Chapter 17
Handwritten Mathematic Expression Conversion to Docx

Bharti Sharma, Tripti Rathee, Minakshi Tomer, and Parvinder Singh

Abstract This paper aims to embed Handwritten Mathematical Expressions (HME) directly into a Docx document. Writing Mathematical Expressions within a WYSIWYG (what you see is what you get) editor is a cumbersome task which requires a lot of manual effort, which this paper tries to automate. Methods: The task of Recognizing Mathematical Expression is bifurcated into two sub-tasks i.e. structural analysis and symbol recognition. This paper proposes to use deep learning techniques to do these sub-tasks using an end-to-end Densenet based encoder and Attention-Based decoder model, respectively. Findings: The model is trained on CROHME (Competition on Recognition of Online Handwritten Mathematical Expressions) dataset which consists of InkML files. These InkML files are initially processed to generate images and MathML from them. Novelty: We have been successful in creating docx from HME with accuracy trade-off of 1–2% by significantly reducing computational complexity than any other Web application based pre-existing techniques.

17.1 Introduction

Mathematics is called the "handmaiden of science", hence it plays a pivotal role in all scientific research done. Mathematics always presents itself in the form of equations, and Handwritten Mathematical expressions are the primary method of writing equations, which later is encoded in LaTeX or mathML for proper rendering on digital documents. However, automatically recognizing and converting them to

B. Sharma · T. Rathee (✉) · M. Tomer
Maharaja Surajmal Institute of Technology, New Delhi, India
e-mail: rathee.tripti@gmail.com

B. Sharma
e-mail: bhartisharma@msit.in

P. Singh
Deenbandhu Chhotu Ram University of Science and Technology, Murthal, India

R. Kountchev et al. (eds.), *New Approaches for Multidimensional Signal Processing*,
Smart Innovation, Systems and Technologies 385,
https://doi.org/10.1007/978-981-97-0109-4_17

an appropriate format remains a difficult task because of the nature of Handwritten Mathematical Expressions (HMEs). These problems include the two-dimensional nature of HMEs [1, 2] i.e. it tends to be related in a spatially.

Solving the HMEs problem can be broken down into two major stages, symbol recognition and structural analysis. Structural analysis may be done in two ways, which is sequentially and globally. Sequential analysis [3] first deals with symbol recognition and then proceeds to structural analysis. Whereas the global approach tends to deal with both of them at the same time. Sequential analysis and global analysis come with their own share of problems such as they require prior knowledge about the type of expressions to be generated for generating the parser. The complexity of the parser increases with increase in symbols dealt by the parser. They do not take into consideration the semantic context among associated symbols to deal with ambiguous symbols in case of sequential parsers.

In the last decade or so, encoder-decoder model have been utilized to solve the problem of HMEs because of its application in machine translation [4]. We propose a variation of encoder-decoder model which requires less time and has less complexity during the training phase. The model is trained to take as input images of HMEs and produce mathML strings which can be directly embedded within any word processor that accepts mathML. The overall goal of this is to generate document file such as .docx which a WYSIWYG text editor containing the mathematical expression since complex mathematical expressions are more often required within scientific documents and writing mathematical equations within the document with the present methods of manually adding expressions within the document is a hassle which we are trying to solve. MathML is a XML based document and since most text editors are XML based indirectly the conversion is a trivial task provided that the text editor supports that symbol and there exists a XLTS transformer that can convert MathML to a format used by the word processor.

The rest of the paper is organized as follows: Sect. 17.2 describes the related work summary. Section 17.3 describes the proposed methodology. Section 17.4 discusses the results and comparison, and finally Sect. 17.5 concludes this paper.

