Unsupervised Adaptation of Neural Networks for Chinese Handwriting Recognition

Hong-Ming Yang*, Xu-Yao Zhang*†, Fei Yin*, Zhenbo Luo‡ and Cheng-Lin Liu*†
*National Laboratory of Pattern Recognition (NLPR)
Institute of Automation, Chinese Academy of Sciences
Beijing 100190, China
{hongming.yang, xyz, fyin, liucl}@nlpr.ia.ac.cn
†CAS Center for Excellence in Brain Science and Intelligence Technology
Beijing, China
‡Beijing Samsung Telecom R&D Center
Beijing, China
zb.luo@samsung.com

***Abstract*—Writer adaptation is an important topic in handwriting recognition, which can further improve the performance of writer-independent recognizer. In this paper, we propose combining the neural network classifier with style transfer mapping (STM) for unsupervised writer adaptation, which only require writer-specific unlabeled data, and therefore is more common and efficient compared to supervised adaptation. We use some techniques like dropout, ReLU, momentum, and deeply supervised strategy to improve the performance of the neural network classifier. For a specific writer in the test data, an adaptation layer is added to the pre-trained neural network classifier. In adaptation process, only the parameters in adaptation layer are updated while other parameters of the neural network are kept unchanged. To train the adaptation layer, we use the same technology as STM learning but redefine the source point set, target point set and the corresponding confidence. Experiments on the online Chinese handwriting database CASIA-OLHWDB1.1 demonstrate that our method is very efficient and effective in improving classification accuracy. The experimental results also show that our proposed method outperforms the previous proposed learning vector quantization (LVQ) and modified quadratic discriminant function (MQDF) with STM methods for writer adaptation.**

***Keywords*-neural network; style transfer mapping; writer adaptation; handwriting recognition;**

I. Introduction

The default assumption in many learning scenarios is that training and test data are independently and identically (iid) drawn from the same distribution. However, many applications observe changing distribution. When the distributions on training and test set do not match, we are facing *sample selection bias* or *concept drift* [1]. To improve the generalization performance in such situations, we should transfer the classifier learned on the training data to the new distributions of the test data, which is known as *transfer learning* [2].

The large variability of handwriting styles across individuals makes handwriting recognition a challenging problem. To deal with this variability, writer-independent classifiers which were trained with large training datasets should be adapted towards the new distribution of the particular writer. This process is known as *writer adaptation* [3]. Writer adaptation is a specific example of transfer learning.

A general framework called style transfer mapping (STM) is proposed by [3] for writer adaptation. STM uses a linear transformation to project writer-specific data onto a style free space in order to improve the prediction of the writer-independent classifier. The STM achieved significant improvement in large category handwriting recognition. Li et al. adopted the STM learning method in historical Chinese character recognition research [4]. Feng et al. [5] proposed using a nonlinear transformation to replace the linear transformation in STM, and the nonlinear transformation is based on Gaussian Process regression. This method is proved to be useful for Dunhuang historical Chinese character recognition.

However, these previous studies only used STM for traditional classifiers. In recent years, the deep neural network classifier outperforms the traditional classifiers in many fields, and some adaptation methods with neural network classifier are proposed. For example, Tzeng et al. proposed adding an adaptation layer and together with a dataset transferring loss in convolutional neural network (CNN) to learn an invariant representation for different domains [6]. Du et al. trained a CNN as feature extractor, and proposed a new criterion to learn the transformation for supervised writer adaptation based on the extracted features from CNN [7].

In this paper, we propose combining the neural network classifier with STM [3] for writer adaptation. An adaptation layer is added to the pre-trained neural network, and then updated according to the STM criterion in the adaptation process. Experimental results show that STM is efficient and effective in improving the performance of neural networks for writer adaptation. Recently, Zhang et al. combine the CNN with STM for writer adaptation, and get a new benchmark for online and offline handwritten Chinese character recognition [8]. The rest of this paper is organized

as following. Section II introduces the writer-independent classifiers that are used in this paper. Section III gives a brief introduction of the STM. Section IV then describes the proposed method. Section V gives the experimental results and the final section sets the conclusion.

