A Study of Designing Compact Classifiers using Deep Neural Networks for Online Handwritten Chinese Character Recognition

Jun Du[1], Jin-Shui Hu[2], Bo Zhu[2], Si Wei[2], Li-Rong Dai[1]
[1]*University of Science and Technology of China, Hefei, Anhui, P. R. China*
[2]*iFlytek Research, Hefei, Anhui, P. R. China*
jundu@ustc.edu.cn, {jshu,bozhu,siwei}@iflytek.com ,lrdai@ustc.edu.cn

Abstract

This paper presents a study of designing compact classifiers using deep neural networks for recognition of online handwritten Chinese characters. Two schemes are investigated based on practical considerations. First, deep neural networks are adopted purely as a classifier with a state-of-the-art feature extractor of online handwritten Chinese characters. Second, the so-called bottleneck features extracted from a bottleneck layer of deep neural networks are fed to the prototype-based classifier. The experiments on an in-house developed online Chinese handwriting corpus with a vocabulary of 15,167 characters show that compared with prototype-based classifier widely developed on the mobile device, deep neural network based classifier can yield significant improvements of recognition accuracy with acceptably increased footprint and latency while the bottleneck-feature approach can bring a more compact classifier with an observable performance gain.

1. Introduction

With the fast development of mobile internet, online handwritten Chinese character recognition as an input mode on a mobile device (e.g., Smartphone, Tablet) becomes increasingly popular. To deliver a compelling user experience, a recognizer has to be designed to have a small footprint, run efficiently on a mobile device, achieve high recognition accuracy. Several solutions have been developed to achieve the above goals [3, 17, 20]. Recently, the results of the Chinese handwriting recognition competition [21] reveal that the best recognition performances of many tasks are achieved by the research group using the deep learning technologies. Based on the competition report this year [21], the system from Fujitsu R&D Center uses multiple convolutional neural networks and yields the best results on the offline character recognition task while the system from University of Warwick also uses a large convolutional neural network and achieves the best performance on the online character recognition task. Furthermore, after the competition, the researchers from the Dalle Molle Institute for Artificial Intelligence (IDSIA) claim that their system using multi-column deep neural networks [2] outperforms the best system for the offline character recognition task after correcting a bug. The long-short-term-memory recurrent neural network is another showcase of deep learning technology which becomes popular in handwriting recognition recently [14, 6, 18]. In spite of those promising results, one common issue of deep learning in real applications is the corresponding footprint and latency of the recognizer. So the motivation of this work is to investigate the possibility of designing a compact product engine on the mobile device for online handwritten Chinese character recognition via the deep learning technology.

One of the state-of-the-art techniques to build a Chinese handwriting recognizer is to use a so-called sample separation margin (SSM) based minimum classification error (MCE) criterion [7] to train a prototype-based classifier as reported in [3]. In spite of the large vocabulary of Chinese characters, such a classifier can be made both compact [20] and efficient [4] in the recognition stage. The main contribution of this paper is to propose two new classifiers via deep neural networks (DNNs) [11, 10]. One classifier is purely based on DNN which is directly modeling the posterior probability of each character class. The other classifier is designed using the so-called bottleneck features (BNF) [5, 22, 19] which is widely used in speech recognition area. Those two classifiers demonstrate the superiority to the prototype-based classifier in both recognition accuracy and compactness of the recognizer.

The remainder of the paper is organized as follows. In Section 2, we give a system overview of our pro-