17.2 Related Work Summary

HMER consists of two elemental components that are symbol recognition and structural analysis. Provided the two dimensional nature of a mathematical expression for its structural analysis, many researchers prefer approaches based on predefined grammars as natural way to solve the problem. Several types of math grammars have been scrutinized. Chan and Yeung [5] have used definite clause grammars in their paper. However, their system works only on online mathematical expression; they have not demonstrated it on offline data set. The authors in [7] showed the fruitfulness of stochastic context-free grammars on various systems as they typically performed great in the CROHME competitions. Approaches based on probabilistic context-free grammars analyses the structure of mathematical expression and deals

with ambiguities in handwritten data, such an approach based on PCFG was proposed by [6, 8]. However, the proposed approach deals with only online maths expressions and in their future work they intend to apply it in offline mathematical expression recognition for both printed and hand written. The authors in [9] have proposed a novel neural network framework, namely encoder-decoder for sequence to sequence learning. The encoder decoder model has many applications including [10–12].

17.3 Methodology

17.3.1 Overview

The encoder within our model is a pretrained Densenet [13] model with two subsequent Fully Convolutional Layer (FCN) [14] that results in encoded image features. The decoder is Recurrent neural network (RNN) [15] with gated recurrent units (GRU) [16] that converts the encoded image features into mathML string which is our desired output. The resultant model is (1) end-to-end trainable. (2) Produces expression based on data rather than predefined grammar (3) takes into account the semantic context of the symbol to choose the best symbol and position. The data used for the training and validations is CROHME dataset which consists of stroke metadata (pen-up, pen-down sequence) during generation of expression as well as the ground truth in the form of MathML. The flowchart of the proposed methodology is represented in Fig. 17.1.

17.3.2 Dataset Preprocessing

The handwritten expression are usually stored in images that can vary in quality and size, and image preprocessing is done to prepare images in specific format to feed into the encoder. The preprocessing includes image resizing image, center cropping and normalizing the image pixel values to keep values in range. The MathML corresponding to each image expression is stored separately with same name as image file. The MathML expression is consist of predefined tags, operator symbols, operand symbols following the pattern of one symbol at a between tags (opening and closing). MathML is a 2 dimensional representation of the input handwritten expression. The mathml expression is divided into tokens of tags, operator symbols, and operand symbols.

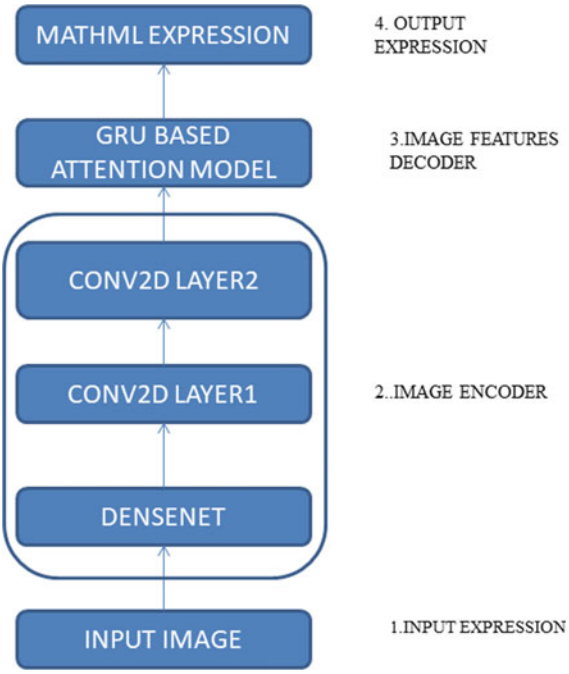

Fig. 17.1 Flowchart of proposed model

17.3.3 Encoder

The Encoder takes transformed image to convert the 3 channel image to N channel feature matrix which is an intermediate form for decoder input. The encoder is consist of Densenet and convolution layers stacked over one another. The Densenet consist of denseblocks, in each denseblock the concatenation of the outputs of preceding layers is fed as input in succeeding layers. Let $\mathbf{H_l}(.)$ denote the convolution functionof the lth layer, then the output of layer l is represented as:

$$x_l = H_l\left(\left[x_0; x_1; x_2; \ldots; x_{l-1}\right]\right) \tag{17.1}$$

where x_0, x_1, …, x_l denote the output features produced in layers 0, 1, …, l, ";" denotes the concatenation operation of feature maps.

