Pattern Recognition 71 (2017) 196–206

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition

Jianshu Zhang[a], Jun Du[a,*], Shiliang Zhang[a], Dan Liu[b], Yulong Hu[b], Jinshui Hu[b], Si Wei[b], Lirong Dai[a]

[a] National Engineering Laboratory for Speech and Language Information Processing, University of Science and Technology of China, Hefei, Anhui, China
[b] IFLYTEK Research, Hefei, Anhui, China

ARTICLE INFO

Article history:
Received 19 November 2016
Revised 26 April 2017
Accepted 7 June 2017
Available online 10 June 2017

Keywords:
Handwritten mathematical expression recognition
Neural network
Attention

ABSTRACT

Machine recognition of a handwritten mathematical expression (HME) is challenging due to the ambiguities of handwritten symbols and the two-dimensional structure of mathematical expressions. Inspired by recent work in deep learning, we present Watch, Attend and Parse (WAP), a novel end-to-end approach based on neural network that learns to recognize HMEs in a two-dimensional layout and outputs them as one-dimensional character sequences in LaTeX format. Inherently unlike traditional methods, our proposed model avoids problems that stem from symbol segmentation, and it does not require a predefined expression grammar. Meanwhile, the problems of symbol recognition and structural analysis are handled, respectively, using a watcher and a parser. We employ a convolutional neural network encoder that takes HME images as input as the watcher and employ a recurrent neural network decoder equipped with an attention mechanism as the parser to generate LaTeX sequences. Moreover, the correspondence between the input expressions and the output LaTeX sequences is learned automatically by the attention mechanism. We validate the proposed approach on a benchmark published by the CROHME international competition. Using the official training dataset, WAP significantly outperformed the state-of-the-art method with an expression recognition accuracy of 46.55% on CROHME 2014 and 44.55% on CROHME 2016.

1. Introduction

Mathematical notations play an essential role in scientific documents and are indispensable for describing problems and theories in math, physics and many other fields. Recently, people have begun to use handwritten mathematical notations as input due to the rapid emergence of new technologies such as digital pens, tablets, smartphones, etc. While this natural input method is convenient, it also requires the development of systems that are able to recognize handwritten mathematical expressions (HMEs).

As a research problem, automatic recognition of a mathematical expression (ME) is different from the traditional OCR text-recognition problem and exhibits several fascinating challenges [1]. For example, the combination of the presence of two-dimensional structures, enormous ambiguities in handwritten input and a strong dependency on contextual information make ME recognition difficult, but fascinate researchers [2,3]. Achieving success in this domain could, in turn, accelerate progress in machine recognition of other two-dimensional languages.

Handwritten mathematical expression recognition (HMER) comprises two major problems [4,5]: symbol recognition and structural analysis. These two problems can be solved sequentially or globally. Sequential solutions [6,7] first segment the input expression into math symbols and then recognize them. The analysis of two-dimensional structures is then carried out based on the best symbol segmentation and symbol recognition results. In contrast, the goal of global solutions [8,9] is to recognize symbols and analyse two-dimensional structures simultaneously. Symbol recognition and structural analysis are optimised using the global information of the mathematical expression. The segmentation is then achieved as a byproduct of the global optimization.

In sequential solutions, when incorrect segmentation occurs or incorrect decisions are made during symbol recognition, errors are subsequently inherited by the structural analysis. Consequently, global solutions seem more appropriate but they are computationally expensive because the probabilities for segmentation composed of strokes (a sequence of points between a pen-down and a pen-up operation with a stylus) are exponentially expanded; there-

* Corresponding author.
E-mail addresses: xysszjs@mail.ustc.edu.cn (J. Zhang), jundu@ustc.edu.cn (J. Du), zsl2008@mail.ustc.edu.cn (S. Zhang), danliu@iflytek.com (D. Liu), ylhu3@iflytek.com (Y. Hu), jshu@iflytek.com (J. Hu), siwei@iflytek.com (S. Wei), lrdai@ustc.edu.cn (L. Dai).

http://dx.doi.org/10.1016/j.patcog.2017.06.017
0031-3203/

fore, additional effective search strategies must be executed [10]. Meanwhile, many approaches for performing structural analysis of an ME language have been investigated, including expression trees [11], two-dimensional HMM [12] and many others [13–15]. Among these, the grammar-based methods seem to be more dependable [16,17] and have performed well in several systems [7–9]. These grammars are constructed using extensive prior knowledge and the corresponding parsing algorithms are also computed. Overall, both sequential and global approaches have limitations that this study aims to address: symbol segmentation during symbol recognition is required, which introduces many difficulties; structural analysis requires a priori knowledge that defines an ME grammar; and the complexity of parsing algorithms increases exponentially with the size of the predefined grammar.

In this paper, we introduce Watch, Attend and Parse (WAP), a neural network model as an improved version of the attention-based encoder-decoder model in [18]. This model learns to "watch" an HME image and parse it into a LaTeX sequence [19]. So WAP has two components: a watcher and a parser, the parser is equipped with the attention mechanism [18,20–22]. The watcher is a convolutional neural network (CNN) [23] encoder that maps ME images to high-level features. The parser is a recurrent neural network (RNN) [24] decoder that converts these high-level features into output sequences, word by word. For each predicted word, the attention mechanism built into the parser scans the entire input ME image and chooses the most relevant region to describe a segmented symbol or implicit spatial operator. Unlike previous approaches, WAP optimises symbol segmentation automatically through its attention mechanism, and structural analysis does not rely on a predefined ME grammar. Moreover, the watcher and the parser are jointly trained. By doing this, not only the watcher extracts good features for the parser to decode but the parser also provides contextual information to tune the watcher and guide the attention.

The contributions of this paper are as follows:

1) We introduce a neural network named Watch, Attend and Parse (WAP) to recognize HMEs. This novel model is inherently different from the traditional methods. It alleviates the problems caused by symbol segmentation and the computational demands of employing an ME grammar.
2) We propose employing a deep fully convolutional neural network (FCN) as the watcher, making it possible to process large-scale input images efficiently and, therefore, allowing variable input sizes [25,26]. We also observed a key problem while training WAP, namely, a lack of coverage [27,28]. To address this problem, we propose to use a coverage-based attention model that appends a coverage vector to incorporate the attention history.
3) We also demonstrate another advantage of including attention in experimental analysis; through attention visualization, we can see how WAP completes the automatic symbol segmentation process and parses the two-dimensional structure.

Finally, we validate the capabilities of our proposed model for HMER by comparing it with state-of-the-art approach on the large dataset published by the international Competition on Recognition of Handwritten Mathematical Expressions (CROHME).

2. Related works

In this section, we describe the previous research on HMER and the recent research based on encoder-decoder frameworks.

2.1. Grammar based HMER

This section provides the relevant previous work on HMER. The problem has been studied for decades [1]. A variety of approaches have been proposed [29]. Here, we will comment on a few approaches based on grammars.

Given the two-dimensional structure of mathematical notation, most approaches are based on predefined grammars as a natural way to solve the problem. In fact, the first proposals on ME recognition belonged to this case [1]. Subsequently, different types of grammars have been investigated. For example, Chan and Yeung [30] used definite clause grammars, the Lavirotte and Pottier [17] model was based on graph grammars, Yamamoto et al. [31] presented a system using Probabilistic Context-Free Grammars (PCFG), and MacLean and Labahn [32] developed an approach using relational grammars and fuzzy sets.

