Deep Neural Networks for Recognizing Online Handwritten Mathematical Symbols

Hai Dai Nguyen[1], Anh Duc Le[2] and Masaki Nakagawa[3]
Tokyo University of Agriculture and Technology
2-24-16 Nakacho, Koganei-shi, Tokyo
[1]haidnguyen0909@gmail.com, [2]leducanh841988@gmail.com, [3]nakagawa@cc.tuat.ac.jp

Abstract

This paper presents application of deep learning to recognize online handwritten mathematical symbols. Recently various deep learning architectures such as Convolution neural network (CNN), Deep neural network (DNN) and Long short term memory (LSTM) RNN have been applied to fields such as computer vision, speech recognition and natural language processing where they have been shown to produce state-of-the-art results on various tasks. In this paper, we apply max-out-based CNN and BLSTM to image patterns created from online patterns and to the original online patterns, respectively and combine them. We also compare them with traditional recognition methods which are MRF and MQDF by carrying out some experiments on CROHME database.

1. Introduction

Online mathematical symbol recognition is an essential component of any pen-based mathematical formula recognition systems. With the increasing availability of touch-based or pen-based devices such as smart phones, tablet PCs and smart boards, interest in this area has been growing. However, the existing system are still far from perfection because of challenges that arise from the two-dimensional nature of mathematical input and the large symbol set with many similar looking symbols. In this work we just address the problem of mathematical symbol recognition. Handwritten character pattern recognition methods are generally divided into two types: online and offline recognition. Online recognition recognizes character patterns captured from a pen-based or touch-based input device where trajectories of pen-tip or finger-tip movements are recorded, while offline recognition recognizes character patterns captured from a scanner or a camera device as two dimensional images.

For online symbol recognition, structure features such as sampling points, line segments or stroke orders are often employed with Hidden Markov models (HMM) or Markov Random field (MRF) [1]. However, since the un-structure features such as direction, gradient histogram or projection features are easily extracted from an online pattern by discarding temporal and structural information, we can also apply the offline recognition methods such as Support vector machine (SVM) or Modified Quadratic Discriminant Function (MQDF). Therefore, we can combine online and offline recognition methods.

Long Short Term Memory Recurrent Neural Networks (LSTM RNNs) have been successfully applied to sequence prediction and labeling tasks. For online handwriting recognition, bidirectional LSTM (BLSTM) networks with a connectionist temporal classification (CTC) output layer using a forward backward type of algorithm have been shown to outperform state-of-the-art HMM-based systems in handwriting recognition [2]. Recently, Álvaro et al. proposed a set of hybrid features that combine both on-line and off-line information and using HMM and BLSTM for online handwritten mathematical symbols [3]. The symbol recognition rate achieved using raw images as local off-line features along the pen-tip trajectory by BLSTM significantly outperformed HMM. However, the hybrid features in combination with BLSTM did not produce the great improvement observed in HMM . In section 2 we describe the way of using gradient features as local off-line features following the method of Kawamura et al. [4] and show that the recognition rate is improved by adding more gradient features when combined with BLSTM.

Following the recent success of Convolutional Neural Network (CNN) in many recognition tasks [5] and the popularity of Deep Learning, a number of different nonlinearities (activation function) have been proposed for Deep Neural Network training. A nonlinearity that has recently become popular is the Rectified Linear Unit (ReLU) , which is a simple activation function $y = \max(0, x)$ and the maxout nonlinearity [6], which could be regarded as a generalization of ReLU, was proposed. Maxout network, combined with "dropout" [7], has also achieved improvement in computer vision tasks [6]. In this paper, we employ maxout function for both Convolution and Full-Connected layers along with drop out to build up a Deep Maxout Convolutional Network (DMCN).

The combination of multiple classifiers has been shown to be suitable for improving the recognition performance in difficult classification problems [8]. In this work, we employ simply linear combination of DMCN and BLSTM. Our experiments also show that the best combination

ensemble has a recognition rate which is significantly higher than the rate achieved by the best individual classifier and the best previous methods on the CROHME database.

2. Bidirectional Long Short Term Memory

This section outlines the principle of BLSTM RNN used for online symbol classification task and our proposed local gradient features to improve the accuracy of BLSTM.