1051-4651/14 $31.00 © 2014 IEEE
DOI 10.1109/ICPR.2014.508

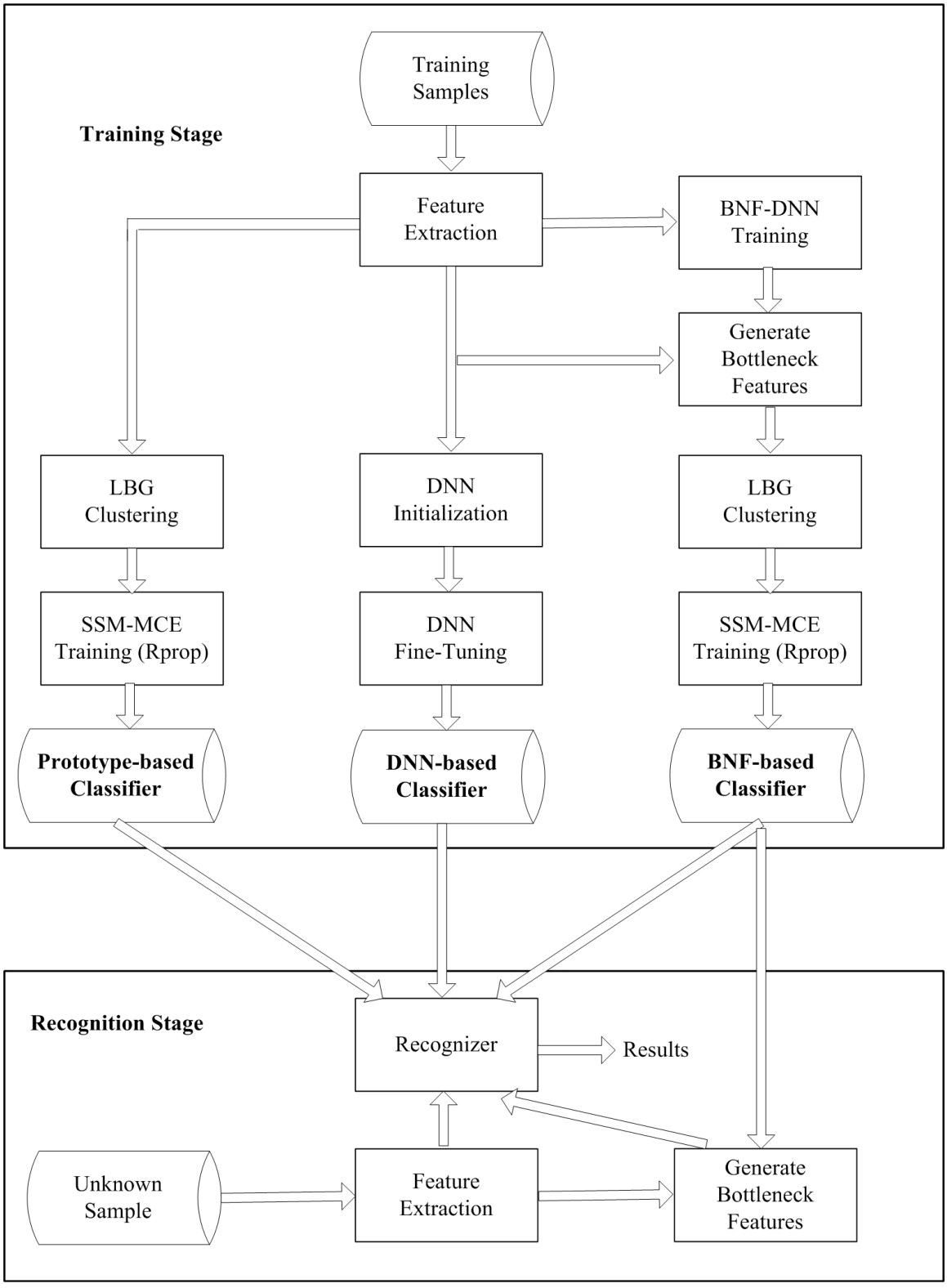

Figure 1. Overall development flow and architecture.

posed approach. In Section 3, we present the detailed description of three classifiers. In Section 4, we report experimental results. Finally we conclude the paper in Section 5.

2. System Overview

An overall system development flow and architecture is illustrated in Fig. 1. Three classifiers, namely prototype-based classifier, deep neural network based classifier, and bottleneck-feature based classifier, are designed and compared. In the training stage, first a raw feature vector is extracted from each training sample [1], which is followed by LDA (linear discriminant analysis) transformation to obtain a lower dimensional feature vector. After that, the prototype-based classifier is constructed by using LBG clustering algorithm [16], which is then refined by SSM-MCE training with an Rprop algorithm [13] while deep neural network based classifier is initialized using RBM (restricted Boltzmann machine) pre-training and then fine-tuned based on the cross-entropy criterion. As for the BNF-based classifier, a DNN with a bottleneck layer (BNF-DNN) is trained first. Then new bottleneck features are generated via BNF-DNN, which are fed to the prototype-based classifier training. At the recognition stage, with the feature vector extracted from the unknown sample, the normal recognition is performed based on prototype-based classifier or DNN-based classifier while an additional bottleneck feature generation step is needed for BNF-based classifier.