Proposals based on PCFG use a probability model to analyse the structure of the expression and address ambiguities in handwritten data. Álvaro and Sanchez [9] proposed a system that parses expressions using two-dimensional stochastic context-free grammars. This is a global approach that combines several stochastic sources of information to globally determine the most likely expression. Yamamoto et al. [31] presented a version of the CYK (Cocke-Younger-Kasami) algorithm for parsing two-dimensional PCFG (2D-PCFG). They defined probability functions based on a region representation called the "hidden writing area". The model proposed by Awal et al. [8] considers several segmentation hypotheses based on spatial information, and the symbol classifier includes a rejection class, namely, "junk", to avoid incorrect segmentations and provide useful information to structure the analyser. MacLean and Labahn [33] presented a Bayesian model for recognizing HME; this model captures all recognizable interpretations of the input and organises them in a parse forest.

2.2. Neural network

Recently, a novel neural network model, namely encoder-decoder, has been exploited specifically to address sequence to sequence learning. The encoder-decoder is designed to handle variable-length input and output sequences [22]. Typically, both the encoder and the decoder are recurrent neural networks (RNN). The encoder RNN learns to encode the sequential variable-length input into a fixed-length vector. The decoder RNN then uses this vector to produce the variable-length output sequence, one word at a time. In the training stage, to generate the next predicted word, the model provides the ground truth labels from previous words as inputs to the decoder. In the inference stage, the model utilises a beam search to obtain suitable candidates for the next predicted word. Such a framework has been applied extensively to many applications including machine translation [34], parsing [35] and speech recognition [36,37].

Problems such as HMER that require mapping input images into a fixed-length vector, usually employ a CNN as the encoder. Such an encoder has been applied to handwriting recognition [38], optical character recognition of natural scenes [39–41] and image caption generation [42–44]. All these areas show positive advances. Also, when handling the image-based sequence prediction, [45] proposes to adopt the CRNN, a combination of deep CNN and RNN, to encode images. The RNN is incorporated due to its strong capability of capturing contextual information within a sequence. Generally, these models first extract a high-level feature from the raw input image. This feature can be considered as an abstract representation that is then decoded to generate a context sequence. Usually, both the encoder and the decoder are jointly trained, so that the entire model will be optimised in a more coordinated way. Although sometimes, as in [42], the encoder has been pre-trained

by using an enormous amount of training data, we would prefer to improve the encoder.

Meanwhile, attention has been found to be one of the most distinct aspects of the human visual system [46,47]. After applying an attention mechanism to an encoder-decoder model, salient regions in the static representation can dynamically rise to the forefront. This attention vector can be viewed as a set of alterable connections that allow the forward information and the backward gradients to flow more effectively.

The generality of this attention framework suggests that HMER may also be one proper application. Recently, we find a parallel work similar to this study submitted as the arXiv preprint [48], named WYGIWYS (What You Get Is What You See), which decompiles a machine-printed mathematical expression into presentational markup. Although both the approach presented in [48] and the proposed one in this study utilize the attention based encoder-decoder framework to translate mathematical expressions into LaTeX notations, there are two significant differences. The first difference lies in the encoder architecture adopted. In [48], a CNN+Bi-RNN architecture is adopted to encode the machine-printed mathematical expressions. In this paper, a deep FCN architecture is adopted as the encoder to deal with the much larger variation in the size of the math symbols in handwritten expressions than in the machine-printed ones with much smaller model footprint. It is observed from the experimental results (see Section 7) that the deep FCN architecture performs much better for HMER task. The second difference is the attention mechanism used in the decoder. WYGIWYS uses a classic attention model (Eq. (8)) which can be parameterized as a multi-layer perceptron (MLP). But for HMER, the classic attention model reveals a problem, namely, lack of coverage. We visualize this problem in Section 5. To attack this problem, we propose to use the coverage based attention model and demonstrate that significant improvement in the performance for HMER can be achieved compared to the classic one.

3. Network architecture of WAP

In this paper, rather than recognizing mathematical expression as a tree structure, we convert the expressions from two-dimensional structures to one-dimensional character sequences. The output character sequences are formatted using the common LaTeX markup system; therefore, they can be directly displayed using existing tools. Furthermore, both the input images and output LaTex sequences may have variable lengths.

Fortunately, our model is an improved version of an encoder-decoder network, while encoder-decoder networks are designed to address variable length input and output sequences [20]. The overall WAP architecture is shown in Fig. 1. The watcher is a fully convolutional network (FCN) encoder that transforms the input image into an intermediate representation. The parser is a Gated Recurrent Units (GRU) decoder [49] that uses the intermediate representation to generate a corresponding LaTeX sequence. The attention mechanism impels the decoder to focus its attention on specific parts of the input image. The encoder, decoder and attention mechanism are trained in a joint manner.

We started our model from [42]. In that model, the convolutional encoder was pre-trained and fixed; they optimised only an LSTM [50] decoder with attention using training data. This procedure is designed for two reasons. First, the Oxford VGGnet they used as an encoder was pre-trained very well to produce high quality features from input images. Second, the input images for image caption generation and those for training VGGnet were a perfect match. Both the image size and the number of image classes were identical. However, here, we did not have sufficient isolated mathematical symbols data to pre-train the convolutional encoder. Additionally, the sizes of mathematical expressions varies.

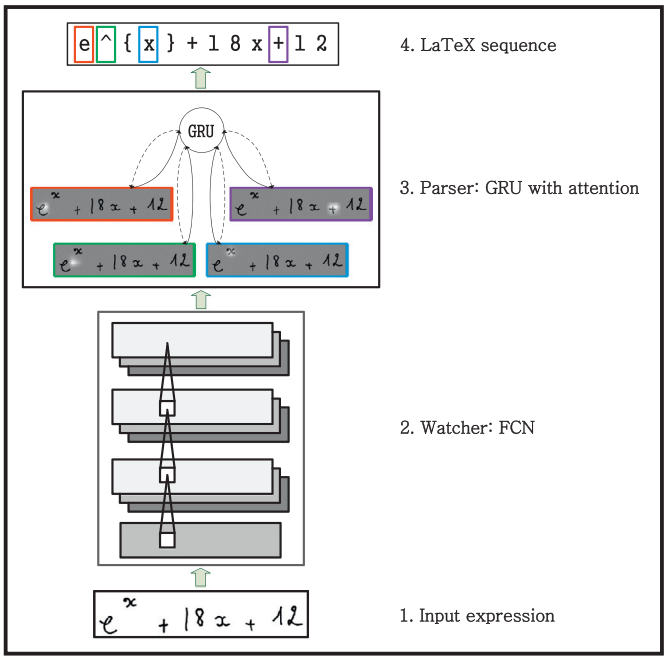

Fig. 1. Architectures of Watch, Attend, Parse for handwritten mathematical expression recognition.

Moreover, mathematical expressions include some invisible characteristics such as superscript "∧" or subscript "_" that must be translated to symbols.

Consequently, our encoder and decoder should be optimised simultaneously through joint training. By jointly optimizing the model, the GRU decoder and the attention mechanism can help adjust the FCN encoder parameters. In turn, the FCN encoder will produce a better representation of the input image, thus improving the performance of the decoder with attention mechanisms.

3.1. Watcher: fully convolutional network

Usually, offline handwritten mathematical expressions are stored as greyscale images and the pixel value is normalized between 0 and 1. When handling online handwritten expressions, such as CROHME task, the data format is a sequence of xy-coordinates points. To fully utilize the online trajectory information, the input of the watcher incorporates the image transformed from xy-coordinates points and the 8-directional pattern images which are the intermediate products of the 8-directional raw features [51,52].

CNN has been widely used for pattern recognition in recent years, especially in the field of image recognition, and has been shown to be highly invariant to translation, scaling, tilt or other types of image modifications. Moreover, CNN does not require traditional handcrafted feature extraction and data reconstruction processes before recognizing an image. The basic components of CNN are the convolution, pooling and activation layers. Convolution layers are determined by the number of input channels, the number of output feature maps and the kernel size and stride. Each kernel can be considered as a filter whose size is usually much smaller than the input. Hence, a kernel operates on a local region of input rather than the whole image. The locations that connect to higher layers are called receptive fields. On a given feature map, the kernel weights are shared, making the CNN more similar to biological neural networks, reducing the complexity of the network model. The pooling layers are usually an average pooling function or a max pooling function; they are used to reduce

model complexity and enlarge the receptive field. The activation layers are operated by element-wise nonlinear functions. If a CNN is used to complete image classification work, fully connected layers and a softmax layer are also required.

