with the highest $p_j^k$.

**Local CTC decoding** With the output $o^k$, we choose the most probable label for the stroke $k$:

$$l^{k*} = \underset{l^k}{argmax}\ p(l^k|o^k) \tag{14}$$

In this work, each stroke outputs only one label which means we have $N-1$ possibilities of label of stroke. $blank$ is excluded because it can not be a candidate label for stroke. With the already known $N-1$ labels, $p(l^k|o^k)$ can be calculated using the algorithm depicted in Section 2.2. Specifically, based on the Eqn. 7 writes Eqn. 15,

$$p(l^k|o^k) = \alpha(|s_k|, 3) + \alpha(|s_k|, 2) \tag{15}$$

with $T = |s_k|$ and $U' = 3$ ($l'$ is blank, label, blank). For each stroke, we compute the probabilities corresponding to $N-1$ labels and then select the one with the largest value. In mathematical expression recognition task, more than 100 different labels are included. If Eqn. 15 is computed more that 100 times for every stroke, undoubtedly it would be a time-consuming task. A simplified strategy is adopted. We sort the $p_j^k$ and keep the top 10 probable labels (excluding $blank$). From these 10 candidates, we choose the one which has the highest $p(l^k|p^k)$. In this way, Eqn. 15 is computed only 10 times for each stroke, greatly reducing the computation time.

Furthermore, we add two constraints when choosing label for stroke: (1) the label of $strokeD$ should be one of the symbol labels, excluding the relationship labels, like strokes $1, 3, 5, 7, 9, 11$ in Figure 10. (2) the label of $strokeU_i$ is divided into 2 cases, if the labels of $strokeD_{i-1}$ and $strokeD_{i+1}$ are different, it should be one of the six relationships (strokes $2, 8, 10$) or '_' (stroke $4$); otherwise, it should be relationships, '_' or the label of $strokeD_{i-1}$ ($strokeD_{i+1}$). Taking stroke 6 shown in Figure 10 for example, if '+' is assigned to it means that the corresponding pair of nodes (strokes 5 and 7) belongs to the same symbol while '_' or relationship refers to 2 nodes belonging

to 2 symbols. Note that the labels of pen-down stokes are chosen first and then pen-up strokes.

After recognition, post-processing (adding edges) should be done in order to build the SLG. The way to proceed has been already introduced in Section 3.2.

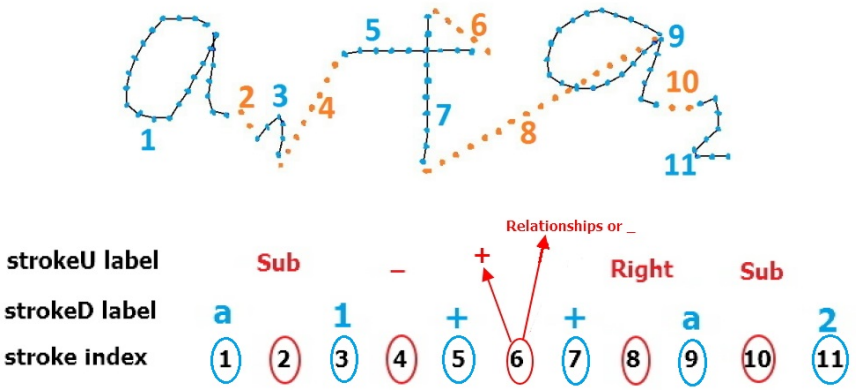



*Figure 10. Illustration for the decision of the label of stroke. For being more readable, all the strokes are given the correct label except stroke* 6

## 5. Features

A stroke is a sequence of points sampled from the trajectory of a writing tool between a pen-down and a pen-up at a fixed interval of time. Then an additional re-sampling is performed with a fixed spatial step to get rid of the writing speed. The number of re-sampling points depends on the size of expression. For each expression, we re-sample with $10 \times (length/avrdiagonal)$ points. Here, $length$ refers to the length of all the strokes in the path (including the gap between successive strokes) and $avrdiagonal$ refers to the average diagonal of the bounding boxes of all the strokes in an expression. Since the features used in this work are independent of scale, the operation of re-scaling can be omitted.

