Writer Adaptation using Bottleneck Features and Discriminative Linear Regression 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 novel approach to writer adaptation using bottleneck features and discriminative linear regression for the recognition of online handwritten Chinese characters. First, bottleneck features extracted from a bottleneck layer of a deep neural network representing a nonlinear and discriminative transformation of the input features are verified to be much more effective in adaptation of writing styles than the conventional features after linear discriminant analysis transformation. Second, discriminative linear regression via a so-called sample separation margin based minimum classification error criterion is adopted for writer adaptation. The experiments on an in-house developed online Chinese handwriting corpus with a vocabulary of 15,167 characters and testing data collected from user inputs of Smartphones show that our proposed approach can achieve very significant improvements of recognition accuracy compared with a state-of-the-art adaptation approach for writer adaptation.

1. Introduction

In the mobile internet era, online handwritten Chinese character recognition as an input mode on a mobile device (e.g., Smartphone, Tablet) becomes increasingly popular. Several solutions have been developed to build product engines for online handwritten Chinese character recognition (e.g., [15, 4, 21]). But due to the large variability of writing styles by different writers, especially the cursive writing style, the user experience is not always satisfactory. Writer adaptation is one of solutions to address this problem, which aims to improve the recognition performance and user experience of a single writer by using the corresponding data samples to be recognized itself via an *unsupervised adaptation* strategy, or by using a small amount of adaptation data samples with labels collected from the target writer via a *supervised adaptation* strategy.

Many research efforts have been made in writer adaptation for online handwriting recognition in the past several decades. For example, in [16], a writer-adaptable online character recognizer was designed via a time delay neural network where the last layer is served as a linear optimal hyperplane classifier which can easily adapt to new writing styles. Platt et al. [17] placed a so-called output adaptation module on top of standard neural networks by using a radial basis function (RBF) network. In [2], a hidden Markov model (HMM) based recognition system for cursive German script could be adapted to the writing style of a new writer using maximum likelihood linear regression (MLLR) or maximum a posteriori (MAP) criterion. Vuori and Korkeakoulu [19] proposed several strategies for adaptation of a prototype-based classifier, including adding new prototypes, reshaping existing prototypes, and inactivating poorly performing prototypes. In [13], the use of biased regularization for support vector machine (SVM) based classifier is adopted for personalization.

In this paper, we study the writer adaptation techniques for online handwritten Chinese character recognition. One of the state-of-the-art techniques to build a Chinese handwriting recognizer is to use a so-called sample separation margin based minimum classification error (SSM-MCE) criterion [8] to train a prototype-based classifier as reported in [4]. In spite of the large vocabulary of Chinese characters, such a classifier can be made both compact [20] and efficient [6] in the recognition stage. In this work, based on this classifier, we propose to use bottleneck features (BNFs) [7, 22, 18] and discriminative linear regression (DLR) [3, 5] for writer adaptation. As a highly nonlinear and discriminative transformation of the input features, BNF is extracted from a bottleneck layer of a deep neu-

2167-6445/14 $31.00 © 2014 IEEE
DOI 10.1109/ICFHR.2014.59

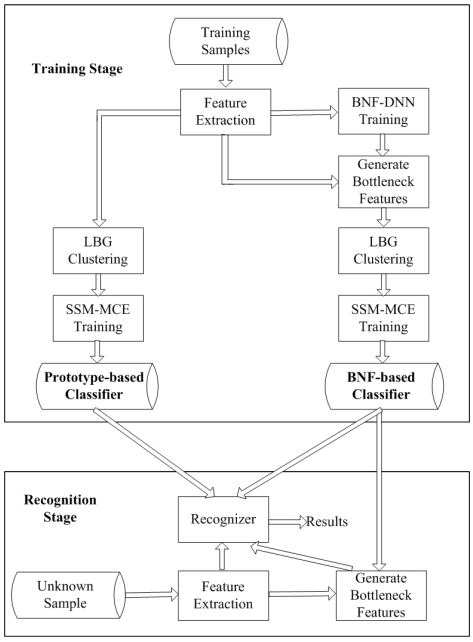

Figure 1. Overall development flow and architecture of two classifiers.

ral network (DNN) [10], which is widely used in speech recognition area. For writer adaptation in handwriting recognition, BNF also demonstrates the superiority to the conventional features after linear discriminant analysis (LDA) transformation. As for the objective function for writer adaptation, DLR via SSM-MCE criterion is adopted. Our work is relevant to the style transfer mapping (STM) [23, 24] work, where a least regularized weighted squared error approach is used to estimate a global feature transform for writer adaptation. The experiments show that our approach using BNF and DLR can achieve very significant improvements of recognition accuracy over STM approach in [23].