Rather than extracting features from a fully connected layer [53], we employ a fully convolutional network (FCN) containing only convolution, pooling and activation layers. The FCN has the property that it can naturally accept input of arbitrary size, which is important for HMER because the sizes of HME images are not fixed based on the lengths of the mathematical expressions. Consequently, we do not need to compress or enlarge the input to a fixed size in FCN. Additionally, FCN as the watcher can help to obtain the level of correspondence between the feature maps and the local regions of the input HME image. This approach makes sense because it allows the parser to selectively pay attention to certain parts of an image by choosing specific portions from among all the feature vectors.

As shown in Fig. 1, our model takes a single raw expression image and generates a corresponding LaTeX sequence. For example, the output sequence $\mathbf{y}$ was encoded as a sequence of one-shot encoded words.

$$y = \{\mathbf{y}_1, \ldots, \mathbf{y}_C\}\ ,\ \mathbf{y}_i \in \mathbb{R}^K \tag{1}$$

where K is the number of total words in the vocabulary and C is the length of a LaTeX sequence.

Assuming that the output of FCN encoder is a three-dimensional array of size $H \times W \times D$, consider the output as a variable-length grid of L elements, $L = H \times W$. Each of these elements is a D-dimensional annotation that corresponds to a local region of the image.

$$a = \{\mathbf{a}_1, \ldots, \mathbf{a}_L\}\ ,\ \mathbf{a}_i \in \mathbb{R}^D \tag{2}$$

3.2. Parser: gated recurrent units

To address the learning of variable-size input images and associate them with variable-length output sequences, we attempt to compute an intermediate fixed-size vector. Here, after the watcher extracts the high-level visual features from different image locations, we compute a context vector $\mathbf{c}_t$ using the weighted sum of these annotation vectors $\mathbf{a}_i$, which will be described in more details later. We then employ the recurrent neural network to produce the LaTeX sequences word by word. The probability of each predicted word is computed by the context vector $\mathbf{c}_t$, the current RNN hidden state $\mathbf{h}_t$ and the previous target word $\mathbf{y}_{t-1}$ using the following equation:

$$p(\mathbf{y}_t|\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}, \mathbf{x}) = f(\mathbf{y}_{t-1}, \mathbf{h}_t, \mathbf{c}_t) \tag{3}$$

where the function f denotes a multi layered perceptron (MLP) expanded in Eq. (11), $\mathbf{x}$ denotes the input ME image.

We utilize the Gated Recurrent Units [49], an improved version of simple RNN. GRU alleviates the vanishing and exploding gradient problem in simple RNN. The GRU hidden state $\mathbf{h}_t$ is computed as follows:

$$\mathbf{z}_t = \sigma\left(\mathbf{W}_{yz}\mathbf{E}\mathbf{y}_{t-1} + \mathbf{U}_{hz}\mathbf{h}_{t-1} + \mathbf{C}_{cz}\mathbf{c}_t\right) \tag{4}$$

$$\mathbf{r}_t = \sigma\left(\mathbf{W}_{yr}\mathbf{E}\mathbf{y}_{t-1} + \mathbf{U}_{hr}\mathbf{h}_{t-1} + \mathbf{C}_{cr}\mathbf{c}_t\right) \tag{5}$$

$$\tilde{\mathbf{h}}_t = \tanh\left(\mathbf{W}_{yh}\mathbf{E}\mathbf{y}_{t-1} + \mathbf{U}_{rh}(\mathbf{r}_t \otimes \mathbf{h}_{t-1}) + \mathbf{C}_{cz}\mathbf{c}_t\right) \tag{6}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \otimes \mathbf{h}_{t-1} + \mathbf{z}_t \otimes \tilde{\mathbf{h}}_t \tag{7}$$

where σ is the sigmoid function, $\otimes$ is an element-wise multiplication, and $\mathbf{z}_t$, $\mathbf{r}_t$ and $\tilde{\mathbf{h}}_t$ are, respectively, the update gate, reset gate and candidate activation. m and n denote the embedding and GRU dimensionality, respectively. The embedding matrix is $\mathbf{E} \in \mathbb{R}^{m \times K}$.

Intuitively, for each predicted word from the parser, the entire input image is not necessary provide the useful information. For example, in Fig. 1, the first word "e" in the output sequence corresponds only to the leftmost part of the input image: the other parts of the input expression do not need to be watched and should not participate in the computation of the context vector, $\mathbf{c}_t$. Therefore, the parser needs an attention model to know which portion is the correct place to attend to generate the next predicted word and then assign a higher weight to the corresponding local annotation vector $\mathbf{a}_i$. The attention model computes weight α_i of each annotation vector $\mathbf{a}_i$ conditioned on the previous GRU hidden state, $\mathbf{h}_{t-1}$. This can be more simply understood that the part of input image the parser should attend to depends on the words in the output sequence that have already been produced. Here, we parameterize the attention model as an MLP that is jointly trained with all the other components of the WAP neural network:

$$e_{ti} = \boldsymbol{\nu}_a^{\mathrm{T}} \tanh\left(\mathbf{W}_a\mathbf{h}_{t-1} + \mathbf{U}_a\mathbf{a}_i\right) \tag{8}$$

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^{L}\exp(e_{tk})} \tag{9}$$

Let n' denote the attention dimension; then, $\boldsymbol{\nu}_a \in \mathbb{R}^{n'}$, $\mathbf{W}_a \in \mathbb{R}^{n' \times n}$ and $\mathbf{U}_a \in \mathbb{R}^{n' \times D}$. After the weights α_{ti} (which sum to one) have been computed, the context vector, $\mathbf{c}_t$, is calculated as follows:

$$\mathbf{c}_t = \sum_{i}^{L} \alpha_{ti}\mathbf{a}_i \tag{10}$$

We can understand this summation of all the annotations using weight coefficients as computing an expected annotation. This weighted-sum annotation has the same length as the output sequence and can then contribute to the decoding phase.

The probability α_{ti} denotes the alignment between the target word y_t and a local region i in source image. By providing the parser with an attention mechanism, it can now determine which part of the source image to pay attention to. With this approach, it is not necessary for the watcher to encode all the information in the source image. It can also be considered as a regularization parameter for the FCN encoder because the attention helps to diminish the gradient back-propagated from the parser. The third part of the architecture in Fig. 1 shows some examples of visualized attention; other visualized attention results will be explained in detail in Section 7.3.

Finally, the output word probability can be computed as follows:

$$p(\mathbf{y}_t|\mathbf{x}, \mathbf{y}_{t-1}) = g(\mathbf{W}_o(\mathbf{E}\mathbf{y}_{t-1} + \mathbf{W}_h\mathbf{h}_t + \mathbf{W}_c\mathbf{c}_t)) \tag{11}$$

where g denotes a softmax activation function over all the words in the vocabulary, $\mathbf{W}_o \in \mathbb{R}^{K \times m}$, $\mathbf{W}_h \in \mathbb{R}^{m \times n}$, $\mathbf{W}_c \in \mathbb{R}^{m \times D}$, and $\mathbf{E}$ are learned parameters initialized randomly.