II. Neural Network Classifier

We use neural network classifier, i.e., multilayer perceptron (MLP) [9] for large category Chinese handwritten character recognition. The MLP has been proposed for a long time, but for Chinese handwriting recognition, the modified quadratic discriminant function (MQDF) classifier is much more popular [10]. To improve the performance of MLP, we use some technologies that are widely used in the deep neural network in recent years, which can be summarized as following.

1. We adopt rectified linear units (ReLU) [11] for the hidden layers. These units use rectifier activation function:

$$s(x) = max(0, x) \tag{1}$$

Rectifier network gives rise to real zeros of hidden activations and thus leads to truly sparse representations, this unit can boost up the performance of the network. Furthermore, from the perspective of the training, the rectifier activation is more linear compared with the sigmoid or hyperbolic tangent activation, thus it can decrease the complexity of the network and make the network easier to train.

2. We add dropout after input layer and every hidden layer. Dropout [12] provides a computationally inexpensive but powerful method to regularize a broad family of models. Specifically, for a specific layer L_i of the network, its output O_i is a vector with dimension d_i, d_i is the number of nodes of the layer. If we add dropout to the layer, then, for every node $j(1 \leq j \leq d_i)$ of the layer, we will randomly set its output O_{ij} to 0 with the probability p, p is a hyper parameter and can be adjusted during the training. Because of the dropout, we actually train different sub-networks of the underlying base network, thus, we train multiple models, and evaluate multiple models on each test sample, which can significantly improve the performance of the network.

3. We use the momentum method [13] during the training. The method of momentum is designed to accelerate learning, especially for high curvature, small but consistent gradients, or gradients that vary greatly from one minibatch to the next. Formally, we introduce variables v that represents the velocity (or momentum) that accumulates gradient, η represents the learning rate, θ represents the parameters to be learned, α represents the weight of the momentum. The update rule is given by:

$$v_t = \alpha_t v_{t-1} - (1 - \alpha_t)\eta_t(\nabla_\theta L_t) \tag{2}$$

$$\theta_t = \theta_{t-1} + v_t \tag{3}$$

α, η are hyper parameters and can be tuned during the training. We fix $\alpha_t = 0.9$ in our experiment.

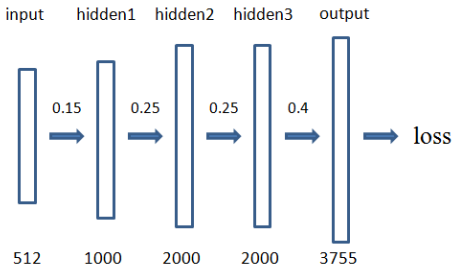

Figure 1. The structure of the neural network classifier C_1 (dropout probabilities are listed on the arrow)

We adopt the above three technologies in MLP, and achieve better result than the MQDF classifier. The structure of our network is shown in Fig. 1, which is denoted as C_1. The MLP is trained as the base classifier. During testing, we adapt the pre-trained MLP towards the new handwriting style of each specific writer. In order to achieve the adaptation, We add an adaptation layer before the output layer (after the third hidden layer). The outputs of the third hidden layer will be the inputs of the adaptation layer. In C_1, the dimension of the inputs to the adaptation layer is 2000, which is too high for the adaptation, because it will make the adaptation to fit a large matrix(2000×2000) and a large biased vector(2000×1).

For better adaptation, we propose a new structure of MLP, which is shown in Fig. 2, and we use symbol C_2 to denote it. In C_2, We add a linear layer after the third hidden layer. The activation of the linear layer is just a linear function, $s(x) = x$, which is used to achieve linear dimensionality reduction. However, The reduction of the dimension will hurt the performance of the MLP. In order to keep the performance, we adopt a strategy called deeply-supervised nets (DSN) [14]. The central idea of DSN is to provide integrated direct supervision to the hidden layers, rather than the standard approach of providing supervision only at the output layer. Therefore, in C_2, we add another output layer after the third hidden layer, and calculate its loss called $loss2$ together with the standard loss called $loss1$. During the training, the objective function is

$$Loss = loss_1 + loss_2 \tag{4}$$

By using the strategy of DSN, we can learn discriminative and robust features in the early layers, meanwhile, the added loss can help avoid the vanishing gradient problem during the training, both advantages will be helpful to improve the performance of the MLP.