2.1. Overview of LSTM RNN

RNNs are a connectionist model containing a self-connected hidden layers. The recurrent connection provides information of previous inputs such that the network can benefit from past contextual information. The LSTM advanced RNN allows that cells can access to context information over long periods of time. This is achieved by using a hidden layer composed of recurrently connected subnets, called memory blocks (see Fig. 2)

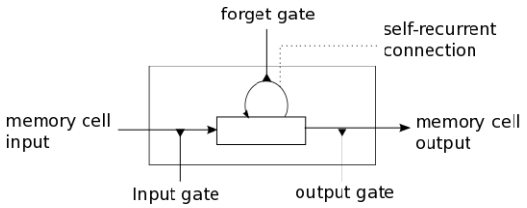


Figure 1: LSTM memory block consisting of one memory cell: The input, output, and forget gates collect activations from inside and outside the block which controls the cell through multiplicative units (depicted as small circles).

Another problem with standard RNNs is that they have access to past but not to future context. This can be overcome by using bidirectional RNNs [9], where two separate recurrent hidden layers scan the input sequences in opposite directions. The two hidden layers are connected to the same output layer, which therefore has access to context information in both directions. The amount of context information that the network actually uses is learned during training, and does not have to be specified beforehand.

2.2. Feature Extraction for BLSTM

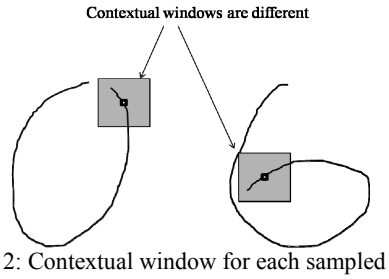


Figure 2: Contextual window for each sampled point.

One of problems of online classifiers is how to classify patterns having the similar stroke or sampled point order. For example, Figure 3 illustrates the case of two symbols '0' and '6'. Using a contextual window around each sampled point is an obvious way to overcome this problem. We can use various ways to extract features from these windows such as the approach presented by Alvaro at el. [3]. They used raw images centered at each point to present the context information and then PCA for dimension reduction. However the recognition rate has not been improved when used with BLSTM for online features since the classifier may not process effectively features of raw images. In this work, we employed the gradient feature, which performed well in mathematical character recognition.

Firstly, linear normalization (LN) is used to convert each online character pattern to offline pattern of standard size (64x64). Then for each point p = (x, y), we use 6 time-based features: end point or not (1 or 0); normalized coordinates (x, y); derivatives: (x', y'); distance between point d(i, i+1). In order to combine these online features with context information around each point, we employ the gradient direction feature as context features.

Regarding gradient features: For each point p = (x, y), we employ a context window centered at p. From the context window, gradient direction features are decomposed into components in 8 chain code directions (depicted as figure 4). We partition the context window into 9 sub-windows of equal-sizes 5x5. We calculate the value for each block by using a Gaussian blurring mask of size 10x10 and finally use PCA to reduce the dimension into 20 dimensions.

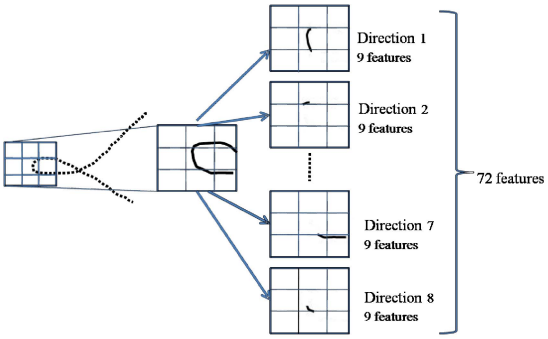


Figure 3: Extracting gradient features.

3. Convolutional Neural Network (CNN)

This section describes the basic structure of CNN and proposed maxout units [6].

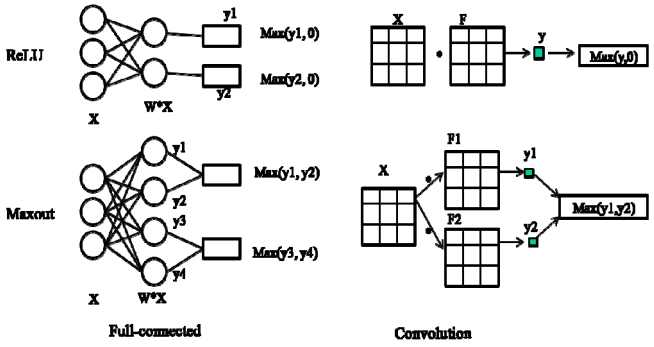
3.1. CNN

A CNN consists of several layers which can be of three types: convolutional, pooling and full-connected layers. Convolutional layers consist of a rectangular grid of neurons. Each of them takes inputs from a rectangular section of the previous layer; the weights for this

