Offline Handwritten English Character Recognition based on Convolutional Neural Network

Aiquan Yuan, Gang Bai, Lijing Jiao, Yajie Liu
College of Information Technical Science
Nankai University
Tianjin City, China
{*yuanaiquan123, jiaolijing0612, liuyajie1988*}*@mail.nankai.edu.cn, baigang@nankai.edu.cn*

***Abstract*—This paper applies Convolutional Neural Networks (CNNs) for offline handwritten English character recognition. We use a modified *LeNet-5* CNN model, with special settings of the number of neurons in each layer and the connecting way between some layers. Outputs of the CNN are set with error-correcting codes, thus the CNN has the ability to reject recognition results. For training of the CNN, an error-samples-based reinforcement learning strategy is developed. Experiments are evaluated on UNIPEN lowercase and uppercase datasets, with recognition rates of 93.7% for uppercase and 90.2% for lowercase, respectively.**

***Keywords*- Handwritten English Character Recognition; Convolutional Neural Networks; Error-Correcting Code; Error-Samples-Reinforcement-Learning;**

I. Introduction

Handwritten English Character Recognition (HECR) has been a fairly challenging research topic in Optical Character Recognition (OCR). Up to now, there have been lots of fruitful researches for HECR [1][2][3][4]. However, most of them are carried out with online information, or with online and offline hybrid classifier; researches with pure offline information are rare [1][2][4][5][6]. As handwritten characters are unconstrained and topologically diverse, HECR with pure offline information has much difficulty.

In 1995, Convolutional Neural Networks (CNNs) was brought about by LeCun and caused huge attention immediately [7]. In a CNN recognition system, 2-D image can be directly input and feature extraction is thus avoided. Many experiments with the CNN have seen moderately good performance. However, until now, most researches with CNNs are for handwritten digits [8][9][10][11], handwritten words [12] (used to score the segments of words with combination of HMMs), or printed character recognition [13]. We have not seen many applications for pure offline HECR with CNN.

This paper focuses mainly on offline HECR on UNIPEN dataset [14], with 26 characters for uppercase and lowercase, respectively. Section II shows the common *LeNet-5* model of CNNs and the modifications we make on it. A training strategy based on reinforcement of *error-samples* are described in section III. Experiments and discussions are provided in section IV.

II. Convolutional Neural Network

A. Basic Architecture of LeNet-5

A common model of CNNs is the *LeNet-5* model [8], as shown in Figure 1. Each unit in the *LeNet-5* model is connected to a local neighborhood in the previous layer, thus it can be seen as a local feature detector. Insensitivity to local transformations is built into the network architecture and the same features on different parts of the input are detected. The outputs of the units in the same position in different feature maps can be thought as a feature vector of the same area. Increasingly complicated features are extracted by neurons in the successive layers. Because of weight-sharing, the number of free parameters in the system is greatly reduced. CNN produces an output vector in every layer. Each dimension in the output vector detects features from different parts of feature maps in the previous layer.

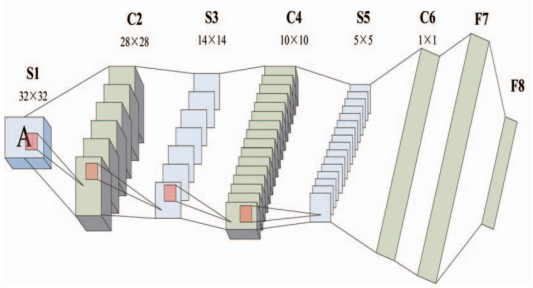

Figure 1. The basic architecture of *LeNet-5*

Layer S1 (input layer) is an image of size 32×32. Layer C2 is the first convolutional layer with 6 feature maps of size 28×28. Each unit of each feature map is connected to a 5×5 neighborhood of the input in layer S1. Layer S3 is the second subsampling layer with 6 feature maps of size 14×14. Each unit in each feature map is connected to a 2×2 neighborhood in the corresponding feature map in layer C2. Layer C4 is the second convolutional layer with 16 feature maps of size 10×10. The connection way between layer S3 and layer C4 takes much importance for the feature formation [15]. Layer S5 is the third subsampling layer with 16 feature maps of size 5×5. Layer C6 is the third convolutional layer with 120