For classifier C_1 and C_2, we use the soft-max activation for all the output layers, meanwhile, we use the negative log-likelihood loss for both two classifiers.

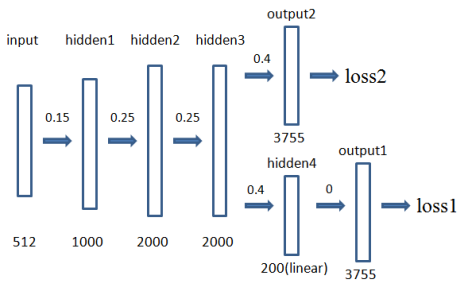

Figure 2. The structure of the neural network classifier C_2 (dropout probabilities are listed on the arrow)

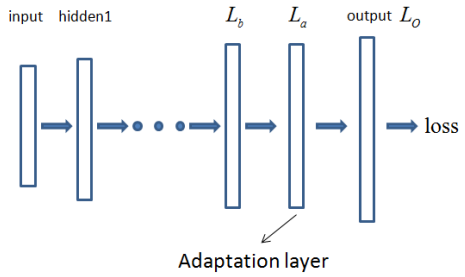

Figure 3. The structure of the neural network for writer adaptation

III. Style Transfer Mapping

As shown in [3], the STM is a linear feature transformation which has a closed-form solution. STM projects the data of different writers onto a style-free space, where the writer-independent classifier needs no change to classify the transformed data and can achieve significantly higher accuracy.

In STM, a source point set is defined as $S = \{s_i \in \mathbf{R}^d \mid i = 1, \ldots, n\}$ and a target point set is defined as $T = \{t_i \in \mathbf{R}^d \mid i = 1, \ldots, n\}$. The parameters of style transfer mapping $A \in \mathbf{R}^{d \times d}$ and $b \in \mathbf{R}^d$ can be learned by minimizing the objective function

$$\min_{A \in \mathbf{R}^{d \times d}, b \in \mathbf{R}^d} \sum_{i=1}^{n} f_i \|As_i + b - t_i\|_2^2 + \beta \|A - I\|_F^2 + \gamma \|b\|_2^2 \quad (5)$$

where $f_i \in [0, 1]$ is the transformation confidence for pair (s_i, t_i); $\|.\|_F$ is the matrix Frobenius norm and $\|.\|_2$ is the vector L_2 norm. This optimization problem is convex, thus it has a closed-form solution as shown by [3]. The β and γ are hyper-parameters to control the tradeoff between style transfer and non-transfer. In STM learning for a specific writer, the source point set is defined as the writer-specific data and the target point set is classifier dependent. Previously, STM is combined with learning vector quantization (LVQ) [15] and MQDF [16] classifier. The confidence f_i setup method is described in [3] which is classifier dependent.

STM can be used for supervised, unsupervised, and semi-supervised adaptation. In this paper, we pay more attention to the unsupervised adaptation. For unsupervised adaptation, the self-training strategy can then be used to learn the labels and the STM simultaneously from the unlabeled data. The specific process can be found in [3].

IV. Neural Network Adaptation with STM

In this paper, we combine MLP with STM for writer adaptation. The neural network classifier is trained with large training datasets from many writers. For adaptation, we add an adaptation layer L_a before the output layer L_O of the trained MLP, as shown in Fig. 3. The activation of L_a is linear for linear transformation. For a specific writer in the test set, the parameters of L_a are learned with writer-specific unlabeled data for writer adaptation, while keeping the rest parameters of the MLP fixed. The main problem is how to learn the parameters of L_a. In this paper, we use the same technique as the STM learning by the redefining of source point, target point, and corresponding confidence from the perspective of neural network.