The connections established between layers enables Densenet to use features extracted in previous layers and easy gradient propagation to initial layers. Also, this mechanism strengthens features extraction in Densenet without implementing much deeper convolution layers.

In this paper, pre trained Densenet model provided by pytorch has been used. Using pre trained Densenet has its advances as it reduces the cost of training such complex and memory consuming architecture is easier to load. The output produced by Densenet is larger in size. CNN has been largely used to reduce the size of

representational n-dimensional matrix without affecting features represented by the n-dimensional matrix. Thus, the last layers of Densenet model are removed to make model work as a feature extractor instead of a classifier. Then two convolution layers are layered over output of Densenet to reduce the size of output to optimal feature representation. The proposed model takes as input a raw expression image and generates corresponding MathML sequence.

17.3.4 Decoder

The input block of the decoder provides one-hot encoding of the input word to the embedding layer. The embedding layer converts the one-hot encoding of input word to word embedding of hidden_size, H length vector. Word embedding is an efficient way to represent relation between words in a vocabulary. Embedding is a dense vector of floating points that represents a word's features and more importantly, these features can be learned via training of the embedding layer. The working of decoder has been shown in Fig. 17.2.

Let x_i be the one hot encoding of input word and O_{en} represent the encoder output. Then, O_e represent the output of embedding layer which takes as input a vector of vocabulary size and gives output vector of H size.

$$O_{\text{emm}} = W_{\text{emm}} x_i \tag{17.2}$$

The previous hidden state h_{t-1}, a vector of size H that corresponds to the last hidden state generated by the GRU.

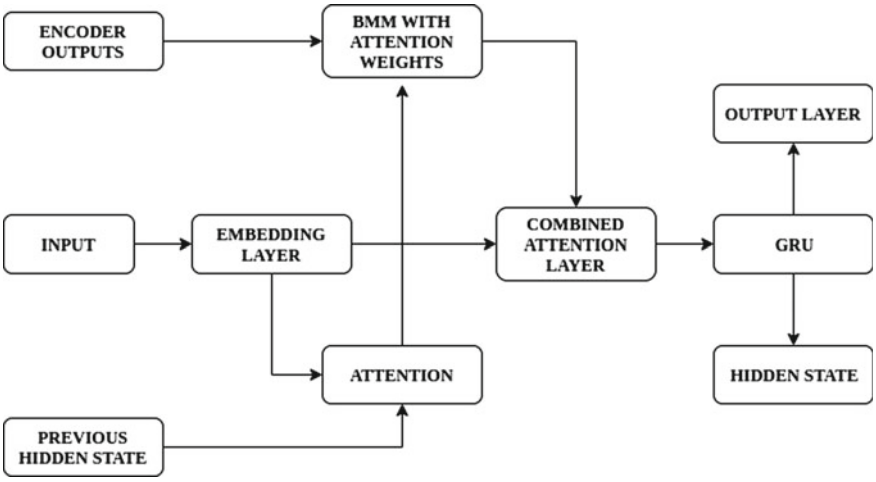

Fig. 17.2 Decoder

$$x_{\text{attn1}} = \{O_{\text{emm}}; h_{t-1}\} \tag{17.3}$$

Then, x_{attn1} is the concatenation of the embedding output and previous hidden state. Attention block which is a linear layer which takes input of size 2*H and gives output of size H is applied on x_{attn1}.

$$O_{\text{attn1}} = \text{softmax}(W_{\text{attn1}} x_{\text{attn1}} + b_{\text{attn1}}) \tag{17.4}$$

O_{attn1} represents the output of attention block which act as attention weights for the encoder output.