978-0-7695-4661-2/12 $26.00 © 2012 IEEE
DOI 10.1109/DAS.2012.61

feature maps of size 1×1. Each feature map is connected to all 16 feature maps of layer S5. Layer F7 contains 130 units and is fully connected to layer C6. Finally, layer F8 (the output layer) is composed of Euclidean RBF units and is fully connected to layer F7 [15].

B. *Configuration of* LeNet-5

The CNN model used in this paper is modified *LeNet-5*. We try various modifications based on the basic architecture of *LeNet-5*, to attain a tradeoff between time-cost and recognition performance. First, we change the number of neurons in each layer of a CNN, thus different models are made, as shown in Table III. Detailed comparisons and discussions of their performance in HECR are demonstrated in section IV.

Moreover, in the original *LeNet-5* model, connecting way between layer C2 and layer C4 (between layer S3 and layer C4 actually) is asymmetrical [15]. This can assure different feature maps in layer C4 to get different sets of features from layer C2. However, this asymmetrical connecting way will lead to two subsequent problems. First, parameters of CNN may be updated in different levels. Secondly, from layer C2 to layer C4, some feature maps will get less information than others. This means that, more feature information will be lost by some feature maps. Although the loss of features also depends on the parameters between layer C2 and layer C4, this asymmetrical connecting map caused the loss. Therefore, a new symmetrical connecting way is developed, as shown in Table II. In the new symmetrical map, each feature map in layer C4 connects to more feature maps in layer C2, considering the redundance of features and the time cost.

It is crucial to normalize the input images. In our experiments, we extend the pixels of an input image with three sizes: 20×20, 28×28 and 32×32, when the left parts of the input are padded with the background value. In features' feed-forward propagation, the CNN with 32×32-normalization and 28×28-normalization lose information more easily and quickly than the one with 20×20-normalization [15]. After the same training iterations, errors for 20×20-normalized dataset is the lowest, as shown in Figure 2.

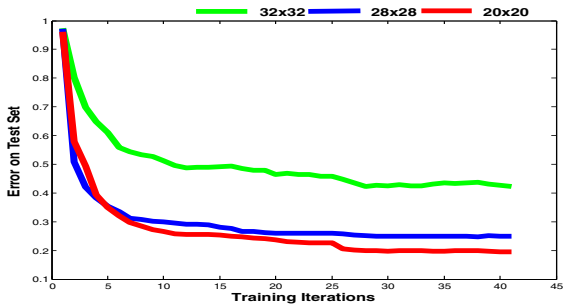

Figure 2. Comparison of Errors among Different Input Normalizations

III. Error-Samples-Reinforcement-Learning

During CNN's training, we usually face two problems. First, comparing with traditional back-propagation network (BPN), training of the CNN is time-consuming, for a CNN has tens of thousands of parameters to update. Meanwhile, feature extraction by convolution produces a large quantity of mathematical calculations. The second problem is, after certain epoches, error descends very slowly, or does not descend any more, while it is still unacceptable. Solution for the first problem depends mostly on the design of the CNN. But for the second one, dynamic adjustments of training parameters may help much. After certain epoches, the number of the misclassified samples (*error-samples*) will reduce very slowly. One reason may be that, the ratio of the *error-samples* in the train set may be too low to cause observable impacts on the CNN. The CNN may learn faster, if we reconstruct the train set dynamically, in which the ratio of the *error-samples* is increased. Therefore, an Error-Samples-Reinforcement-Learning algorithm (ESRL) is developed, as shown in Table I.