A. The Source Point Set

Let L_b denotes the layer that stays before the adaptation layer. As introduced in section III, the source point set should be the writer-specific data or transformed writer-specific data. After the feed forward process, the input data get a new representation at the layer L_b and then flow to the adaptation layer. Thus, the source point set in our setting is the new representation at L_b of the writer-specific data. Let ϕ denotes the mapping from the input to the output of L_b, then the source point set is $\hat{S} = \{\phi(s_i) \mid i = 1, \ldots, n\}$ where s_i is writer specific data.

B. The Target Point Set

Both the LVQ and MQDF classifier have the concept of "class center". For LVQ classifier, given a specific class y, it has one or several prototypes m_{yi} of the class y, which can be seen as the "center" of y. For MQDF classifier, given a specific class y, it has the mean μ_y of the class, which can also be seen as the "center" of y. From [3], it is obvious that the definition of the target point set is closely related to the "class center". However, in MLP, there is no concept of "class center" for every class. To define the target point set, we should first adopt some method to calculate the "center" for every class. Given a specific class y, we can find all samples with label y in the training set, denoted by

$$X_y = \{x_{yi} \mid i = 1, \ldots, n_y\} \quad (6)$$

By passing X_y to the MLP, we can get their new representation at layer L_b

$$\hat{X}_y = \{\hat{x}_{yi} = \phi(x_{yi}) \mid i = 1, \ldots, n_y\} \quad (7)$$

thus, the center of the given class y can be computed as

$$c_y = \frac{1}{n_y} \sum_{i=1}^{n_y} \hat{x}_{yi} \quad (8)$$

With the centers for all class, the target point set can be defined directly. For example, let x be a writer-specific sample with label y, the source point is then defined as $\phi(x)$ where the target point should be c_y.

C. Confidence Setup

In MLP classifier, there exists a natural and direct way to decide the confidence for a pair $\{x_i, t_i\}$ where x_i is the input and t_i is the corresponding target. At the output layer L_O, the activation is soft-max, thus, it outputs the probability vector $P = \{p_i\}_{i=1}^{n_c}$ for a given input (n_c is the number of classes), p_i is the confidence that the input belonging to class i. Normally, the label given by the MLP classifier to the input is $\hat{y} = \arg\max_{i=1}^{n_c} p_i$ and the confidence is $p_{\hat{y}}$. Therefore, we can directly use this as the confidence representing the transformation from source point to target point. Formally, we set the confidence as $f_i = \max_{i=1}^{n_c} p_i$.

Given the source point set, target point set, and confidence, we can get the parameters of adaptation layer L_a by solving the equation (5). Particularly, We learn the parameters of layer L_a only with the unlabeled writer-specific data, thus the writer adaptation is unsupervised. For unsupervised adaptation, we still use the self-training strategy that learns the labels and the parameters of L_a simultaneously. Our complete algorithm is summarized in Algorithm 1.

Algorithm 1 Neural Network Classifier with STM for Unsupervised Adaptation

Input:
writer-specific unlabeled data $\{x_i\}_{i=1}^n$
the trained base Neural Network classifier C_{MLP}
hyper-parameter β, γ, iterNum
insert a adaptation layer L_a before the output layer of C_{MLP}, the weight of L_a is initialized by $W = I$, the bias of L_a is initialized by $b = 0$
for iter=1 to iterNum **do**
  for $i = 1$ to n **do**
    feed forward x_i in the C_{MLP}
    calculate the source $s_i = \phi(x_i)$
    prediction $\hat{y}_i$ by C_{MLP}
    target $t_i = c_{\hat{y}_i}$
    get the confidence f_i from soft-max
  end for
  learn $\{W, b\}$ by E.q. (5) using $\{s_i, t_i, f_i\}_{i=1}^n$
end for
Output: predicted labels $\{\hat{y}_i\}_{i=1}^n$

V. Experiments

A. Database

We evaluated the performance of MLP classifier and MLP+STM for writer adaptation on a large scale unconstrained Chinese online handwriting database CASIA-OLHWDB1.1 [17], which consists handwritten character data from 300 different writers (no. 1001-1300). For each writer, there are around 3755 samples. The number of character class is 3755. The training set contains 898,573 samples from writers no. 1001-1240, and the test set contains 224,559 samples from writers no. 1241-1300.