The remainder of the paper is organized as follows. In Section 2, we present the detailed description of prototype-based and BNF-based classifiers. In Section 3, two adaptation criteria, namely STM and DLR, are described. In Section 4, we report experimental results. Finally we conclude the paper in Section 5.

2. Classifiers Description

An overall system development flow and architecture is illustrated in Fig. 1. Two classifiers, namely prototype-based classifier and BNF-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 transformation to obtain a lower dimensional feature vector. After that, the prototype-based classifier is constructed by using LBG clustering algorithm [14], which is then refined by SSM-MCE training with an Rprop algorithm [12]. 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 while an additional bottleneck feature generation step is needed for BNF-based classifier. More details can refer to the following subsections.

2.1. 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 $\mathbf{x} \in \mathcal{R}^D$ is first extracted. Then $\mathbf{x}$ is compared with each of the M classes by evaluating a Euclidean distance based discriminant function for each class C_i as follows

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

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

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

In the training stage, given a set of training feature vectors $\mathcal{X} = \{\mathbf{x}_r \in \mathcal{R}^D|r = 1, ..., R\}$, first we initialize $\mathbf{\Lambda}$ by LBG clustering [14]. Then $\mathbf{\Lambda}$ can be re-estimated by minimizing the following SSM-MCE objective function:

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

where α, β are two control parameters, and $d(\mathbf{x}_r; \mathbf{\Lambda})$ is a misclassification measure defined by using a so-called sample separation margin (SSM) as follows [8]:

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

where

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

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

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

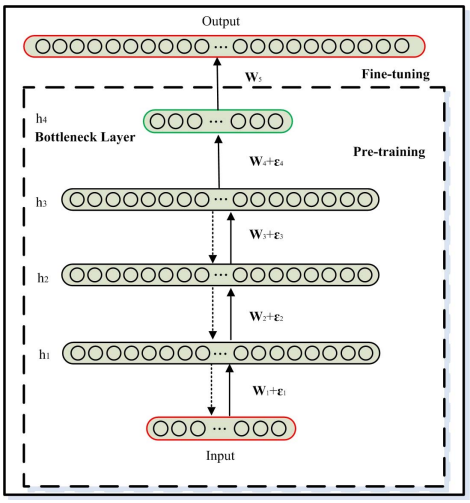

Figure 2. DNN for bottleneck features.

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 [4] is adopted here.

2.2. Bottleneck-feature based classifier

As shown in Fig. 2, bottleneck features [7, 22, 18] 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 consists of generative pre-training and supervised fine-tuning.

The pre-training procedure treats each consecutive pair of layers as a restricted Boltzmann machine (RBM) [11] whose joint probability is defined as:

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

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

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

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

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

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 [11]. 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 probabilities $\bar{p}$ and the outputs of the softmax p:

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

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.

3. Writer Adaptation via Linear Regression

In this work, we only focus on the supervised adaptation. Suppose we are given a set of labeled adaptation

data $\mathcal{Y} = \{\mathbf{y}_r \in \mathcal{R}^D | r = 1, ..., R'\}$ collected from a single writer. Then a linear feature transformation is adopted for writer adaptation:

$$\mathbf{x}_r = \mathcal{F}(\mathbf{y}_r; \mathbf{\Theta}) = \mathbf{A}\mathbf{y}_r + \mathbf{b} \quad (13)$$

where $\mathbf{\Theta} = \{\mathbf{A}, \mathbf{b}\}$ denotes the set of transform parameters. $\mathbf{A}$ is a $D \times D$ nonsingular matrix and $\mathbf{b}$ is a D-dimensional bias vector. $\mathbf{y}_r$ and $\mathbf{x}_r$ are the r^{th} D-dimensional input and transformed feature vectors, respectively. In the recognition stage, the estimated parameters $\{\mathbf{A}, \mathbf{b}\}$ are used to transform the feature vector of each unknown sample first, which is then fed to the classifier for recognition. In the following subsections, two approaches are elaborated to learn the parameters of the linear transformation.

3.1. Style transfer mapping

One approach is the style transfer mapping proposed in [23, 24]. In STM, first define the source point set as the set of feature vectors of adaptation samples (i.e., $\mathcal{Y}$), and the target point set as the set of the corresponding prototypes with the smallest Euclidean distances to those features vectors. Then find a style transfer matrix $\mathbf{A}$ to solve the following optimization problem:

$$\min_{\mathbf{A}} \sum_{r=1}^{R'} f_r \|\mathbf{A}\mathbf{s}_r - \mathbf{t}_r\|_2^2 + \beta_1 \|\mathbf{A} - \mathbf{I}\|_2^2 \quad (14)$$

where the r^{th} source point $\mathbf{s}_r$ is transformed to the target point $\mathbf{t}_r$ with the confidence f_r, which is set as 1 for supervised adaptation. The hyperparameter β_1 is set as