Table I
Error-Samples-Reinforcement-Learning Algorithm

Step 1	Get the CNN's initial error rate ε on the validation set;
Step 2	For each batch i of epoches, do step 3 to step 8 :
Step 3	(1) Get current error rate δ_i on the validation set; (2) if $\frac{\delta_i + \delta_{i-1} + \delta_{i-2}}{3} > \varepsilon * \theta$ break; (3) $\varepsilon = \delta_{i-2}$;
Step 4	For each epoch j in batch i : Train the CNN on current train set;
Step 5	(1) Update the learning rate η; (2) if $\eta < min(\eta)$ $\eta = \eta * \xi$ (or $\eta = \eta + \xi$);
Step 6	(1) Test current CNN on the original train set; (2) Select out *error-samples* and *well-recognized-samples*;
Step 7	Regenerate new samples based on error-samples;
Step 8	Combine new samples and *well-recognized-samples*, and adjust their ratios in the new train set.

A important rule to stop the training can be seen in step 3. Except when overfit occurs or η is dynamically increased (in Figure 3, η is increased at the epoch of 8, 18, 28, 38 and 48.), if error rate ascends abnormally, training process should be terminated. θ in is set with 1.1.

Dynamic update of the learning rate η is crucial, as in step 5. If η is too great, the error rate may ascend rather than descend, while a too small η may lead to the local optimum for the CNN. In our experiments, η ranges from 10^{-3} to 10^{-5} and is multiplied by 0.95 every two epoches during training. If η is smaller than the lower limit allowed, η will be increased to a greater level. This increase takes place in two ways: multiply or add to it with ξ. Impacts of dynamic regulation of η can be seen in Figure 3. When η increases to a higher level, error rate ascends at the same time, cause

Table II
NEW SYMMETRICAL CONNECTING WAY BETWEEN LAYER C2 AND LAYER C4

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
feature maps in layer C2 (S3)	1	0	1	1	1	0	1	1	1	1	1	1	0	1	1	1	0
	2	1	0	1	0	1	0	1	1	1	1	0	1	0	1	0	1
	3	1	1	0	1	1	1	0	1	1	0	1	1	1	0	1	1
	4	1	1	0	1	1	1	0	1	1	0	1	1	1	0	1	1
	5	1	0	1	0	1	0	1	1	1	1	0	1	0	1	0	1
	6	0	1	1	1	0	1	1	1	1	1	1	0	1	1	1	0

severe update of parameters for the CNN is unavoidable. However, this also leads the CNN to step out from the local optimum point and learn better. In our experiments, η is multiplied by ξ to increase, and ξ is set with 10.

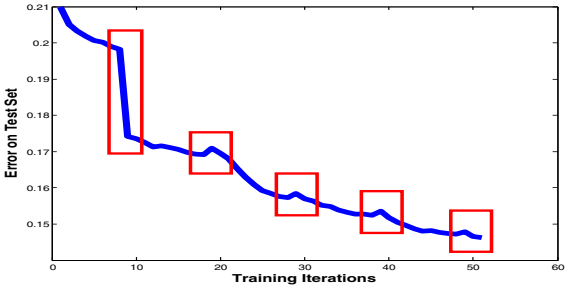

Figure 3. Error rates with η dynamically regulated

In step 7, new samples are regenerated from the *error-samples*. They are regenerated mainly through geometric transformation and noise addition. First, an original image is rotated and distorted. Then a gaussian noise is added and smoothness for it is followed. A key point is the ratio of the new regenerated samples in the new train set, which is kept between $33\% \sim 40\%$ by dynamic adjustment. When actual ratio of *error-samples* is lower than 33%, more samples are generated and added into the new train set, or some of *well-recognized-samples* are randomly removed. The size of new train set is the same with that of the original train set. Performance of the CNN on the validation set can be seen in Figure 4. New train set is reconstructed at the epoch of 6 and 16.