Table I
TEST ACCURACIES OF DIFFERENT CLASSIFIERS ON OLHWDB1.1

Classifier	Accuracy (%)
MQDF	93.22
C_1	93.29
C_2	93.31

Table II
SETTINGS OF HYPER-PARAMETERS FOR ADAPTATION

Classifier	β	γ	iterNum
C_1	1000000	0	5
C_2	2100000	0	5

B. Neural Network Classifier

The neural network classifier plays as a base classifier in this paper. As introduced in section II, we trained two different neural network classifier C_1 and C_2. The structure of them are shown in Fig. 1 and Fig.2. For classifier C_2, during testing, we remove the output2 and loss2, and only use the standard output1 for classification. The accuracies on the OLHWDB1.1 are shown in Table I.

Both two neural network classifiers outperform the MQDF classifier. Compared with classifier C_1, although the dimension of the input to the output layer is decreased, the classifier C_2 still achieve better performance. This shows that the strategy of DSN really works.

C. Overall Performance

We adapted the trained classifier C_1 and C_2 to every writer in the test set of OLHWDB1.1 by Algorithm 1. To measure the performance of writer adaptation, the measurement called *error reduction rate* [3] is used in our experiments. The parameter settings of our experiments are shown in Table II. For almost all the writers in the test data, the adaptation is helpful, but the ranges of the boost are different across the writers. Table III shows the error rate of 10 representative writers and the average error rates over 60 writers for C_1 and C_2. Table IV shows the comparison of our proposed Neural Network + STM model with the MQDF/LVQ + STM model.

Table III shows that our proposed model is efficient for writer adaptation, the error rate after adaptation is significantly reduced. The adaptation handles 2000-dimension inputs for C_1, whereas 200-dimension inputs for C_2. From

Table III
ERROR RATES (%) WITH AND WITHOUT ADAPTATION FOR C_1 AND C_2.

Writer no.	C_1			C_2		
	Before adaptation	After adaptation	Reduction (%)	Before adaptation	After adaptation	Reduction (%)
1242	36.77	34.34	6.61	36.50	33.40	8.49
1255	3.25	2.34	28.00	3.01	2.37	21.26
1259	6.45	4.93	23.57	7.10	5.03	29.15
1261	14.33	8.96	37.47	14.81	9.07	38.76
1266	3.64	2.19	39.84	3.64	2.30	36.81
1273	5.32	4.81	9.59	4.84	4.25	12.19
1280	2.26	2.00	11.50	2.32	1.55	33.19
1285	6.50	4.53	30.31	6.53	4.35	33.38
1292	3.70	2.80	24.32	3.70	2.56	30.81
1299	3.94	3.00	23.86	3.69	2.84	23.04
average	6.71	5.55	17.29	6.69	5.41	19.13

Table IV
COMPARISON OF THE ADAPTATION PERFORMANCE FOR DIFFERENT CLASSIFIERS.

Classifier	Before adaptation	After adaptation	Reduction (%)
LVQ(1) [3]	10.36	9.05	12.64
MQDF(10) [3]	8.33	7.11	14.65
C_1	6.71	5.55	17.29
C_2	6.69	5.41	19.13

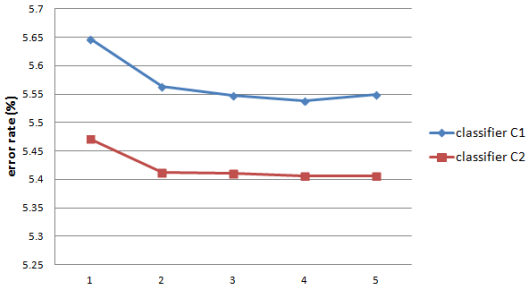

Figure 4. The error rate of different iteration numbers in writer adaptation

Table III, the 200-dimension inputs are better, but the distinction between them is small. Considering the cost for computation, We still recommend to use inputs with lower dimension for adaptation. Compared to MQDF/LVQ classifier, our neural network classifier has a higher baseline, and the result of writer adaptation is still better than the MQDF/LVQ+STM.[1] This again demonstrates the efficiency of our model.