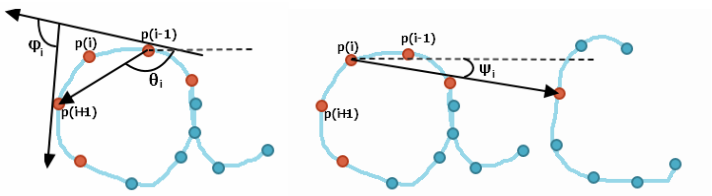
Subsequently, we compute five local features per point, which are quite close to the state of art (Álvaro *et al.*, 2013; Awal *et al.*, 2014). For every point $p_i(x, y)$ we obtained 5 features:

$$[\sin\theta_i, \cos\theta_i, \sin\phi_i, \cos\phi_i, PenUD_i]$$

with:

- $\sin\theta_i$, $\cos\theta_i$ are the sine and cosine directors of the tangent of the stroke at point $p_i(x, y)$;

- $\phi_i = \Delta\theta_i$, defines the change of direction at point $p_i(x, y)$;
- $PenUD_i$ refers to the state of pen-down or pen-up.




(a) (b)

*Figure 11. The illustration of (a) $\theta_i$, $\phi_i$ and (b) $\psi_i$ used in feature description. The points related to feature computation at $p_i$ are depicted in red*

Even though BLSTM can access contextual information from past and future in a long range, it is still interesting to see if there is any difference when contextual features are added in the recognition task. Thus, we extract two contextual features for each point:

$$[\sin\psi_i, \cos\psi_i]$$

with:

- $\sin\psi_i$, $\cos\psi_i$ are the sine and cosine directors of the vector from the point $p_i(x, y)$ to its closest pen-down point which is not in the current stroke. For the single-stroke expressions, $\sin\psi_i = 0$, $\cos\psi_i = 0$.

## 6. Experiments

We use the RNNLIB library (Graves, 2013) for training a BLSTM model. Both frame-wise training and local CTC training are adopted in experiments. For each training process, the network having the best classification error (frame-wise) or CTC error (local CTC) on validation data set is saved. Then, we test this network on the test data set. The maximum decoding (Eqn. 13) is used for frame-wise training network. With regard to local CTC, either the maximum decoding or local CTC decoding (Eqn. 15) can be used.

With the Label Graph Evaluation library (LgEval) (Mouchère *et al.*, 2014), the recognition results can be evaluated on symbol level and on expression level. We introduce several evaluation criteria: symbol segmentation ('Segments'), refers to a symbol that is correctly segmented whatever the label; symbol segmentation and recognition ('Seg+Class'), refers to a symbol that is segmented and classified correctly; spatial relationship classification ('Tree Rels.'), a correct spatial relationship

between two symbols requires that both symbols are correctly segmented and with the right relationship label.

For all experiments the network architecture and configuration are as follows:

- The input layer size: 5 or 7 (when considering the 2 additionnal context features)
- The output layer size: the number of class (up to 109)
- The hidden layers: 2 layers, the forward and backward, each contains 100 single-cell LSTM memory blocks
- The weights: initialized uniformly in [-0.1, 0.1]
- The momentum: 0.9

This configuration has obtained good results in both handwritten text recognition (Graves *et al*., 2009) and handwritten math symbol classification (Álvaro *et al*., 2013; 2014a).

### *6.1. Data sets*

There are three data sets in this paper.

**Data set 1**. We select the expressions which do not include 2-D spatial relation, only left-right relation from CROHME 2014 training and test data. 2609 expressions are available for training, about one third of the full training set and 265 expressions for testing. In this case, there are 91 classes of symbols. Next, we split the training set into a new training set and validation set, 90% for training and 10% for validation. The output layer size is 94 (91 symbol classes + $Right$ + $NoRelation$ + $blank$). In left-right expressions, $NoRelation$ will be used for indicating delayed strokes.

**Data set 2**. The depth of expressions in this data set is limited to 1. It imposes that two sub-expressions having a spatial relationship ($Above$, $Below$, $Inside$, $Superscript$, $Subscript$) should be left-right expressions. 5820 expressions are selected for training from CROHME 2014 training set; 674 expressions for test from CROHME 2014 test set. Also, we divide 5820 expressions into the new training set and validation set, 90% for training and 10% for validation. The output layer size is 102 (94 symbol classes + 6 relationships + $NoRelation$ + $blank$).