rectangular section are the same for each neuron in the convolutional layer. Thus, it is just an image convolution of the previous layer, where the weights specify the convolution filter. After each convolutional layer, there may be a pooling layer that takes small rectangular blocks from the convolutional layer and subsamples it to produce a single output from that block. There are several ways to do this pooling, such as taking the average or the maximum, or a learned linear combination of the neurons in the block. Our pooling layers are always max-pooling layers; that is, we take the maximum of the block they are pooling. Finally, after several convolutional and pooling layers, the high-level reasoning in the neural network is done via full-connected layers that take all neurons in the previous layer.

3.2. Deep Maxout Convolutional Network

DMCNs consist of multiple layers which generate hidden activations via the maxout function. Maxout function groups the linear activations in detection layer and passes forward the maximum value within each group as described in Fig 4a instead of applying any form of non-linearity as ReLU or sigmoid.



(a) (b)

Figure 4: A scheme of a single maxout layer with the pool size K = 2 for (a) full-connected and (b) convolutional layer. Layers can be stacked on each other to form deeper structures.

In a convolutional layer, a maxout feature map can be constructed by taking the maximum across *k* affine feature maps (in figure 4b). A DMCN can be constructed by connecting multiple maxout layers consecutively and finally adding the softmax layer. We use the dropout technique and max-pooling layers after each convolutional layer which sparsifies the activation values maybe a desirable.

4. Experiments

4.1. Dataset

CROHME is a contest for online handwritten mathematical expression (ME) recognition, initially organized at ICDAR 2011 [11]. The sample pattern dataset is selected from 5 different MEs databases. The dataset contains 8,836 MEs for training as well as 761 MEs and 987 MEs for testing in the 2013 version and the 2014 version, respectively. In the last and current competition, there are 101 symbol classes. We extract isolated symbols from them and for a datasets of isolated symbols, which we call SymCROHME. The detailed information of this dataset is listed in Table 1. SymCROHME consists of various kinds of symbols shown in Figure 7.

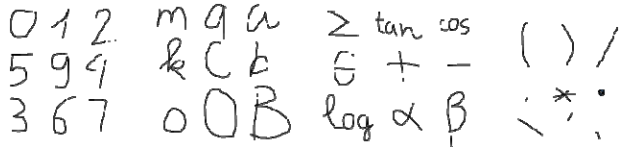


Figure 5: Some samples in CROHME database.

Set name	TrainCROHME	TestCROHME	
		2013	2014
#symbols	120,341	6,080	9,999

Table 1:The detailed information of SymCROHME datasets.

A random 10% of the training set is held out as the validation set, which is used for validation purposes, and for tuning the meta-parameters.

4.2. Results

We use two recognition schemes which are based on MQDF and MRF to compare with DMCN and BLSTM in our experiments, corresponding to off-line and online methods, respectively. In offline handwritten-character recognition, line-density equalization based on nonlinear normalization has been proven effective when used with MQDF; accordingly, the three normalization methods including LN, P2DBMN and LDPI were used before the feature-extraction stage in the MQDF-based system and DMCN.

The MQDF based recognizer represents each character sample as a 150-dimensional feature vector. It converts every input pattern to a 64x64 grid by one of three normalization methods mentioned above, and smoothes it by a connectivity-preserving procedure. Then, it decomposes the normalized image into eight contour sub-patterns, one for each main direction. Finally, it extract 64-dimensional feature vector for each contour pattern from their convolution with a blurring mask(Gaussian filter) so 512 in total. This feature vector will be reduced to 150 in our experiments before fed into MQDF developed by Kimura[12]. The parameters of MQDF are chosen according to Phan et al [13] to obtain the best performance.

The architecture of DMCN contains nine learned layers-six convolution + maxpooling (32, 64, 96 filters,

respectively), two full connected layers (1024 and 512 neurons, respectively) and a softmax layer finally (101 classes). The network are trained by using stochastic gradient descent with a batch size of 64, momentum of 0.95 on GPU. We initialized the weights in each layer from a zero-mean Gaussian distribution with standard deviation 0.01 and the biases with the constant 0.

The results listed in Table 4 show that DMCN outperforms MQDF significantly when LN and P2DBMN are used and slightly when LDPI is used.