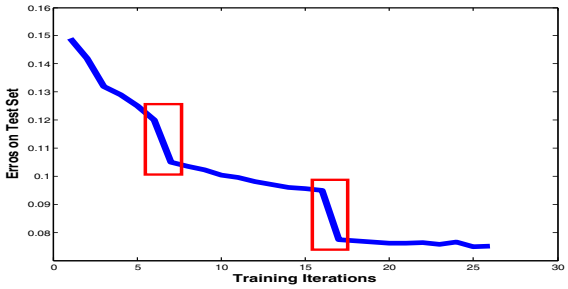

Figure 4. Error rates with dataset dynamically reconstructed

IV. EXPERIMENTS AND DISCUSSIONS

Experiments are based on 1b (uppercase) and 1c (lowercase) section of UNIPEN Train-R01/V07 database. These two sections contain 28069 samples for uppercase and 61351 samples for lowercase, respectively. In order to obtain offline character images, we employ some preprocessing steps, such as connecting the adjacent points with DDA method [16], extending to the width of strokes to 3 pixels, and anti-aliasing, etc. Although the UNIPEN dataset consists of very difficult data due to the variety of writers and noisy or mislabeled data, we used it without any cleaning. A subset with 30000 samples that are randomly selected from UNIPEN lowercase are made, for experiments of different modified CNN models and the error-correcting codes.

A. Performance of Modified CNN Models

Models with different number of neurons in some layers from *LeNet-5* and the comparisons among them are shown in Table III. For the connecting way between layer C2 and layer C4, M1 $\sim$ M7 use the connecting map which is used in the original *LeNet-5*, when M8 utilizes the new symmetrical connecting map in Table II. There are 17 neurons in layer F8 for all models, as the length of error-correcting code as the output of the CNN is 17.

Table III
SETTING OF THE NUMBER OF NEURONS AND CORRESPONDING PERFORMANCES(%)

Model	M1	M2	M3	M4	M5	M6	M7	M8
C2	6	6	6	6	6	2	10	6
C4	16	16	16	16	16	16	16	16
C6	120	120	120	160	80	80	80	80
F7	200	150	100	100	100	100	100	100
F8	17	17	17	17	17	17	17	17
TOP1	89.7	89.6	89.6	86.4	87.6	85.5	86.8	87.9
TOP2	95.0	94.9	94.9	93.0	92.8	91.2	92.5	93.9
TOP3	96.5	96.5	96.4	95.5	95.0	93.8	94.7	95.7

Through these comparisons above, we try to find the impact of the numbers of neurons in layer C2, C6 and F7, and the performance of the new symmetrical connecting way between layer C2 and C4. All models have 16 feature maps in layer C4, the same with *LeNet-5*. From M1 to M3, when the number of neurons in F7 layer descends from 200 to 100, recognition rates vary little. A possible reason may be that, unlike in prior layers with convolution and subsampling, feature in layer F7 experiences less variance. From M3

to M4, although there are more neurons in layer C6, the performance gets worse. A possible explanation is, a model with more complicated architecture, needs more training iterations to adjust the randomly-initialized parameters. In fact, M4 performances better when M3 and M4 are both after a longer training process.

M5 and M8 have the same parameters for each layer, except the connecting way between layer C2 and C4. The increase of recognition rate from M5 to M8, shows the good robustness of the new symmetrical connecting way. Our experiments for latter HECR are based on M8.

B. Error-Correcting Code and Rejection

In the original *LeNet-5* for traditional *Competitive-Learning* (CL), outputs are set with place code. *LeNet-5* with place code is incapable of rejection for the recognition results. Deng set the outputs of *LeNet-5* with error-correcting codes (EC codes), thus *LeNet-5* had the ability to reject illegal samples in his research for printed character recognition [13]. However, rejection should be more meaningful for handwritten character recognition, in which many samples are in unconstrained style, even illegal. These samples should be rejected in latter recognition. Some of the illegal lowercase samples can be seen in Figure 5.