**Data set 3**. The complete data set from CROHME 2014, 8834 expressions for training and 982 expressions for test. Also, we divide 8834 expressions for training (90%) and validation (10%). The output layer size is 109 (101 symbol classes + 6 relationships + $NoRelation$ + $blank$).

$blank$ is only for local CTC training. Figure 12 show a handwritten math expression extracted from CROHME 2014 data set.

*Figure 12. A real example from CROHME 2014 data set (sample from the data set 1)*

### *6.2. Experiment 1*

In this experiment, we evaluate the proposed solution on data sets of different complexity. We take local CTC training and local CTC decoding methods. Only 5 local features are extracted at each point for training. Each system is trained only once.

The evaluation results on symbol level for the 3 data sets are provided in Table 1 including recall ('Rec.') and precision ('Prec.') rates for 'Segments', 'Seg+Class', 'Tree Rels.'. As can be seen, the results in 'Segments' and 'Seg+Class' are increasing while the training data set is growing. The recall for 'Tree Rels.' is decreasing among the three data sets. It is understandable since the number of missed relationships grows with the complexity of expressions knowing the limitation of our method. The precision for 'Tree Rels.' fluctuates as the data set is expanding. The results of data set 3 are comparable to the results of CROHME 2014 because the same training and testing data sets are used. The second part of Table 1 gives the symbol level evaluation results of the top 4 participants to CROHME 2014 sorted by recall of correct symbol segmentation. The best 'Rec.' of 'Segments' and 'Seg+Class' reported in CROHME 2014 are 98.42% and 93.91% respectively. Ours are 93.26% and 84.40%, both ranked 3 out of 8 systems (7 participants in CROHME 2014 + our system). Our solution presents competitive results on symbol recognition task and segmentation task even though the symbols with delayed strokes are missed. However, our proposal, at that stage, shows limited performances at the relationship level, with the 4th position ('Rec.' = 61.85, 'Prec.' = 75.06). This is mainly because some spatial relationships are missed in the time sequence.

Table 2 shows the recognition rates at the global expression level with no error, and with at most one to three errors in the labels of SLG. This metric is very strict. For example one label error can happen only on one stroke symbol or in the relationship between two one-stroke symbols; a labeling error on a 2-stroke symbol leads to 4 errors (2 nodes labels and 2 edges labels). The recognition rate with no error on CROHME 2014 test set is 12.63%. The best one and worst one reported by CROHME 2014 are 62.68% and 15.01%. When looking at the recognition rate having less than three errors, four participants ranked between 27% and 37%, while our result is 31.98%.

*Table 1. The symbol level evaluation results on CROHME 2014 test set, including the experiment results in this work and CROHME 2014 participant results (Top 4 by recall of Segments)*

| Data set, features | Segments (%) | | Seg + Class (%) | | Tree Rels. (%) | |
|---|---|---|---|---|---|---|
| | Rec. | Prec. | Rec. | Prec. | Rec. | Prec. |
| 1, 5 | 90.11 | 80.75 | 78.91 | 70.71 | 79.87 | 73.66 |
| 2, 5 | 91.88 | 84.47 | 82.42 | 75.77 | 64.75 | 71.96 |
| **3, 5** | **93.26** | **86.86** | **84.40** | **78.61** | **61.85** | **75.06** |
| system | CROHME 2014 participant results (Top 4) | | | | | |
| III | 98.42 | 98.13 | 93.91 | 93.63 | 94.26 | 94.01 |
| I | 93.31 | 90.72 | 86.59 | 84.18 | 84.23 | 81.96 |
| VII | 89.43 | 86.13 | 76.53 | 73.71 | 71.77 | 71.65 |
| V | 88.23 | 84.20 | 78.45 | 74.87 | 61.38 | 72.70 |

*Table 2. The expression level evaluation results on CROHME 2014 test set, including the experiment results in this work and CROHME 2014 participant results (Top 4)*