(a)TestCROHME2013

Recog. method	Norm. method	N-best rate (%)		
		1st	3rd	5th
MQDF	**LN**	81.68	94.67	97.22
	P2DBMN	82.40	94.47	97.22
	LDPI	83.92	96.51	98.24
DMCN	**LN**	84.93	97.47	98.78
	P2DBMN	85.18	97.22	98.40
	LDPI	84.21	97.17	98.24

(b)TestCROHME2014

Recog. method	Norm. method	N-best rate (%)		
		1st	3rd	5th
MQDF	**LN**	81.91	91.23	96.07
	P2DBMN	82.20	93.51	97.26
	LDPI	83.24	94.18	97.75
DMCN	**LN**	89.39	97.83	98.78
	P2DBMN	87.39	97.08	98.30
	LDPI	88.85	97.63	98.70

Table 2:Accuracy of MQDF and DMCN on TestCROHME 2013 and 2014 in the case of three normalization methods

For BLSTM, the size of input is six for online features and 26 for a combination of online and off line features. Two hidden BLSTM layers(with 32 and 128 memory blocks per layer) are used. The output layer is a softmax layer, which size is 101, namely, the number of math-symbol classes. The weights of the network are initialized from a zero-mean Gaussian distribution with standard deviation of 0.1. The network is trained by using online stochastic gradient descent with learning rate of 0.0001 and momentum of 0.9. The training algorithm is stopped when the error rate is not improved for 20 epochs.

To check the effect of gradient features on BLSTM, three BLSTM networks were experimentally evaluated. The first net only uses six online features (mentioned in Section 2.2) for each point, the second net only uses reduced gradient features, and the final uses hybrid features which are combinations of online features and gradient features. BLSTM with hybrid features outperforms that with only on-line features or gradient features (better by about 0.9%) The reason for this superior performance is because the proposed gradient features include more context information on each point.

In the next experiment BLSTM are compared with an MRF model with weighting parameters optimized by CRFs which are successfully applied for online recognition of handwritten Japanese characters [1]. The model effectively integrates unary and binary features and introduces adjustable weighting parameters to the MRF's, which are optimized according to CRF. The proposed method extracts feature points along the pen-tip trace from pen-down to pen-up and matches those feature points with states for character classes probabilistically based on this model. Experimental results demonstrated the superiority of the method and that MRFs exhibited higher recognition accuracy than HMMs on Japanese characters as well as Chinese characters. The parameters are maintained as previous experiments [1]. For simplicity, LN is used to normalize online patterns to a size of 48x48. Next, the recognition accuracies of BLSTM and MRF were experimentally compared. As shown by the accuracies listed in Table 4, BLSTM outperforms MRF.

Method	Features	Acc. Rate (%)
BLSTM+ Online features	End point(2) or not(1), x, y, x', y', d	86.8
BLSTM+ gradient features	20 gradient features	86.9
BLSTM+Online+ gradient features	6 online features + 20 gradient features	87.7

Table 3:Results obtained by BLSTM with different features on testing set TestCROHME2014

Dataset	TestCRHOME2013			TestCRHOME2014		
Method	N-best rate (%)			N-best rate (%)		
	1st	3rd	5th	1st	3rd	5th
MRF	76.53	90.72	93.29	81.40	93.91	95.94
BLSTM	83.44	95.35	97.09	87.29	95.70	97.61

Table 4:Comparative results concerning MRF and BLSTM applied to both of testing set

Combining multiple individual classifiers is proven to be a suitable way to improve recognition rate for difficult classification problems [8]. Many sophisticated methods can be combined, but in this work, linear combination is employed for simplicity as the following formulation:

$$Score_{Combine} = \alpha\ Score_{offline} + (1-\alpha)\ Score_{online} \quad (2)$$

where $Score_{offline}$ is the softmax output of DMCN , and $Score_{online}$ is the softmax outputs of BLSTM, while α is the weighting parameter for combination. This parameter is estimated by an experiment using a validation set.

Combined DMCN and BLSTM was experimentally compared with individuals. A validation set was used to tune the weighting parameter, and best values of 0.65 were obtained for the combination of DMCN and BLSTM. The results listed in Tables 5 show that combining classifiers significantly improves recognition rate when compared

with individual classifiers. In particular, the combined DMCN and BLSTM improves accuracy by 2.5% and 1.6% on TestCROHME 2013 and 2014, respectively, compared to that of the best individual method. Moreover, we also compare our system with others published in [11][14]. As shown in Table 6, our system outperform slightly the system Ⅰ which is currently holding the-state-of-the-art on TestCRHOME 2013 and 2014.