Softmax activation function is used to convert real values to probabilities so it can be applied on encoder output.

$$x_{\text{in}} = O_{\text{attn1}} \otimes O_{\text{en}} \tag{17.5}$$

x_{in} is the element-wise multiplication of attention weights and encoder output

$$x_{\text{out}} = \{O_{\text{emm}}; x_{\text{in}}\} \tag{17.6}$$

x_{out} represents the concatenation of embedding output and x_{in}, , which is input to second attention block called attention combined which is also a linear layer which takes input of size 2*H and gives output of size H.

$$O_{\text{attn2}} = \text{RELU}(W_{\text{attn2}} x_{\text{out}} + b_{\text{attn2}}) \tag{17.7}$$

The rectified linear activation function (RELU) is used as it is a piecewise linear function that will output the input directly if is positive; otherwise, it will output zero. O_{attn2} is the output of combined attention layer, and it is a vector of size H. O_{attn2} is the input to GRU block of the decoder.

$$x_t = O_{\text{attn2}} \tag{17.8}$$

GRU is an improved version of RNN which solves the problems of vanishing and exploding gradients. Let x_t be given input to GRU and the output h_t is computed as:

$$h_t = \text{GRU}(x_t, h_{t-1}) \tag{17.9}$$

Softmax activation function is applied on GRU output to generate vector of output probabilities, and argmax is applied to predict the output word.

17.3.5 Document

The predicted mathml is parsed to a tree structure and inserted into a word document using python libraries i.e. python-docx, xET.

17.4 Result and Comparison

This section describes the system settings for the experimentation purpose and the evaluation matrices used

17.4.1 Experimental Setup

The system is implemented on Intel(R) Core(TM) i5, 3.30 GHz CPU, 4 cores and 8 GB RAM. During training of the model the factors considered are loss and Validation.

The red line in Fig. 17.3 represents the value of Log loss, and the blue line represents validation loss.

Figure 17.4 shows decrease in loss in Red and increase in Bleu score on Test set in Blue curve with epochs.

Model comparison By Bleu Scores (see Table 17.1):

Initial predictions—<mrow><mi><mi></mi><mrow><mo><mi></mi> <mrow><mi><mo></mi></mrow>

Original value—<mrow><mi> x</mi> <mrow> <mo> + </mo> <mi> y </mi></ mrow></mrow>

The resultant output of Fig. 17.5 image comes out to be $x + y$.

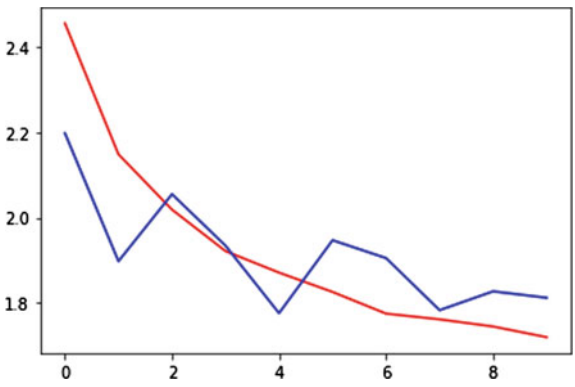

Fig. 17.3 Loss and validation graph

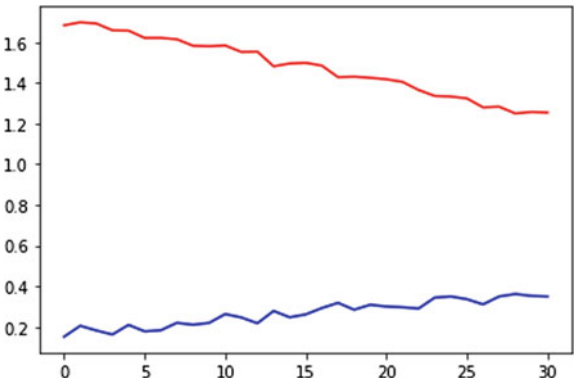

Fig. 17.4 Loss and BLEU score

Table 17.1 Comparison results

After epochs	Multi-scale model	Encoder-decoder model
10	0.36	0.32
20	0.43	0.47
30	0.55	0.56
40	0.59	0.62