4. Very deep convnets architecture

Because convolutional neural networks have become a popular tool in the computer vision field, a number of works have been studied to improve the original architecture describe in [23] to achieve better performance. The HMEs in CROHME dataset are often complex and long while the input images are relatively large, with a maximum size of 311 × 1224 pixels. To address these large-scale input images, in this paper, we focus on the depth of improved CNNs as an important aspect of convolutional network architecture design, as in [25]. In our experiments, we fixed the other parameters of the watcher and then gradually increased the depth

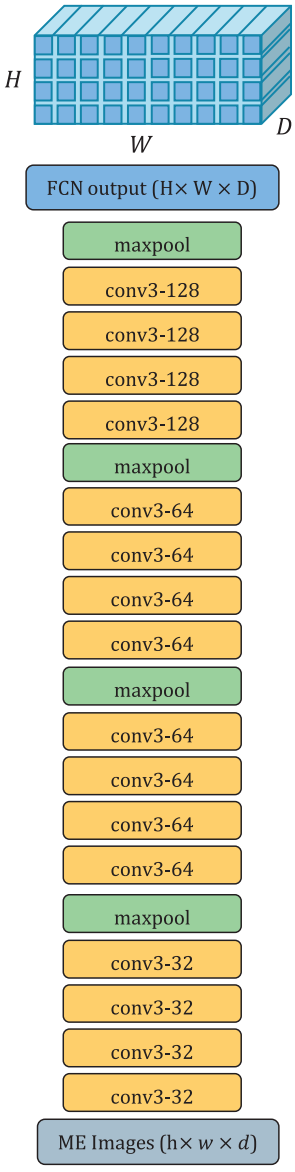

Fig. 2. FCN configurations. The convolutional layer parameters are denoted as "conv(receptive field size)-[number of channels]". For brevity, the batch normalization layer and ReLU activation function is not shown.

of the watcher network by adding more convolutional layers. The size of the convolution filters we used is small, only (3 × 3). As a result, the watcher parameters are acceptable. Finally, we determined the FCN configurations for this study, which are shown in Fig. 2.

The watcher input consists of greyscale images and 8-directional pattern images, described as a three-dimensional array of size $h \times w \times d$, where h and w are the spatial dimensions (the height and width of the image, respective) and d is the channel dimension ($d = 9$ in this study). The image is then passed through a stack of convolutional layers. Every four convolutional layers are stacked as a block. Within each block, the number of feature maps is fixed. Spatial pooling is carried out by four max-pooling layers followed by each block. All the hidden layers in this architecture are equipped with a batch normalization layer [54] and a non-linear activation layer. Here we use the ReLU [23] activation function and the batch normalization layer is performed before the activation layer. Each convolutional layer utilizes a square convolution kernel with a size of (3 × 3). The convolution stride size is fixed to (1 × 1). The four max-pooling layers are carried out on a (2 × 2) kernel, with a stride fixed to (2 × 2).

Intuitively, to process large scale images, we should use large convolution kernels in the convolution layers to extend the receptive fields. For example, [23] used an (11 × 11) kernel as the convolutional filter, with a stride of (4 × 4). Similarly, [55] adopted a (7 × 7) kernel with a stride of (2 × 2) . However, in our work, we chose a (3 × 3) convolution kernel, which possesses the smallest receptive field able to capture left/right, up/down, and centre locations simultaneously. The reason to utilize a small kernel is a stack of layers with small convolution kernels has a receptive field equivalent to a single layer with a large convolution kernel. In addition, the stack of convolutional layers imposes a regularization on the large filters of a single convolutional layer. For example, four convolutional layers (with a kernel size of (3 × 3) and a stride of (1 × 1)) stacked have a (9 × 9) effective receptive field. Assuming that both the input and the output of a four-layer (3 × 3) convolution stack have C channels, the stack is parameterized by $4(4^2 \times C^2) = 64C^2$ weights. In contrast, a single (9 × 9) convolutional layer would require $9^2 \times C^2 = 81C^2$ parameters, i.e, the layered architecture decreases the number of parameters required by about 20%. Such regularization is quite important in this study because our training dataset is insufficient. Moreover, this approach allows us to incorporate four non-linear ReLU layers rather than just one, which makes the decision function more discriminative.

5. Coverage-model

Although the attention-based encoder-decoder model is capable of jointly completing the process of watching, attending and parsing a handwritten mathematical expression, there is still a serious problem, namely, lack of coverage [27]. Coverage means the overall alignment information that indicates whether a local region of the source HME images has been translated. The past alignment information is especially important when recognizing HME; misalignment may result in over- or under-parsing. Over-parsing denotes that some parts of an HME image are unnecessarily parsed multiple times, while under-parsing means that some parts remain unparsed. To address the problem of lacking coverage, we chose to append a coverage vector to the attention model (Eq. (8)). Similar to [36], the coverage vector aims keeping track of past alignment information. We compute the coverage vector based on the sum of all past attention probabilities rather than on previous step attention probabilities. We believe that the sum of the past attention probabilities can better describe the alignment history. Fig. 3 shows a schematic representation of the coverage vector based attention model. We rewrote the attention model based on the coverage vector as follows:

$$\boldsymbol{\beta}_t = \sum_{l}^{t-1} \boldsymbol{\alpha}_l \tag{12}$$

$$\mathbf{F} = \mathbf{Q} * \boldsymbol{\beta}_t \tag{13}$$

$$e_{ti} = \boldsymbol{\nu}_a^{\mathrm{T}} \tanh(\mathbf{W}_a \mathbf{h}_{t-1} + \mathbf{U}_a \mathbf{a}_i + \mathbf{U}_f \mathbf{f}_i) \tag{14}$$

Here $\boldsymbol{\beta}_t$ is the sum of past attention probabilities, $\mathbf{f}_i$ is the coverage vector of annotation $\mathbf{a}_i$, which is initialized as a zero vector. At each time step t in the decoding phase, the previous attention probabilities (those before time step t) serve as additional input to the attention model, which provide complementary past alignment information about whether a local region of source images has been attended to. The coverage vector is produced through a convolutional layer because we think the coverage vector of annotation $\mathbf{a}_i$ should also be associated with its adjacent attention probabilities.

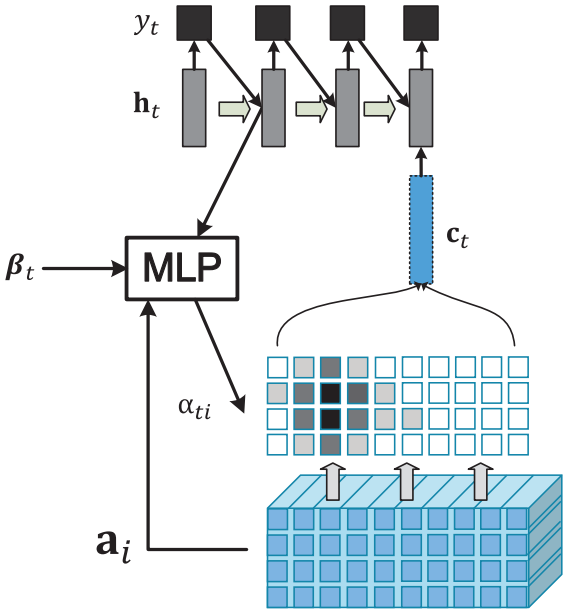

Fig. 3. Schematic representation of the coverage vector based attention model. At each time step t, an **MLP** combines the hidden state $\boldsymbol{h}_{t-1}$ and all the annotation vectors $\boldsymbol{a}_i$ with past alignment information $\boldsymbol{\beta}_t$ to compute the attention weights α_{ti}.