$$\beta_1 = \frac{\tilde{\beta}_1}{2D} \text{tr}\left(\sum_r f_r(\mathbf{s}_r + \mathbf{t}_r)\mathbf{s}_r^\top\right) \quad (15)$$

where $\text{tr}(\cdot)$ is the trace of a matrix, and $\tilde{\beta}_1$ takes a value between 0 and 3. The closed-form solution of the above problem is as follows:

$$\mathbf{A} = \left[\sum_r f_r \mathbf{t}_r \mathbf{s}_r^\top + \beta_1 \mathbf{I}\right] \left[\sum_r f_r \mathbf{s}_r \mathbf{s}_r^\top + \beta_1 \mathbf{I}\right]^{-1} . \quad (16)$$

3.2. Discriminative linear regression

Another approach is the discriminative linear regression, which is first proposed in [5] for Chinese OCR adaptation. To learn the transformation, the SSM-MCE objective function is defined as follows:

$$l(\mathcal{Y}; \mathbf{\Lambda}, \mathbf{\Theta}) = \frac{1}{R'} \sum_{r=1}^{R'} \frac{1}{1 + \exp[-\alpha d(\mathbf{y}_r; \mathbf{\Lambda}, \mathbf{\Theta}) + \beta]} \quad (17)$$

where

$$d(\mathbf{y}_r; \mathbf{\Lambda}, \mathbf{\Theta}) = \frac{-g_p(\mathbf{x}_r; \lambda_p) + g_q(\mathbf{x}_r; \lambda_q)}{2 \parallel \mathbf{m}_{p\hat{k}} - \mathbf{m}_{q\bar{k}} \parallel} . \quad (18)$$

The notations in Eq. (18) are defined in Eq. (5), Eq. (6), Eq. (7), and Eq. (13). The optimization procedure for $\mathbf{\Theta}$ is the same as in [5]. From the viewpoint of classification measure, SSM-MCE seems a more reasonable objective function to learn the feature transform compared with STM, which is also confirmed by our experiments.

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. As for writer adaptation, the handwriting data of input method on mobile devices from 105 real users collected in several months are used for our experiments. The number of character samples for each user is in a range from 5000 to 30000. For each writer, one half of samples are randomly selected as adaptation data to learn the feature transformation while the other half is used as testing data. 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 [4]. The tuning parameters of DNN are set according to [9]. 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 classifiers with different features and different training criteria on the testing set of all 105 writers. "LBG" denotes a system trained using LBG clustering while "SSM-MCE" refers to a system trained by the SSM-MCE criterion. "LDA" and "BNF" represent the two types of features. The DNN architecture for bottleneck feature is 96-1024-1024-1024-96-15167, which denotes that the sizes are 96 for input layer, 1024 for three hidden layers, 96 for bottleneck layer (the same dimension as that of original feature vector), and 15167 for output layer. Several observations can be made. First, the BNF systems can achieve significant error reductions over the LDA systems for all testing cases, especially under a small number of prototype setting. Second, the performance gain between BNF

Table 1. Performance (character error rate in %) comparison of systems using prototype-based classifiers with different features and different training criteria on the testing set of all 105 writers.

	#prototype	LBG	SSM-MCE
LDA	1	33.97	22.16
	2	30.63	20.20
	4	27.08	19.14
BNF	1	26.06	19.66
	2	23.56	19.12
	4	22.01	18.79

Table 2. Performance (character error rate in %) comparison of systems using different adaptation strategies averaged across each testing set of all 105 writers.

	LDA		BNF	
	STM	DLR	STM	DLR
Baseline	19.14		18.79	
WA(1000)	13.49	11.96	9.79	9.42
WA(3000)	13.29	10.51	9.31	8.53
WA(5000)	13.24	10.11	9.24	8.14

system and LDA system using LBG clustering is more significant than that using SSM-MCE based discriminative training. This is reasonable because as a highly nonlinear and discriminative transformation, BNF carries more discriminative information than LDA feature, which is more powerful for LBG clustering (like a generative model) and less powerful for SSM-MCE training (like a discriminative model). Finally, the increased footprint (the size of the recognition engine) and runtime latency from LDA system to BNF system is perfectly acceptable under the same prototype setting (e.g., increased by 40% in footprint and 50% in latency for the best performance setting, namely SSM-MCE classifier with 4 prototypes).