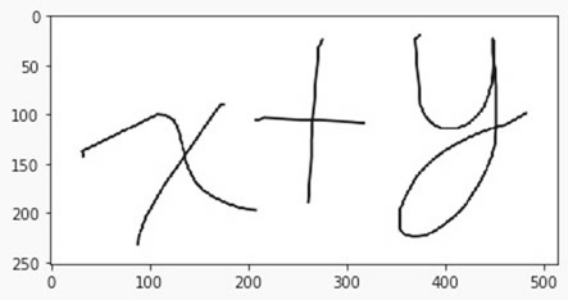

Fig. 17.5 Input handwritten expression

17.5 Conclusion

In this paper, we concluded that using a pre-trained dense encoder model we can train an attention model with features to provide good accuracy. Densenet provides better image features than most of the state of the art models present for image segmentation and feature extraction. This reduces the computational cost significantly that is used to train a Densenet. Also, the MathML conversion of feature vectors provides a base for conversion to other standard formats of mathematical expressions. Also, the GRU based Architecture of Decoder is uniquely defined and experiments have been done regarding its effectiveness.

References

1. Anderson, R.H.: 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, pp. 436–459 (1967)
2. Belaid, A., Haton, J.P.: A syntactic approach for handwritten mathematical formula recognition. IEEE Trans. Pattern Anal. Mach. Intell. **1**, 105–111 (1984)
3. Zanibbi, R., Blostein, D., Cordy, J.R.: Recognizing mathematical expressions using tree transformation. IEEE Trans. Pattern Anal. Mach. Intell. **24**(11), 1455–1467 (2002)
4. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation (2014). arXiv:1406.1078
5. Chan, K.F., Yeung, D.Y.: Error detection, error correction and performance evaluation in on-line mathematical expression recognition. Pattern Recogn. **34**(8), 1671–1684 (2001)
6. Álvaro, F., Sánchez, J.A., Benedí, J.M.: An integrated grammar-based approach for mathematical expression recognition. Pattern Recogn. **51**, 135–147 (2016)
7. Sako, S., Nishimoto, T., Sagayama, S.: On-line recognition of handwritten mathematical expressions based on stroke-based stochastic context-free grammar. In: Tenth International Workshop on Frontiers in Handwriting Recognition. Suvisoft (2006)
8. MacLean, S., Labahn, G.: A new approach for recognizing handwritten mathematics using relational grammars and fuzzy sets. Int. J. Doc. Anal. Recognit. (IJDAR) **16**, 139–163 (2013)
9. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate (2014). arXiv:1409.0473
10. Bahdanau, D., Chorowski, J., Serdyuk, D., Brakel, P., Bengio, Y.: End-to-end attention-based large vocabulary speech recognition. In: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4945–4949. IEEE (2016)
11. Chan, W., Jaitly, N., Le, Q.V., Vinyals, O.: Listen, attend and spell (2015). arXiv:1508.01211
12. Luong, M.T., Sutskever, I., Le, Q.V., Vinyals, O., Zaremba, W.: Addressing the rare word problem in neural machine translation (2014). arXiv:1410.8206
13. Iandola, F., Moskewicz, M., Karayev, S., Girshick, R., Darrell, T., Keutzer, K.: Densenet: implementing efficient convnet descriptor pyramids (2014). arXiv:1404.1869
14. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440 (2015)
15. Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., Khudanpur, S.: Recurrent neural network based language model. In: Interspeech, vol. 2, no. 3, pp. 1045–1048 (2010)
16. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling (2014). arXiv:1412.3555