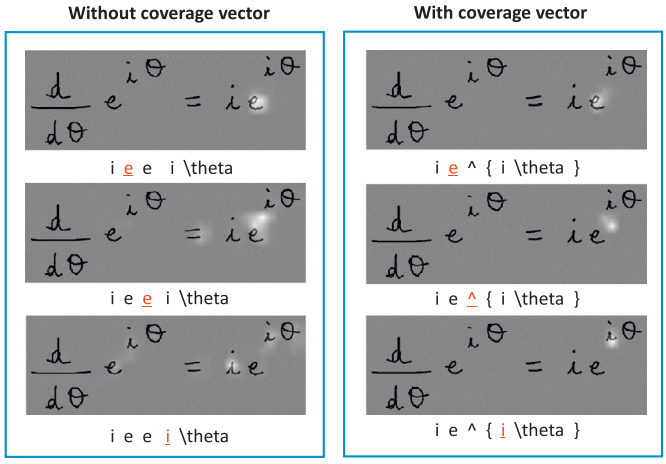

Fig. 4. Examples of attention with and without the coverage vector. The recognized LaTeX sequences of the right side of the equation are printed below each image (the white areas in the images indicate the attended regions, and the underlined text in the LaTeX sequences indicates the corresponding words).

The coverage vector is fed to the attention model to help adjust future attention. We expect the coverage vector to guide the attention model to assign higher attention probabilities to the untranslated regions of images. More specifically, the pixels of input images that have significantly contributed to the generation of target words in the past should be assigned lower attention probabilities, causing pixels with less contribution to possess higher attention probabilities in future decoding phases. Consequently, the coverage vector can guide the encoder-decoder and alleviate the problems of over- or under-parsing.

The comparison between attention with coverage vector and without coverage vector is shown in Fig. 4. In the examples without the coverage vector, the basic symbol "e" is over-parsed while the operator symbol "∧" is mistakenly unparsed. When parsing the basic symbol "i", the model focuses too much attention on the previous "i". However, the coverage model ameliorates these problems, and the alignment remains correct.

6. The training and decoding procedure

6.1. Training

The Watch, Attend and Parse model is trained jointly for end-to-end handwritten mathematical expression recognition. The image to sequence methods condition the prediction on the previous words and current input image, and the predicted word probability can be computed as shown in Eq. (11). The training objective is to maximize the predicted word probability and we use cross-entropy (CE) as the objective function.

The architecture of the watcher is shown in Fig. 2, the GRU state dimension of the parser is 256, and the embedding dimension is 256. We trained our model with batch normalization [54] to reduce the internal covariate shift. The adadelta algorithm [56] with gradient clipping was used for optimization. We employed dropout [57] to prevent overfitting: the dropout was performed only on the last 4 convolution filters and the drop ratio was set to 20%. The weight noise [58] was also implemented as a regularization measure. We annealed the best model in terms of word error rate (WER) [59] by restarting the training with weight noise.

6.2. Decoding

In the recognition stage, we aim to find the most likely LaTeX sequence given the input image:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \log P(\mathbf{y}|\mathbf{x}) \tag{15}$$

Different from training procedure, we do not have the ground truth of predicted word of the LaTeX sequence. Consequently, a simple left-to-right beam search algorithm [60] is employed to implement the decoding procedure. We maintain a set of 10 partial hypotheses, begining with the start-of-sentence $< s >$ token. At each timestep, each partial hypothesis in the beam is expanded with every possible word and only the 10 most likely beams are kept. When the $< s >$ token is encountered, it is removed from the beam and added to the set of complete hypotheses. This procedure is repeated until the output word becomes a symbol corresponding to the end-of-sentence $< /s >$.

We also adopt the ensemble method [61] to improve the performance. First we train 5 neural network models on the same training set but with different initialized parameters. Then we average their prediction probabilities on the predicted word during the beam search process.

7. Experiments

We validated our proposal using the large public dataset available from CROHME [29]. We would also analyse how WAP finished the symbol segmentation automatically and parsed the two-dimensional structure in its own way through attention visualization.

7.1. Dataset

The train set of CROHME task consists of several datasets received from the participants, which is currently the largest public dataset of handwritten mathematical expressions. The complexity and size of both the train and test sets have increased over the years. The CROHME 2014 dataset had a train set of 8836 math expressions (86K symbols) and a test set of 986 math expressions (6K symbols). There were 101 math symbol classes. None of the handwritten expressions or LaTeX notations in the test set appeared in the train set. Following other participants in CROHME 2014, we used the CROHME 2013 test set as a validation set for estimating

the models during the training. Consequently, our reported results on the competition test set are fairly comparable with others. As for the CROHME 2016 task, the train set was the same as CROHME 2014. But the test set was newly collected and labeled by the organizers at the University of Nantes. There were totally 1147 expressions and the math symbol classes remained unchanged.

Each sample of expressions is stored as an InkML file, which contains two important pieces of information: (i) the ink: a set of traces made of points and (ii) the expression level ground truth, which can be either a MathML structure or a LaTeX expression sequence. (The samples also include some information about the expression writers such as identification, handedness (left/right), age and gender; however, we did not use this additional information in this paper.)

7.2. Evaluation

The participating systems in all of the CROHME competitions were ranked by expression recognition rates, e.g. the percentage of predicted LaTeX formula sequences matching ground truth, which is simple to understand and provides a useful global performance metric. The CROHME competition compared the competing systems not only by expression recognition rates (ExpRate) but also those with at most one to three word-level errors. In our experiments, we computed these metrics by using the official tool provided by the organizer of CROHME.

It is inappropriate to evaluate an expression recognition system only at the expression level. Here, we also evaluate our system at the word level. WER [59] is a common metric for the performance of machine translation systems. Because the output of our system consists of word sequences, errors such as substitutions, deletions and insertions can always occur. The intuition behind deletions and insertions involves the path from the target to the output. For example, if the target is "1+1=2" and the output is "1+=2", we call the error a deletion; if the output is "1+1==2", we call the error an insertion; and if the output is "1+2=2", we call the error a substitution.

Word error rate can then be computed as:

$$WER = \frac{N^W_{sub} + N^W_{del} + N^W_{ins}}{N^W} = \frac{N^W_{sub} + N^W_{del} + N^W_{ins}}{N^W_{sub} + N^W_{del} + N^W_{cor}} \tag{16}$$

where

- N^W_{sub} is the number of substitutions
- N^W_{del} is the number of deletions
- N^W_{ins} is the number of insertions
- N^W_{cor} is the number of corrects
- N^W is the number of words in the target

7.3. Attention visualization

In this section, we show through attention visualization how the model is able to analyse the two-dimensional structure of ME language. The approach WAP to perform symbol segmentation automatically is also explained.

To analyse the two-dimensional structure of ME, it is essential to identify the spatial relationships between mathematical symbols, which are statistically determined and might be horizontal, vertical, subscript, superscript or inside. As shown in Fig. 5, the horizontal and vertical relationships are easy to learn by focusing on the middle symbol. When dealing with subscripts and superscripts, the parser must precisely attend to the bottom-right and upper-right directions. For inside relationships, the parser must attend to the bounding symbols. More precisely, in Fig. 6, consider the handwritten mathematical expression $(\sin(x))^2 + (\cos(x))^2$ as

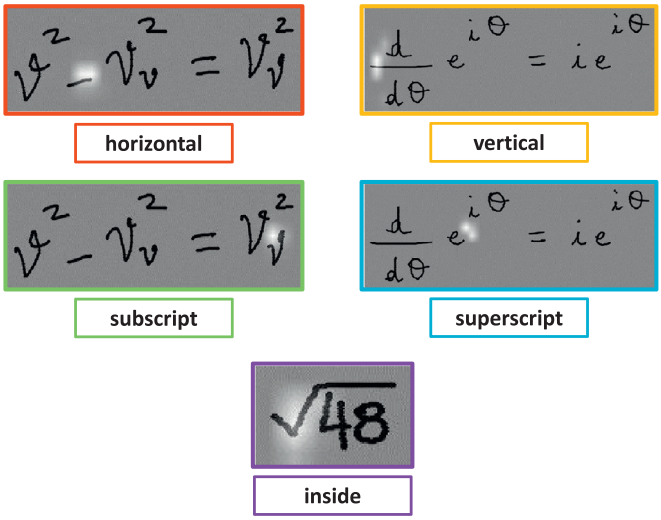

Fig. 5. The model learning procedure of determining five spatial relationships (horizontal, vertical, subscript, superscript and inside) through attention visualization.