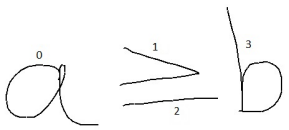
| Data set, features | correct (%) | <= 1 error | <= 2 errors | <= 3 errors |
|---|---|---|---|---|
| 1, 5 | 25.28 | 40.75 | 49.06 | 52.08 |
| 2, 5 | 12.76 | 25.07 | 31.16 | 36.20 |
| **3, 5** | **12.63** | **21.28** | **27.70** | **31.98** |
| system | CROHME 2014 participant results (Top 2) | | | |
| III | 62.68 | 72.31 | 75.15 | 76.88 |
| I | 37.22 | 44.22 | 47.26 | 50.20 |
| VII | 26.06 | 33.87 | 38.54 | 39.96 |
| VI | 25.66 | 33.16 | 35.90 | 37.32 |

### *6.3. Experiment 2*

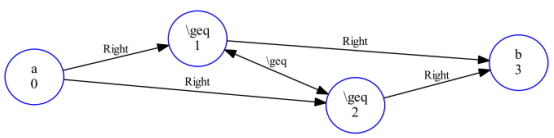
In this experiment, we test the effects of different training and decoding methods and the contextual features on recognition results. All the systems are trained and evaluated with data set 3. As the weights are initialized randomly, in order to get a convincing conclusion, each system is trained four times. Thus, each result of this exp. is a mean value.

As shown in Table 3, with the first 2 networks, we can conclude that local CTC training improve the performance compared to frame-wise training. Furthermore, this kind of training method reduce training time significantly. Observing the results of the second and third systems, Local CTC decoding does not help the recognition process but cost more computation. Maximum decoding is a better choice in this work. In the fourth system, we test the effect of contextual features on BLSTM networks. However, the results stay at the same level with system trained with only local features.

We present a correctly recognized sample and an incorrectly recognized sample in Figure 13 and Figure 14 respectively.

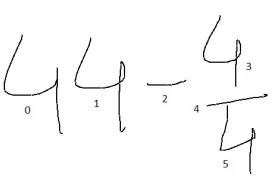



(a)

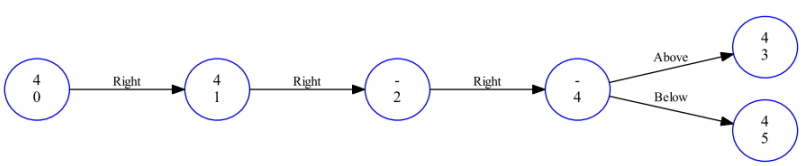



(b)

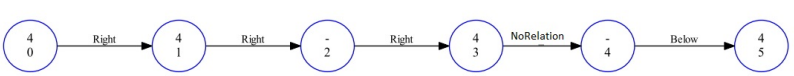
*Figure 13. (a) $a \geq b$ written with four strokes; (b) the built SLG of $a \geq b$ according to the recognition result, all labels are correct*




(a)




(b)




(c)

*Figure 14. (a) $44 - \frac{4}{4}$ written with six strokes; (b) the ground-truth SLG; (c) the rebuilt SLG according to the recognition result. Three edge errors occurred: the $Right$ relation between stroke 2 and 4 was missed because there is no edge from stroke 2 to 4 in the time path; the edge from stroke 4 to 3 was missed for the same reason; the edge from stroke 2 to 3 was wrongly recognized and it should be labeled as $NoRelation$*

*Table 3. The symbol level evaluation results on CROHME 2014 test set with different training and decoding methods, features*

| Feat. | Train | Decode | Segments (%) | | Seg + Class (%) | | Tree Rels. (%) | |
|---|---|---|---|---|---|---|---|---|
| | | | Rec. | Prec. | Rec. | Prec. | Rec. | Prec. |
| 5 | frame-wise | maximum | 92.71 | 85.88 | 83.76 | 77.59 | 59.84 | 73.71 |
| 5 | local CTC | maximum | 93.21 | 86.73 | 84.11 | 78.26 | 61.25 | 74.51 |
| 5 | local CTC | local CTC | 93.2 | 86.71 | 84.11 | 78.25 | 61.71 | 74.67 |
| 7 | local CTC | maximum | 93 | 86.43 | 84 | 78.06 | 61.73 | 74.06 |

## 7. Conclusion

The capability of BLSTM networks to process graphical two-dimensional languages such as handwritten mathematical expressions is explored in this study. Using online math expressions, which are available as a temporal sequence of strokes, we produce a labeling at the stroke level using a BLSTM network with a local CTC. Then we propose to build a two-dimensional (2-D) expression from this sequence of labels. Our solution presents competitive results with CROHME 2014 on symbol recognition task and segmentation task. Proposing a global solution to perform segmentation, recognition and interpretation, with no dedicated stages, is a major advantage of the proposed solution. To some extent, at the present time, it fails on the relationship recognition task. This is primarily due to an intrinsic limitation, since currently, a single path following the time sequence of strokes in SLG is used to build the expression. Actually, some important relationships are discarded. In future, a better solution will be proposed such that less relationships will be missed. Several paths can be merged to build a more robust decision which takes into account most of possible stroke configurations in time and space.