3. Classifiers Description

3.1. Prototype-based classifier

Suppose our classifier can recognize M character classes denoted as $\{C_i | i = 1, ..., M\}$. For a multi-prototype based classifier, each class C_i is represented by K_i prototypes, $\lambda_i = \{\mathbf{m}_{ik} \in \mathcal{R}^D | k = 1, ..., K_i\}$, where $\mathbf{m}_{ik}$ is the k^{th} prototype of the i^{th} class. Let's use $\Lambda = \{\lambda_i\}$ to denote the set of prototypes. In the classification stage, a feature vector $\mathbf{y} \in \mathcal{R}^D$ is first extracted. Then $\mathbf{y}$ is compared with each of the M classes by evaluating a Euclidean distance based discriminant function for each class C_i as follows

$$g_i(\mathbf{y}; \lambda_i) = -\min_k \parallel \mathbf{y} - \mathbf{m}_{ik} \parallel^2 \ . \tag{1}$$

The class with the maximum discriminant function score is chosen as the recognized class $r(\mathbf{y}; \mathbf{\Lambda})$, i.e.,

$$r(\mathbf{y}; \mathbf{\Lambda}) = \arg\max_i g_i(\mathbf{y}; \lambda_i) \ . \tag{2}$$

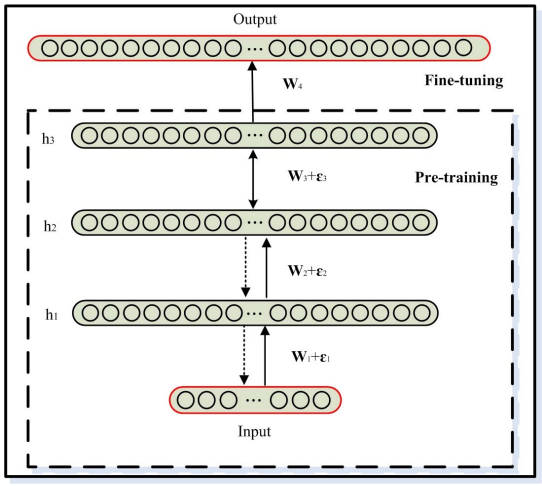

Figure 2. Deep neural network.

In the training stage, given a set of training feature vectors $\mathcal{Y} = \{\mathbf{y}_r \in \mathcal{R}^D | r = 1, ..., R\}$, first we initialize $\mathbf{\Lambda}$ by LBG clustering [16]. Then $\mathbf{\Lambda}$ can be re-estimated by minimizing the following SSM-MCE objective function:

$$l(\mathcal{Y}; \mathbf{\Lambda}) = \frac{1}{R} \sum_{r=1}^{R} \frac{1}{1 + \exp[-\alpha d(\mathbf{y}_r; \mathbf{\Lambda}) + \beta]} \tag{3}$$

where α, β are two control parameters, and $d(\mathbf{y}_r; \mathbf{\Lambda})$ is a misclassification measure defined by using a so-called sample separation margin (SSM) as follows [7]:

$$d(\mathbf{y}_r; \mathbf{\Lambda}) = \frac{-g_p(\mathbf{y}_r; \lambda_p) + g_q(\mathbf{y}_r; \lambda_q)}{2 \parallel \mathbf{m}_{p\hat{k}} - \mathbf{m}_{q\bar{k}} \parallel} \tag{4}$$

where

$$\hat{k} = \arg\min_k \parallel \mathbf{y}_r - \mathbf{m}_{pk} \parallel^2 \tag{5}$$

$$q = \arg\max_{i \in \mathcal{M}_r} g_i(\mathbf{y}_r; \lambda_i) \tag{6}$$

$$\bar{k} = \arg\min_k \parallel \mathbf{y}_r - \mathbf{m}_{qk} \parallel^2 \tag{7}$$

and $\mathcal{M}_r$ is the hypothesis space for the r^{th} sample, excluding the true label p.

To optimize the objective function in Eq. (3), the same implementation of Rprop algorithm as described in [3] is adopted here.

3.2. Deep neural network based classifier

A deep neural network is a feed-forward, artificial neural network that has more than one layer of hidden units between its inputs and outputs [9]. In this work, DNN is adopted for classification to model the posterior

probability of the character class. The DNN training is illustrated in Fig. 2, which consists of generative pre-training and supervised fine-tuning.