Methods	TestCRHOME2013	TestCRHOME2014
	1-best rate (%)	1-best rate (%)
DMCN	84.93	89.39
BLSTM	83.44	87.29
DMCN+BLSTM ($\alpha = 0.65$)	**87.39**	**91.08**

Table 5:Accuracy (top 1) of combined DMCN and BLSTM when compared with individuals on both of testing set

Dataset	TestCRHOME2013	TestCRHOME2014
Systems	1-best Rate (%)	1-best Rate (%)
Ⅰ	87.10	91.24
Ⅱ		82.72
Ⅲ		91.04
Ⅳ		88.66
Ⅴ		85.00
Ⅵ		84.31
Ⅷ		77.25
Ours	**87.35**	**91.28**

Table 6: Accuracy(top 1) of our best system when compared with other systems on both of testing set without Junk (101 classes)(results from competition and F. Álvaro et al [11][14])

5. Conclusion

This paper studied Deep learning for recognizing online handwritten mathematical symbols. Following experiments on the CROHME database, we draw the following conclusions: 1) compared with MQDF, Deep neural network can improve the offline recognition of mathematical symbols because it may extract wider yet specific features; 2) as for online methods, BLSTM outperform MRF because it can access the whole context of input sequence flexibly; 3) Combining both online and offline recognition methods improves classification performance by taking their advantages. Finally our future work will be focused on the integration of this classifier in a mathematical expression recognition system, where the recognition of the whole system will help to solve the problem of similar shaped classes.

References

[1] B. Zhu, M. Nakagawa, Online Handwritten Japanese Characters Recognition Using a MRF Model with Parameter Optimization by CRF, Proc. 10th ICDAR, Beijing, China, 18-21 Sept. 2011, pp.603-607.

[2] A. Graves, Supervised Sequence Labeling with Recurrent Neural Networks, Studies in Computational Intelligence, Springer, 2012.

[3] F. Álvaro, J.-A. Sanchez, J.-M. Benedi, Classification of On-line Mathematical Symbols with Hybrid Features and Recurrent Neural Networks, Proc. 12th ICDAR, Washington DC, USA, 25-28 Aug. 2013, pp.1012-1016.

[4] A. Kawamura, K. Yura, T. Hayama, Y. Hidai, On-line recognition of Freely Handwritten Japanese Characters using Directional Feature Densities, Proc. 11th ICPR, vol. 2. Conference B: Pattern Recognition Methodology and Systems, The Hague, 30 Aug-3 Sep 1992, pp.183-186.

[5] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based Learning Applied to Document Recognition, Proc. of the IEEE, Vol. 86, Issue 11, 1998, pp.2278–2324.

[6] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, Maxout networks, arXiv: 1302.4389, 2013.

[7] G. E. Hinton, N. Srivastava, A. Krizhevsky ,I. Sutskever, and R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, The computing research repository, 2012.

[8] J. Kittler, M. Hatef, R. Duin, and J. Matas, On Combining classiffiers, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol.20, no.3, 1998, pp.226-239.

[9] M. Schuster and K. K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. on Signal Processing, vol. 45, no. 11, Nov. 1997, pp. 2673-2681.

[10] A. Graves, RNNLIB: A recurrent neural network library for sequence learning problems, http://sourceforge.net/projects/rnnl/, 2013.

[11] H. Mouchere, C. V. Gaudin, R. Zanibbi, U. Garain, ICFHR 2014 Competition on Recognition of online Handwitten Mathematical Expressions (CROHME 2014), Crete, Greece, 1-4 Sep. 2014.

[12] F. Kimura, K. Takashina, S. Tsuruoka, Y. Miyake, Modified quadratic discriminant functions and the application to Chinese character recognition, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol.9, no. 1, 1987, pp.149-153.

[13] T. V. Phan, J. Gao, B. zhu, M. Nakagawa, Effects of line densities on nonlinear normalization for online handwritten Japanese character recognition, Proc. 10th ICDAR, Beijing, China, 18-21 Sept. 2011, pp.834-838.

[14] F. Álvaro, J.-A. Sanchez, J.-M. Benedi, Offline features for classifying Handwritten Math Symbols with Recurrent Neural Network, Proc. 22nd ICPR, Stockholm, Sweden, 24-28 Aug. 2014, pp.2944-2949.