*Acknowledgements*

*We would like to thank the China Scholarship Council for the financial support for Ting ZHANG PhD studentship at Nantes university.*

## References

Álvaro F., Sánchez J.-A., Benedí J.-M. (2013). Classification of on-line mathematical symbols with hybrid features and recurrent neural networks. In *Document analysis and recognition (icdar), 2013 12th international conference on*, pp. 1012–1016.

Álvaro F., Sánchez J.-A., Benedí J.-M. (2014a). Offline features for classifying handwritten math symbols with recurrent neural networks. In *Pattern recognition (icpr), 2014 22nd international conference on*, pp. 2944–2949.

Álvaro F., Sánchez J.-A., Benedí J.-M. (2014b). 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.

Álvaro F., Sánchez J.-A., Benedí J.-M. (2016). An integrated grammar-based approach for mathematical expression recognition. *Pattern Recognition*, Vol. 51, pp. 135–147.

Awal A.-M., Mouchère H., Viard-Gaudin C. (2014). A global learning approach for an on-line handwritten mathematical expression recognition system. *Pattern Recognition Letters*, Vol. 35, pp. 68–77.

Bluche T., Ney H., Louradour J., Kermorvant C. (2015). Framewise and ctc training of neural networks for handwriting recognition. In *Document analysis and recognition (icdar), 2015 13th international conference on*, pp. 81–85.

Bourlard H. A., Morgan N. (2012). *Connectionist speech recognition: a hybrid approach* (Vol. 247). Springer Science & Business Media.

Frinken V., Fischer A., Baumgartner M., Bunke H. (2014). Keyword spotting for self-training of blstm nn based handwriting recognition systems. *Pattern Recognition*, Vol. 47, No. 3, pp. 1073–1082.

Graves A. (2013). *Rnnlib: A recurrent neural network library for sequence learning problems.*

Graves A., Liwicki M., Fernández S., Bertolami R., Bunke H., Schmidhuber J. (2009). A novel connectionist system for unconstrained handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 31, No. 5, pp. 855–868.

Graves A. *et al.* (2012). *Supervised sequence labelling with recurrent neural networks* (Vol. 385). Springer.

Graves A., Schmidhuber J. (2005). Framewise phoneme classification with bidirectional lstm networks. In *Neural networks, 2005. ijcnn'05. proceedings. 2005 ieee international joint conference on*, Vol. 4, pp. 2047–2052.

Hochreiter S., Schmidhuber J. (1997). Long short-term memory. *Neural computation*, Vol. 9, No. 8, pp. 1735–1780.

Mouchère H., Viard-Gaudin C., Zanibbi R., Garain U. (2014). Icfhr 2014 competition on recognition of on-line handwritten mathematical expressions (crohme 2014). In *Frontiers in handwriting recognition (icfhr), 2014 14th international conference on*, pp. 791–796.

Mouchère H., Zanibbi R., Garain U., 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)*, Vol. 19, No. 2, pp. 173–189.

Ray A., Rajeswar S., Chaudhury S. (2015). Text recognition using deep blstm networks. In *Advances in pattern recognition (icapr), 2015 eighth international conference on*, pp. 1–6.

Schuster M., Paliwal K. K. (1997). Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, Vol. 45, No. 11, pp. 2673–2681.

Zanibbi R., Blostein D. (2012). Recognition and retrieval of mathematical expressions. *International Journal on Document Analysis and Recognition (IJDAR)*, Vol. 15, No. 4, pp. 331–357.

Zanibbi R., Mouchère H., Viard-Gaudin C. (2013). Evaluating structural pattern recognition for handwritten math via primitive label graphs. In *Is&t/spie electronic imaging*, pp. 865817–865817.