The pre-training procedure treats each consecutive pair of layers as a RBM [10] whose joint probability is defined as:

$$p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} \exp\{-E(\boldsymbol{v}, \boldsymbol{h})\} \tag{8}$$

where $\boldsymbol{v}$ and $\boldsymbol{h}$ denote the observable variables and latent (hidden) variables, respectively. E is an energy function and Z is the partition function to ensure $p(\boldsymbol{v}, \boldsymbol{h})$ is a valid probability distribution. If both $\boldsymbol{v}$ and $\boldsymbol{h}$ are binary states, i.e., the Bernoulli-Bernoulli RBM, the energy function is given by

$$E(\boldsymbol{v}, \boldsymbol{h}) = -(\boldsymbol{b}_v^\top \boldsymbol{v} + \boldsymbol{b}_h^\top \boldsymbol{h} + \boldsymbol{v}^\top \boldsymbol{W}_{vh} \boldsymbol{h}) \tag{9}$$

where $\boldsymbol{b}_v$, $\boldsymbol{b}_h$ are bias vectors of $\boldsymbol{v}$ and $\boldsymbol{h}$ respectively, and $\boldsymbol{W}_{vh}$ is the weight matrix between them. If $\boldsymbol{v}$ is real-valued data and $\boldsymbol{h}$ is binary, i.e., the Gaussian-Bernoulli RBM, the energy function is:

$$E(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{2}(\boldsymbol{v} - \boldsymbol{b}_v)^\top (\boldsymbol{v} - \boldsymbol{b}_v) - \boldsymbol{b}_h^\top \boldsymbol{h} - \boldsymbol{v}^\top \boldsymbol{W}_{vh} \boldsymbol{h} \tag{10}$$

where we assume that the visible units follow the Gaussian noise model with an identity covariance matrix if the input data are pre-processed by mean and variance normalization.

The RBM parameters can be efficiently trained in an unsupervised fashion by maximizing the likelihood over training samples of visible units with the approximate contrastive divergence algorithm [10]. As for our DNN, a Gaussian-Bernoulli RBM is used for the first layer while a pile of Bernoulli-Bernoulli RBMs can be stacked behind the Gaussian-Bernoulli RBM. Then the parameters of RBMs can be trained layer-by-layer. Hinton *et al.* indicate that this greedy layer-wise unsupervised learning procedure always helps over the traditional random initialization.

After pre-training for initializing the weights of the first several layers, a supervised fine-tuning of the parameters in the whole neural network with the final output layer is performed. For multiclass classification, output unit j converts its total input x_j into a class probability p_j by using the "softmax" non-linearity:

$$p_j = \frac{\exp(x_j)}{\sum_k \exp(x_k)} \tag{11}$$

where k is an index over all classes. Then the objective function is the cross-entropy between the target prob abilities $\bar{p}$ and the outputs of the softmax p:

$$C = -\sum_j \bar{p}_j \log p_j \tag{12}$$

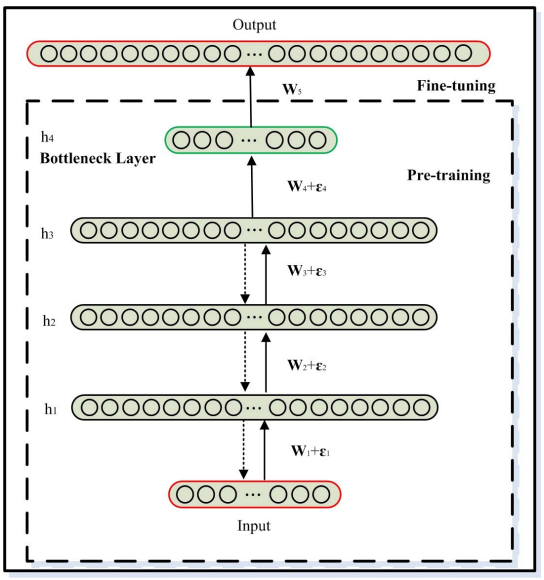

Figure 3. DNN for bottleneck features.

where the target probabilities, taking values of one or zero, are the supervised information provided to train the DNN classifier. As our task involves large training samples, the objective function is optimized using back-propagation procedure with stochastic gradient descent in mini-batch mode. To address the over-fitting problem of DNN training, the dropout strategy in [12, 15] is adopted. The main principle of dropout is randomly omitting hidden units with certain probabilities on each training case, which can be explained in two ways, one is to prevent co-adaptations of the units, and the other is to average the predictions of many different networks.