Fig. 3 gives a performance comparison of different adaptation strategies for each testing set of 25 selected writers, among which there are more than 5000 adaptation data samples for each writer. "LDA" and "BNF" are two feature types. "Baseline" refers to the best system in Table 1 (i.e., SSM-MCE training with 4 prototypes) without writer adaptation. "STM" and "DLR" represent two criteria for learning the transform parameters. First, for baseline system, the results of LDA feature and BNF are mixed for different writers but BNF can

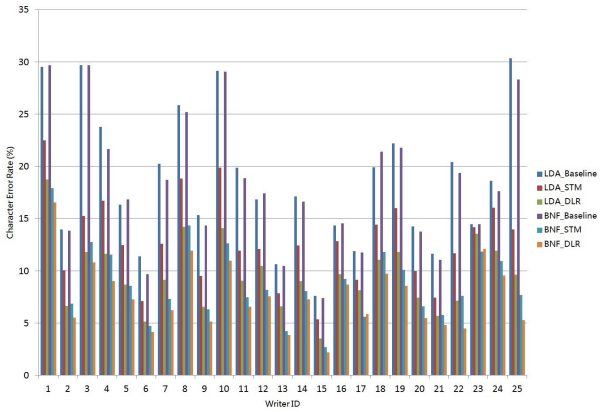

Figure 3. Performance (character error rate in %) comparison of different adaptation strategies for each testing set of 25 selected writers.

achieve an overall better performance like in Table 1. Second, for STM-based adaptation, BNF systems consistently and significantly outperform LDA systems on all testing sets. This is also applicable to DLR-based adaptation where the performance gain between BNF system and LDA system is still remarkable although not that significant compared with STM-based adaptation. Third, for the same feature type, namely LDA feature or BNF, DLR-based adaptation yields better performance than STM-based adaptation and there are only two exceptions on No.17 and No.23 writers. Overall, the system using BNF and DLR-based adpatation can achieve the best recognition performance.

Table 2 lists a performance comparison of systems using different adaptation strategies averaged across each testing set of all 105 writers. The baseline system without adaptation is the same as that in Fig. 3. Three configurations of writer adaptation using different number of adaptation samples are compared, namely 1000, 3000, and 5000 corresponding to "WA(1000)", "WA(3000)", "WA(5000)" in Table 2. First, writer adaptation with LDA feature and STM can achieve about 30% error reduction over baseline system with LDA feature, which is much more significant than those results in [23]. This is most likely because our data samples are collected from real users in which there are many samples for commonly used characters and we just randomly divide them into two halves for adaptation and testing while in [23] the design of adaptation and testing data is for a extremely difficult case where there is no overlap of character classes for adaptation and testing and there is only one sample for each character class. With the increased amount of adapta-

tion data, the performance of the system with LDA and STM is quickly saturated while the error rate of other three adaptation systems using our proposed BNF and DLR techniques is still significantly decreased, especially for DLR-based adaptation. The best system using BNF and DLR yields about 40% error reduction (from 13.24% to 8.14%) over the system using LDA and STM, which is very promising.

5. Conclusion

In this work, we investigate to use bottleneck features and discriminative linear regression for writer adaptation of online handwritten Chinese character recognition. Very promising results are achieved on the data from real applications by comparing to a conventional adaptation approach. As for future work, unsupervised, semi-supervised adaptation, and online adaptation strategy will be further explored.

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] A. Brakensiek, A. Kosmala, and G. Rigoll. Comparing adaptation techinieuqs for on-line handwriting recognition. In *ICDAR*, pages 486–490, 2001.

[3] J. Du and Q. Huo. A discriminative linear regression approach to OCR adaptation. In *ICPR*, pages 629–632, 2012.

[4] 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.

[5] J. Du and Q. Huo. A discriminative linear regression approach to adaptation of multi-prototype based classifiers and its applications for Chinese OCR. *Pattern Recognition*, 46(8):2313C2322, 2013.

[6] 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.

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

[8] 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.

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

[10] 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.

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

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

[13] W. Kienzle and K. Chellapilla. Personalized handwriting recognition via biased regularization. In *ICML*, 2006.

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

[15] 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.

[16] J. D. V. V. N. Matić, I. Guyon. Writer adaptation for on-line handwritten character recognition. In *ICDAR*, pages 187–191, 1993.

[17] J. C. Platt and N. P. Matić. A constructive RBF netwrok for writer adaptation. In *NIPS*, 1997.

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

[19] V. Vuori and T. Korkeakoulu. *Adaptive methods for online recognition of isolated handwritten characters*. PhD thesis, Helsinki University of Technology, 2002.

[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] Y.-Q. Wang and Q. Huo. Building compact recognizers of handwritten Chinese characters using precision constrained Gaussian model, minimum classification error training and parameter compression. *International Journal on Document Analysis and Recognition*, 14(3):255–262, 2011.

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

[23] X.-Y. Zhang and C.-L. Liu. Style transfer matrix learning for writer adaptation. In *CVPR*, pages 393–400, 2011.

[24] X.-Y. Zhang and C.-L. Liu. Writer adaptation with style transfer mapping. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35(7):1773–1787, 2013.