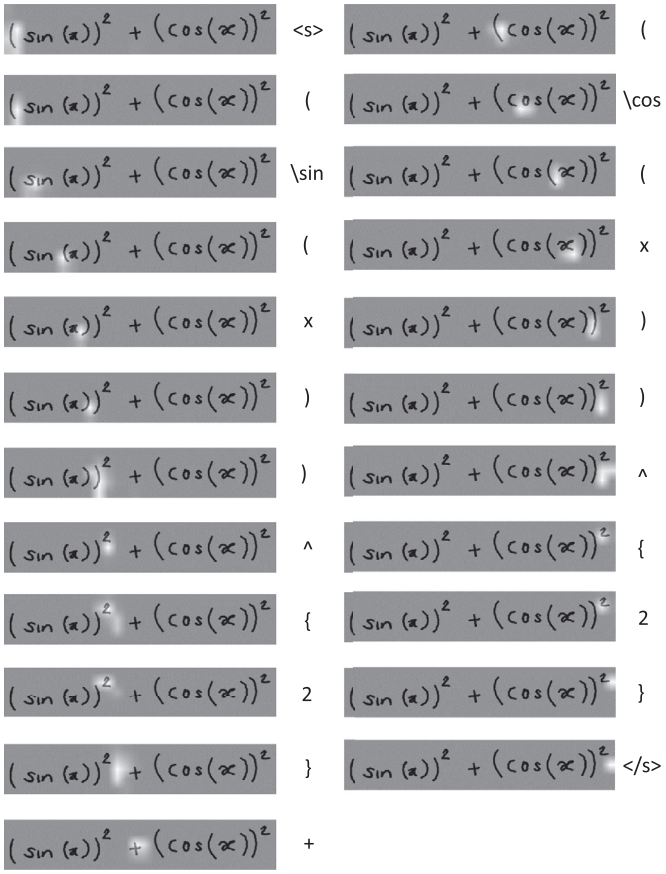

Fig. 6. Attention visualization of a tested mathematical expression image whose LaTeX sequence is " (sin (x)) ∧ { 2 } + (cos (x)) ∧ { 2 } ".

an example. We show how our system learned to translate this HME into the LaTeX sequence " (sin (x)) ∧ { 2 } + (cos (x)) ∧ { 2 } " in a step-by-step fashion through attention visualization. When encountering basic symbols such as "(", ")", "sin", "x", "cos" and "2", the model learns alignments that strongly correspond with human intuition. These correct alignments let our model automatically segment symbols by their nature. When encountering a superscript relationship, the implicit operator "∧" is parsed. More interestingly,

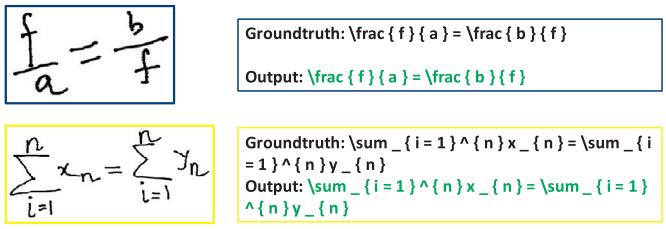

Fig. 7. Examples of ME where the LaTeX sequences are correctly generated. The green text denotes that the generated output sequence correctly matches the ground truth. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

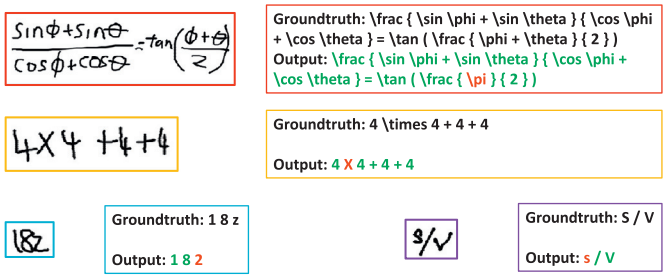

Fig. 8. Examples of ME where some errors are present in the generated LaTeX notation. The green text denotes the correct output, while red denotes the incorrect output. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Correct expression recognition rate (in %) on CROHME 2014, we erase system III because it used extra training data.

System	Correct(%)	≤ 1(%)	≤ 2(%)	≤ 3(%)
I	37.22	44.22	47.26	50.20
II	15.01	22.31	26.57	27.69
IV	18.97	28.19	32.35	33.37
V	18.97	26.37	30.83	32.96
VI	25.66	33.16	35.90	37.32
VII	26.06	33.87	38.54	39.96
WAP	**46.55**	**61.16**	**65.21**	**66.13**

we find the parser successfully generates a pair of braces "{}" right after the superscript relation being detected, which are used to compose the exponent grammar of an ME in LaTeX.

7.4. Correct and wrong examples

Fig. 7 shows two typical examples of ME, a fraction and a sum expression. The output in green denotes that our model has successfully grasped these two grammars. Fig. 8 shows several examples of common errors. Typically, most of the expression structure is preserved, but with one or two symbol recognition errors. Sometimes, errors occur when the ME is long. Many errors occur due to the ambiguities of handwritten mathematical symbols. Some symbols are difficult to distinguish when they are not written clearly, such as "times" and "X", "z" and "2". Similarities between capital letter and small letter also introduce many difficulties. However, errors such as translating "4 times 4 + 4 +4" into "4 X 4 + 4 +4" indicate the disadvantage of throwing away a predefined ME grammar. Such errors could be corrected if an extra rule was added as a postprocessing of WAP approach.

7.5. Validation on CROHME

The expression recognition rates of submitted systems from CROHME 2014 are listed in Table 1. Details of these systems can be seen in [29]. To make a fair comparison between results by different systems, here, we only show the results using the CROHME train set. Our model achieved an ExpRate of 46.55%, while its WER was only 17.73%. More training data could further improve the performance because our method requires a large amount of training data to ensure the coverage to well train a neural network. System I named "seehat" was awarded first place on CROHME 2014 competition using only the CROHME training data; its ExpRate was 37.22%. There was a large gap between the first place and the second place results, indicating a huge improvement based on the different methods. Although another system named "MyScript" achieved a higher result, that system used a large private dataset we were unable to access; consequently, we did not compare our system with theirs. Additionally, a gap existed between the correct and error percentages ($\leq$ 1%), showing that the corresponding systems have a large room for further improvements. In contrast, the small differences between error ($\leq$ 2%) and error ($\leq$ 3%) show that as additional errors are introduced, it is difficult to improve the accuracy by incorporating a single correction.

Table 2
Correct expression recognition rate (in %) on CROHME 2016, we erase team MyScript because it used extra training data.

	Correct(%)	≤ 1(%)	≤ 2(%)	≤ 3(%)
Wiris	49.61	60.42	64.69	–
Tokyo	43.94	50.91	53.70	–
São Paolo	33.39	43.50	49.17	–
Nantes	13.34	21.02	28.33	–
WAP	**44.55**	**57.10**	**61.55**	**62.34**

Table 3
The recognition performance (in %) comparison on CROHME 2014 when appending deep FCN, coverage based attention model and trajectory information to the WYGIWYS system proposed in [48].