3.3. Bottleneck-feature based classifier

As shown in Fig. 3, bottleneck features [5, 22, 19] can be generated from a DNN where one of the internal layers (bottleneck layer) has a small number of hidden units, relative to the size of the other layers. The bottleneck layer creates a constriction in the network that forces the information pertinent to classification into a low dimensional representation. In this work, bottleneck features are created from a deep neural network trained to predict character classes. The inputs to the hidden units of the bottleneck layer are used as features for prototype-based classifier. These bottleneck features represent a nonlinear and discriminative transformation of the input features. The training procedure of DNN for bottleneck feature is almost the same as that described in Section 3.2 except the dropout strategy as our experiments show that dropout can not improve the performance due to the small size of the bottleneck layer.

4. Experiments and Results

The experiments are conducted on the task of recognizing isolated online handwritten characters with a vocabulary of 15,167 character classes including 62 alphanumeric characters, 101 punctuation marks and 15,004 frequently used Chinese characters. For training, we totally use 14,846,606 character samples, about 1,000 samples per character class. There are 644,500 samples from 3,755 character classes in the testing set which are written in regular style or cursive style. For feature extraction, a 392-dimensional raw feature vector is extracted as described in [1], which is followed by LDA transformation to obtain a 96-dimensional feature vector. For Rprop-based SSM-MCE training, the setting of the control parameters can refer to [3]. The tuning parameters of DNN are set according to [8]. To handle the large-scale training data, the computations of LBG clustering, SSM-MCE training with Rprop algorithm are parallelized on the CPU cluster while DNN training is implemented and optimized on GPUs.

Table 1 shows a performance comparison of systems using prototype-based classifier with original features under different settings of number of prototypes on the testing set. "LBG" denotes a system trained using LBG clustering while "SSM-MCE" refers to a system trained by the SSM-MCE criterion. SSM-MCE systems consistently and significantly outperform LBG systems on the testing set with different number of prototypes. Table 2 lists a performance comparison of systems using BNF-based classifier under different settings of number of prototypes on the testing set. The DNN architecture for bottleneck feature is 96-1024-1024-1024-K-15167, which denotes that the sizes are 96 for input layer, 1024 for three hidden layers, K for bottleneck layer, and 15167 for output layer. Three configurations of K (64, 96, 256) are compared, where BNF-96 uses the same dimension as that of original feature vector. For prototype-based classifier using bottleneck features, SSM-MCE training can still yield significant improvements over LBG clustering for all testing cases. With the increased dimension of bottleneck feature vector, the recognition accuracies of the testing set also increase monotonically under the same prototype setting. By considering the tradeoff between recognition accuracy and the footprint (or size) of classifier, BNF-96 is the best system which means BNF-64 is more compact than BNF-96 but with observable performance degradation while BNF-256 outperforms BNF-64 but with a much bigger footprint. Compared with the results in Table 1, BNF-based systems (e.g. BNF-96) always outperform the corresponding systems with original features, especially for LBG clustering cases.

Table 1. Performance (recognition accuracies in %) comparison of systems using prototype-based classifier with original features on the testing set.

#prototype	1	2	3	4
LBG	75.20	77.10	78.10	79.55
SSM-MCE	82.74	84.64	85.07	85.62

Table 2. Performance (recognition accuracies in %) comparison of systems using BNF-based classifier on the testing set.

	#prototype	LBG	SSM-MCE
BNF-64	1	80.13	84.76
	2	81.23	85.32
	4	82.10	85.74
BNF-96	1	80.94	85.33
	2	81.97	85.83
	4	82.84	86.28
BNF-256	1	81.11	85.37
	2	82.26	85.99
	4	83.22	86.65

Table 3 summarizes a performance comparison of systems using three classifiers on the testing set. "Baseline" denotes the SSM-MCE system with 4 prototypes in Table 1. For the system footprint and latency, the value is normalized to that of Baseline system. "BNF-96-P1", "BNF-96-P2", "BNF-96-P4" represent the SSM-MCE trained BNF-96 systems in Table 2 with 1 prototype, 2 prototypes, and 4 prototypes, respectively. "DNN-96" and "DNN-1024" are the systems using DNN-based classifier where the architectures are 96-1024-1024-1024-96-15167 and 96-1024-1024-1024-15167, respectively. Based on those results, we can observe that on top of the state-of-the-art Baseline system, DNN-1024 system can achieve significant improvements of recognition accuracy with an acceptably increased footprint and latency while BNF-96-P4 system can be considered as a tradeoff between Baseline and DNN-1024 systems. Furthermore, BNF-96-P2 system, with a smaller footprint and comparable latency, still yields higher accuracy than Baseline system, which indicates the compactness of BNF-based classifier. Finally, with the same footprint and latency, BNF-96-P1 system is superior to DNN-96 system in recognition accuracy for the case of extremely compact and efficient recognizer. Another advantage of BNF-based classifier is that all the other techniques (e.g., writer adaptation) can be directly applied.