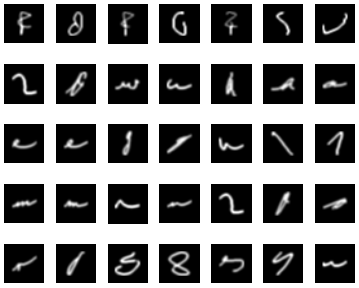

Figure 5. Some illegal lowercase samples

For rejection, a rejecting distance (rej-dist) is predefined. The Hamming distances between the EC vector of the CNN's outputs and all EC codes for 26 characters are calculated respectively. If the minimum of these 26 distances are greater than the rej-dist, recognition results will be rejected. The recognition results after rejection are then sorted by Hamming distances in ascending order. TOPX results are then attained, which are needed in actual applications of OCR.

Since EC code is consisting of 1 and -1, outputs of the CNN are converted to a EC code by comparing with a threshold, as Equation 1.

$$output = \begin{cases} 1, & if\ output > threshold \\ -1, & if\ output \leq threshold \end{cases} \quad (1)$$

It seems the median of 1 and -1 for this threshold has the priority. However, experiments tell us that, the very threshold with which the error in TOP1 result descends to the lowest, is 0.65, not 0, as shown in Figure 6. When the threshold is between -0.55 and 0.65, the closer it gets to 0, the greater the error becomes. Although the error with threshold of 0.65 is the lowest, the confidence level of recognition results is lower, too, for the average minimum of Hamming distances and the actual average number of X candidates are both higher. In our experiments, 0 is used as the threshold, with which the average minimum of Hamming distances and the actual average number of X candidates for all samples are both the lowest.

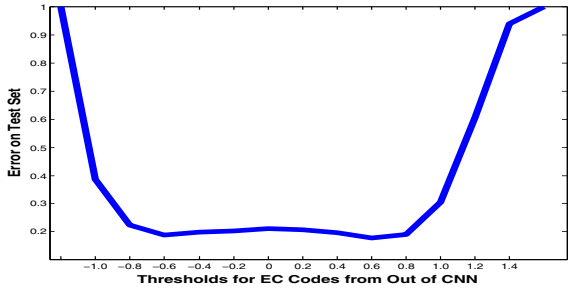

Figure 6. Errors in TOP1 with different thresholds for EC codes

Definition of the rej-dist needs more explorations. The minimum of Hamming distances among EC codes of 26 characters is 6. Recognition rates with different rej-dists are shown in Table IV. We can notice that, TOP1 recognition rate increases rapidly as the rej-dist increases from 0 to 3, and slowly as the rej-dist goes beyond than 4. So, it seems 4 is a good choice for the rej-dist.

Table IV
RECOGNITION RATES (%) IN TOPX WITH DIFFERENT REJ-DISTS

rej-dist	0	1	2	3	4	5	6
TOP1	46.95	59.15	67.41	74.03	78.11	79.16	79.21
TOP2	*	59.15	67.41	74.03	79.71	84.68	87.67
TOP3	*	*	67.41	74.03	79.71	84.86	89.17

With Hamming distances, X candidate result may be a set of characters rather than a single character, for the outputs of network may have the same Hamming distance to several EC codes of 26 characters. In the end, the actual character number for TOPX results maybe more than X. A new problem emerges: too many candidate results will lead TOPX results meaningless. The actual average numbers of candidate results are 1.29 (the first candidate), 5.54 (the second candidate) and 0.66 (the third candidate), respectively. Experiments also demonstrates that, almost all of the samples have no more than 3 candidates. Ratios for samples which have one, two and three candidates are 2.0%, 86.3% and 11.7%, respectively. Therefore, TOP1 $\sim$ TOP3 results are attained in Table IV.

C. Results

Train and test sets for final HECR experiments are created with jackknife method. All of samples of uppercase or lowercase are randomly divided into N subsets. Training is on the first $N-1$ subsets; 33% and 67% of the N^{th} subset are validation set and test set respectively. Training is repeated N times. In our experiments, N is 3. Final recognition results are shown in Table V. For better comparison, results of other state-of-the-art researches for offline HECR on UNIPEN dataset are also listed.