System	ExpRate	WER	Sub	Del	Ins
WYGIWYS	28.70	32.35	11.65	13.38	7.32
+ deep FCN	35.09	28.41	9.37	13.63	5.41
+ coverage	44.42	19.40	7.11	8.76	3.52
+ trajectory	46.55	17.73	6.63	7.94	3.16

Table 2 was the same one as that was shown in Table 1 and it achieved an ExpRate of 44.55% which was quite a competitive result compared with other participating systems. The team Wiris was awarded the first place on CROHME 2016 competition using only the CROHME training data with an ExpRate of 49.61%. Please note that Wiris used a Wikipedia formula corpus, consisting of over 592,000 formulae, to train a strong language model. In our WAP approach, the language model was not used, which might be investigated as the future work. The details of other systems can be found in [62].

In Table 3, we showed the improvements via deep FCN, coverage vector and trajectory information by appending each of them to their previous system step by step. Meanwhile, the initial system WYGIWYS is proposed in [48].

First, the system "+ deep FCN" replaced the encoder of WYGIWYS with deep FCN architecture. The handwritten expressions with arbitrary lengths required a large receptive field in the watcher when encoding HME images. The deep FCN had larger receptive fields but fewer parameters than the CNN in WYGIWYS encoder. The CNN parameter size of WYGIWYS encoder was about 5.5 times of the deep FCN parameter size of our WAP encoder while deep FCN watcher increased the ExpRate from 28.70% to 35.09%.

Then, the ExpRate was improved from 35.09% to 44.42% after the coverage vector was appended into the classic attention mech-

Table 4
Sequence length vs. ExpRate and WER (in %) on CROHME 2014.

Length	ExpRate	WER
1–5	66.18	21.78
6–10	56.70	15.06
11–15	43.71	14.19
16–20	41.67	14.93
21–25	43.85	12.44
26–30	22.95	16.64
31–35	27.03	20.50
36+	11.43	26.11

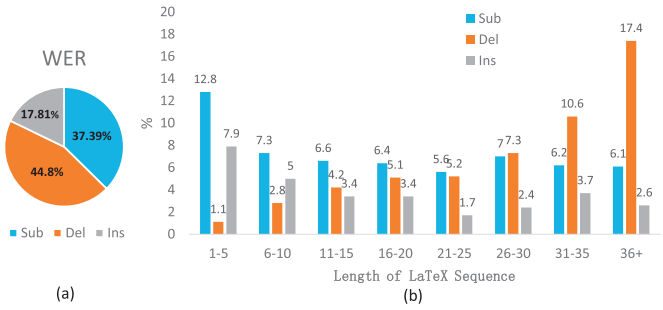

Fig. 9. Percentages of substitution, deletion and insertion errors by WER (in %) on CROHME 2014.

anism. Meanwhile, the deletion error rate decreased an absolute of 4.87%, which implied that our recognition system with the coverage model was more capable to handle longer expressions. The past alignment information helped the parser to focus on untranslated pixels. It is important to ensure that the decoding process ends when the whole input image has been translated, especially when the LaTeX sequences are quite long.

Finally, the trajectory information embedded in 8-directional pattern images could also improve the ExpRate from 44.42% to 46.55%, which played an important role in distinguishing ambiguous math symbols. For example, 'a' and 'α' are visually similar in printed expressions. However the online trajectories of writing these two symbols are quite different for the recognizer to make the correct decision accordingly.

7.6. Effects of sequence length

Another interesting analysis concerns the distribution of errors with respect to the length of ME LaTeX sequences. Intuitively, the longer the expression, the larger the size of the images. We expected the model to perform poorly on longer sequences due to the limited number of long training sequences in our training dataset. Additionally, a large image generates a corresponding large representation, which increases the difficulty for the parser. Table 4 illustrates this behaviour. However, it was surprising that the WER was lowest when the length of the LaTeX sequences ranged from 21 to 25. On short sequences (those whose length was smaller than 5) the WER scores were quite poor due to many ambiguities; additional context information is useful in helping the watcher identify symbols correctly. Fig. 9(a) shows the percentages of substitution, deletion and insertion errors that occurred in the overall WER. Substitution and deletion errors dominate the WER (37.39% and 44.80% respectively). Moreover, the percentages of substitution, deletion, and insertion errors varied as the length of the LaTeX sequence increased. As illustrated in Fig. 9(b), many deletion errors occurred in long sequences, indicating an inability to model long term dependencies. In contrast, for short sequences, substitutions and insertions were the main sources of errors. Substitution were primarily from ambiguities, while insertions suggested that the model may split words apart.

8. Conclusions

In this paper we introduce a novel system named Watch, Attend and Parse to recognize handwritten mathematical expression. It gives state-of-the-art performance on CROHME 2014 competition. We show from experiment results that WAP is capable of learning an ME grammar and dealing with symbol segmentation automatically, and demonstrate that the learned alignments correspond quite well to human intuition through attention visualization.

In future work, we plan to improve the system by making good use of online ink trajectory information and the implementation of a language model will be considered. We will also explore to apply the WAP framework to other two-dimensional languages.

Acknowledgments

The authors want to thank Junfeng Hou for insightful comments and suggestion. This work was supported in part by the National Natural Science Foundation of China under Grants 61671422 and U1613211, in part by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDB02070006, and in part by the National Key Research and Development Program of China under Grant 2016YFB1001300.

References

[1] R.H. Anderson, Syntax-directed recognition of hand-printed two-dimensional mathematics, in: Symposium on Interactive Systems for Experimental Applied Mathematics: Proceedings of the Association for Computing Machinery Inc. Symposium, ACM, 1967, pp. 436–459.
[2] A. Belaid, J.-P. Haton, A syntactic approach for handwritten mathematical formula recognition, IEEE Trans. Pattern Anal. Mach. Intell. (1) (1984) 105–111.
[3] E.G. Miller, P.A. Viola, Ambiguity and constraint in mathematical expression recognition, in: AAAI/IAAI, 1998, pp. 784–791.
[4] K.-F. Chan, D.-Y. Yeung, Mathematical expression recognition: a survey, Int. J. Doc. Anal. Recogn. 3 (1) (2000) 3–15.
[5] R. Zanibbi, D. Blostein, Recognition and retrieval of mathematical expressions, Int. J. Document Anal. Recognit. (IJDAR) 15 (4) (2012) 331–357.
[6] R. Zanibbi, D. Blostein, J.R. Cordy, Recognizing mathematical expressions using tree transformation, IEEE Trans. Pattern Anal. Mach. Intell. 24 (11) (2002) 1455–1467.
[7] F. Álvaro, J.-A. Sánchez, J.-M. Benedí, Recognition of on-line handwritten mathematical expressions using 2d stochastic context-free grammars and hidden Markov models, Pattern Recognit. Lett. 35 (2014) 58–67.
[8] A.-M. Awal, H. Mouchère, C. Viard-Gaudin, A global learning approach for an online handwritten mathematical expression recognition system, Pattern Recognit. Lett. 35 (2014) 68–77.
[9] F. Álvaro, J.-A. Sánchez, J.-M. Benedí, An integrated grammar-based approach for mathematical expression recognition, Pattern Recognit. 51 (2016) 135–147.
[10] T.H. Rhee, J.H. Kim, Efficient search strategy in structural analysis for handwritten mathematical expression recognition, Pattern Recognit. 42 (12) (2009) 3192–3201.
[11] J. Ha, R.M. Haralick, I.T. Phillips, Understanding mathematical expressions from document images, in: Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on, 2, IEEE, 1995, pp. 956–959.
[12] A. Kosmala, G. Rigoll, On-line handwritten formula recognition using statistical methods, in: Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on, 2, IEEE, 1998, pp. 1306–1308.
[13] H.-J. Lee, M.-C. Lee, Understanding mathematical expressions in a printed document, in: Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on, IEEE, 1993, pp. 502–505.
[14] H.-J. Winkler, H. Fahrner, M. Lang, A soft-decision approach for structural analysis of handwritten mathematical expressions, in: Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on, 4, IEEE, 1995, pp. 2459–2462.
[15] N.S. Hirata, F.D. Julca-Aguilar, Matching based ground-truth annotation for online handwritten mathematical expressions, Pattern Recognit. 48 (3) (2015) 837–848.
[16] P.A. Chou, Recognition of equations using a two-dimensional stochastic context-free grammar, Visual Communi. Image Process. IV 1199 (1989) 852–863.
[17] S. Lavirotte, L. Pottier, Mathematical formula recognition using graph grammar, in: Photonics West'98 Electronic Imaging, International Society for Optics and Photonics, 1998, pp. 44–52.
[18] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, arXiv preprint arXiv: 1409.0473 (2014).