Table 3. Overall performance comparison of systems using three classifiers on the testing set.

	Accuracy	Footprint	Latency
Baseline	85.62	1	1
BNF-96-P4	86.28	1.39	1.48
BNF-96-P2	**85.83**	**0.89**	**0.98**
BNF-96-P1	85.33	0.64	0.73
DNN-96	85.20	0.64	0.73
DNN-1024	**88.53**	**3.04**	**3.10**

5. Conclusion

In this work, we design compact classifiers using deep neural networks for online handwritten Chinese character recognition. Based on our experiments, compared with the state-of-the-art prototype-based classifier, bottleneck-feature based classifier can be made more compact and efficient while deep neural network based classifier can significantly improve the recognition accuracy with acceptably increased footprint and latency for the development on the mobile device.

6. Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grants No. 61305002, the Programs for Science and Technology Development of Anhui Province, China under Grants No. 13Z02008-4 and No. 13Z02008-5, and Youth Innovation Fund of USTC (University of Science and Technology of China) under Grants No. 2013.

References

[1] Z.-L. Bai and Q. Huo. A study on the use of 8-directional features for online handwritten Chinese character recognition. In *ICDAR*, pages 262–266, 2005.

[2] D. C. Ciresan and J. Schmidhuber. Multi-column deep neural networks for offline handwritten Chinese character classification. In *Preprint arXiv:1309.0261*, 2013.

[3] J. Du and Q. Huo. Designing compact classifiers for rotation-free recognition of large vocabulary online handwritten Chinese characters. In *ICASSP*, pages 1721–1724, 2012.

[4] Z.-D. Feng and Q. Huo. Confidence guided progressive search and fast match techniques for high performance Chinese/English OCR. In *ICPR*, pages III–89–92, 2002.

[5] F. Grézl, M. Karafiát, S. Kontár, and J. Černocký. Probabilistic and bottle-neck features for LVCSR of meetings. In *ICASSP*, pages 757–761, 2007.

[6] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 31(5):855–868, 2009.

[7] T. He and Q. Huo. A study of a new misclassification measure for minimum classification error training of prototype-based pattern classifiers. In *ICPR*, 2008.

[8] G. Hinton. A practical guide to training restricted Boltzmann machines. Technical report, University of Toronto, 2010.

[9] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[10] G. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

[11] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[12] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. In *Preprint arXiv:1207.0580*, 2012.

[13] C. Igel and M. Hüsken. Improving the Rprop learning algorithm. In *2nd Int. Symp. on Neural Computation*, pages 115–121, 2000.

[14] M. Kozielski, P. Doetsch, and H. Ney. Improvements in RWTH's system for off-line handwriting recognition. In *ICDAR*, 2013.

[15] J. Li, X. Wang, and B. Xu. Understanding the dropout strategy and analyzing its effectiveness on LVCSR. In *ICASSP*, pages 7614–7618, 2013.

[16] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Trans. on Communications*, 28(1):84–95, 1980.

[17] C.-L. Liu, S. Jaeger, and M. Nakagawa. Online recognition of Chinese characters: the state-of-the-art. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(2):198–213, 2004.

[18] M. Liwicki, A. Graves, H. Bunke, and J. Schmidhuber. A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks. In *ICDAR*, 2007.

[19] T. Sainath, B. Kingsbury, and B. Ramabhadran. Auto-encoder bottleneck features using deep belief networks. In *ICASSP*, pages 4153–4156, 2012.

[20] Y.-Q. Wang and Q. Huo. A study of designing compact recognizers of handwritten Chinese characters using multiple-prototype based classifiers. In *ICPR*, pages 1872–1875, 2010.

[21] F. Yin, Q.-F. Wang, X.-Y. Zhang, and C.-L. Liu. ICDAR 2013 Chinese handwriting recognition competition. In *ICDAR*, pages 1464–1470, 2013.

[22] D. Yu and M. L. Seltzer. Improved bottleneck features using pretrained deep neural networks. In *INTERSPEECH*, pages 237–240, 2011.