Table V
RECOGNITION ERROR RATES (%)

	Methods	TOPX(%)	Multi-writer	Omni-writer	Ref.
Upper case	CNN	TOP 1		6.3	
		TOP 2		2.3	
		TOP 3		1.8	
	KNN		8.3		[4]
	KNN/SVC		5.2		[4]
Lower case	CNN	TOP 1		9.8	
		TOP 2		4.5	
		TOP 3		2.9	
	KP-NN			18.8	[1]
	MLP			14.4	[2]

V. CONCLUSION

This paper shows the solution for HECR with CNNs. The output of the CNN are set with EC codes, thus the CNN has the ability for rejection in recognition. Comparing with other state-of-the-art methods, the CNN has provided an encouraging solution for offline HECR. Nevertheless, further explorations for CNN are still needed. We will continue to find more effective CNN models, in combination with other methods of online HECR for handwritten recognition.

ACKNOWLEDGMENT

Experiments in this paper are conducted on the UNIPEN dataset. Here, we acknowledge the organizer, and all participators and volunteers of UNIPEN project.

REFERENCES

[1] J.F. Hébert, M. Parizeau, and N. Ghazzali. Learning to segment cursive words using isolated characters. In *Proc. of the Vision Interface Conference*, pages 33–40, 1999.

[2] M. Parizeau, A. Lemieux, and C. Gagné. Character recognition experiments using unipen data. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pages 481–485. IEEE, 2001.

[3] E.H. Ratzlaff. Methods, reports and survey for the comparison of diverse isolated character recognition results on the unipen database. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pages 623–628. IEEE, 2003.

[4] L. Vuurpijl and L. Schomaker. Two-stage character classification: A combined approach of clustering and support vector classifiers. In *Proc 7th International Workshop on Frontiers in Handwriting Recognition, Amsterdam, Netherlands*, pages 423–432, 2000.

[5] R. Ghosh and M. Ghosh. An intelligent offline handwriting recognition system using evolutionary neural learning algorithm and rule based over segmented data points. *Journal of Research and Practice in Information Technology*, 37(1):73–88, 2005.

[6] V. Nguyen and M. Blumenstein. Techniques for static handwriting trajectory recovery: a survey. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, pages 463–470. ACM, 2010.

[7] Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361, 1995.

[8] D. Bouchain. Character recognition using convolutional neural networks. *Institute for Neural Information Processing*, 2007, 2006.

[9] F. Lauer, C.Y. Suen, and G. Bloch. A trainable feature extractor for handwritten digit recognition. *Pattern Recognition*, 40(6):1816–1824, 2007.

[10] Y. LeCun, K. Kavukcuoglu, and C. Farabet. Convolutional networks and applications in vision. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 253–256. IEEE, 2010.

[11] P.Y. Simard, D. Steinkraus, and J.C. Platt. Best practices for convolutional neural networks applied to visual document analysis. *Document Analysis and Recognition*, 2:958, 2003.

[12] Y. Bengio, Y. LeCun, C. Nohl, and C. Burges. Lerec: A nn/hmm hybrid for on-line handwriting recognition. *Neural Computation*, 7(6):1289–1303, 1995.

[13] H. Deng, G. Stathopoulos, and C.Y. Suen. Error-correcting output coding for the convolutional neural network for optical character recognition. In *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pages 581–585. IEEE, 2009.

[14] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet. Unipen project of on-line data exchange and recognizer benchmarks. In *Pattern Recognition, 1994. Vol. 2-Conference B: Computer Vision & Image Processing., Proceedings of the 12th IAPR International. Conference on*, volume 2, pages 29–33. IEEE, 1994.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[16] A. Fujimoto, T. Tanaka, and K. Iwata. Arts: Accelerated ray-tracing system. *Computer Graphics and Applications, IEEE*, 6(4):16–26, 1986.