[19] L. Lamport, LaTeX: A Document Preparation System: User's Guide and Reference. Illustrations by Duane Bibby, Reading, Mass: Addison-Wesley Professional. ISBN 0-201-52983-1, 1994.
[20] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, arXiv preprint arXiv: 1406.1078 (2014).
[21] J. Chorowski, D. Bahdanau, K. Cho, Y. Bengio, End-to-end continuous speech recognition using attention-based recurrent NN: first results, arXiv preprint arXiv: 1412.1602 (2014).
[22] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, in: Advances in neural information processing systems, 2014, pp. 3104–3112.
[23] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[24] K. Kawakami, Supervised sequence labelling with recurrent neural networks.
[25] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv: 1409.1556 (2014).
[26] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.
[27] Z. Tu, Z. Lu, Y. Liu, X. Liu, H. Li, Modeling coverage for neural machine translation, arXiv preprint arXiv: 1601.04811 (2016).
[28] M.-T. Luong, H. Pham, C.D. Manning, Effective approaches to attention-based neural machine translation, arXiv preprint arXiv: 1508.04025 (2015).
[29] H. Mouchere, C. Viard-Gaudin, R. Zanibbi, U. Garain, ICFHR 2014 competition on recognition of on-line handwritten mathematical expressions (CROHME 2014), in: Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on, IEEE, 2014, pp. 791–796.
[30] K.-F. Chan, D.-Y. Yeung, Error detection, error correction and performance evaluation in on-line mathematical expression recognition, Pattern Recognit. 34 (8) (2001) 1671–1684.
[31] R. Yamamoto, S. Sako, T. Nishimoto, S. Sagayama, On-line recognition of handwritten mathematical expressions based on stroke-based stochastic context-free grammar, tenth International Workshop on Frontiers in Handwriting Recognition, Suvisoft, 2006.
[32] S. MacLean, G. Labahn, A new approach for recognizing handwritten mathematics using relational grammars and fuzzy sets, Int. J. Document Anal. Recognit. (IJDAR) 16 (2) (2013) 139–163.
[33] S. MacLean, G. Labahn, A bayesian model for recognizing handwritten mathematical expressions, Pattern Recognit. 48 (8) (2015) 2433–2445.
[34] M.-T. Luong, I. Sutskever, Q.V. Le, O. Vinyals, W. Zaremba, Addressing the rare word problem in neural machine translation, arXiv preprint arXiv: 1410.8206 (2014).
[35] O. Vinyals, Ł. Kaiser, T. Koo, S. Petrov, I. Sutskever, G. Hinton, Grammar as a foreign language, in: Advances in Neural Information Processing Systems, 2015, pp. 2773–2781.
[36] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, Y. Bengio, End-to-end attention-based large vocabulary speech recognition, in: Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on, IEEE, 2016, pp. 4945–4949.
[37] W. Chan, N. Jaitly, Q.V. Le, O. Vinyals, Listen, attend and spell, arXiv preprint arXiv: 1508.01211 (2015).
[38] D.C. Ciresan, U. Meier, L.M. Gambardella, J. Schmidhuber, Deep big simple neural nets excel on handwritten digit recognition, Neural Comput. 22 (2010) 3207–3220.
[39] T. Wang, D.J. Wu, A. Coates, A.Y. Ng, End-to-end text recognition with convolutional neural networks, in: Pattern Recognition (ICPR), 2012 21st International Conference on, IEEE, 2012, pp. 3304–3308.
[40] M. Jaderberg, K. Simonyan, A. Vedaldi, A. Zisserman, Deep structured output learning for unconstrained text recognition, arXiv preprint arXiv: 1412.5903 (2014).
[41] M. Jaderberg, K. Simonyan, A. Vedaldi, A. Zisserman, Reading text in the wild with convolutional neural networks, Int. J. Comput. Vis. 116 (1) (2016) 1–20.
[42] K. Xu, J. Ba, R. Kiros, K. Cho, A.C. Courville, R. Salakhutdinov, R.S. Zemel, Y. Bengio, Show, attend and tell: neural image caption generation with visual attention., in: ICML, 14, 2015, pp. 77–81.
[43] O. Vinyals, A. Toshev, S. Bengio, D. Erhan, Show and tell: a neural image caption generator, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3156–3164.
[44] A. Karpathy, L. Fei-Fei, Deep visual-semantic alignments for generating image descriptions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3128–3137.
[45] B. Shi, X. Bai, C. Yao, An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition, IEEE Trans. Pattern Anal. Mach. Intell. (2016).
[46] R.A. Rensink, The dynamic representation of scenes, Vis. cognit. 7 (1–3) (2000) 17–42.
[47] M. Corbetta, G.L. Shulman, Control of goal-directed and stimulus-driven attention in the brain, Nat. Rev. Neurosci. 3 (3) (2002) 201–215.
[48] Y. Deng, A. Kanervisto, A.M. Rush, What you get is what you see: a visual markup decompiler, arXiv preprint arXiv: 1609.04938 (2016).
[49] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv preprint arXiv: 1412.3555 (2014).
[50] J. Zhang, J. Tang, L.-R. Dai, Rnn-blstm based multi-pitch estimation, Interspeech 2016 (2016) 1785–1789.
[51] Z.-L. Bai, Q. Huo, A study on the use of 8-directional features for online handwritten Chinese character recognition, in: Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on, IEEE, 2005, pp. 262–266.
[52] J. Du, J.-F. Zhai, J.-S. Hu, Writer adaptation via deeply learned features for online Chinese handwriting recognition, Int. J. Document Anal. Recognit. (IJDAR) 20 (1) (2017) 69–78.
[53] C.-Y. Lee, S. Osindero, Recursive recurrent nets with attention modeling for OCR in the wild, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2231–2239.
[54] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv: 1502.03167 (2015).
[55] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun, Overfeat: integrated recognition, localization and detection using convolutional networks, arXiv preprint arXiv: 1312.6229 (2013).
[56] M.D. Zeiler, Adadelta: an adaptive learning rate method, arXiv preprint arXiv: 1212.5701 (2012).
[57] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting., J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[58] A. Graves, Practical variational inference for neural networks, in: Advances in Neural Information Processing Systems, 2011, pp. 2348–2356.
[59] D. Klakow, J. Peters, Testing the correlation of word error rate and perplexity, Speech Commun. 38 (1) (2002) 19–28.
[60] K. Cho, Natural language understanding with distributed representation, arXiv preprint arXiv: 1511.07916 (2015).
[61] T.G. Dietterich, Ensemble methods in machine learning, in: International Workshop on Multiple Classifier Systems, Springer, 2000, pp. 1–15.
[62] H. Mouchère, C. Viard-Gaudin, R. Zanibbi, U. Garain, ICFHR 2016 CROHME: competition on recognition of online handwritten mathematical expressions, in: International Conference on Frontiers in Handwriting Recognition (ICFHR), 2016.

Jianshu Zhang received the B.Eng. degree of EEIS from USTC in 2015. He is currently a Ph.D. candidate of USTC. His current research area is handwriting mathematical expression recognition and neural network.