D. Effects of Iteration Numbers

In this experiment, we evaluate the effects of iteration numbers for writer adaptation. We changed the iteration numbers on C_1 and C_2, and keep the other hyper-parameters fixed. The results are shown in Fig. 4. From Fig. 4, we can find that with the increase of the iteration number, the error rate will decrease gradually. When the iteration number is small, an addition for the iteration number will significantly improve the performance. However, when the iteration reaches a specific number, the increase is helpless. More iteration will need more computation, considering this, a proper iteration number should be selected in practical application, which can simultaneously keep the performance and decrease the cost of computation.

[1]Note that the experimental setting in this paper is not exactly the same as [3], and Table IV just gives an intuitive comparison.

E. Effects of β

In Algorithm 1, β control the tradeoff between style transfer and nontransfer. In this experiment, we evaluate the effects of the β for writer adaptation. We adopt different β in this experiment while keeping the other hyper-parameters fixed. This experiment is conducted only on C_2, which has a lower computation cost contrast to C_1. The results of this experiment is shown in Fig. 5. In Fig. 5, the error rate first decreases with the increase of β, then, when the β reach a specific number, the error rate will increase. This phenomenon is reasonable. When the β is small, the constraint to the transformation is weak, which will result over transformation and this is harmful to the adaptation. When the β is too big, the constraint to the transformation is strong, this will make the transformation approach a identity transformation, thus, the adaptation is weak and helpless. The β is important for the writer adaptation, it will significantly effect the performance. In practical application, the β should be carefully selected for better adaptation.

VI. CONCLUSION

This paper presents a new method that combines neural network classifier with style transfer mapping technology for unsupervised writer adaptation. For training the neural

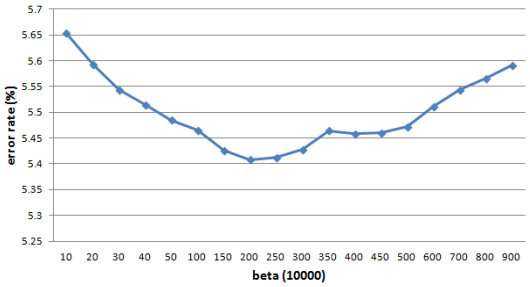

Figure 5. The error rate of different β for classier C_2 with writer adaptation

network classifier, some techniques like dropout, ReLU, momentum, and deeply supervised strategy are used to improve the performance of the classifier. During testing, an adaptation layer is added to the classifier and trained for writer adaptation. The method to train the adaptation layer is similar to the STM learning, but the source point set, target point set and confidence setting method are redefined in our model. The adaptation is unsupervised in our model, and we use the self training strategy to handle this. Experiments on a large scale online Chinese handwriting database demonstrate that our model is efficient, which can significantly reduce the error rate for handwriting recognition.

The linear activation of the adaptation layer leads to linear transformation. In future research, we will explore the new structure of the adaptation layer for nonlinear transformation. We use the STM learning method to train the adaptation layer, This is efficient but not a natural way to train the neural network. In future, we will consider more suitable training methods for the adaptation layer. Another straight-forward extension is using STM for the adaptation of deep convolutional neural networks.

ACKNOWLEDGMENT

This work has been supported in part by the National Basic Research Program of China (973 Program) Grant 2012CB316302, the National Natural Science Foundation of China (NSFC) under Grants 61403380 and 61573355, the Strategic Priority Research Program of the Chinese Academy of Sciences (Grants XDA06040102 and XDB02060009).

REFERENCES

[1] Mark G Kelly, David J Hand, and Niall M Adams. The impact of changing populations on classifier performance. In *Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 367–371. ACM, 1999.

[2] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[3] Xu-Yao Zhang and Cheng-Lin Liu. Writer adaptation with style transfer mapping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1773–1787, 2013.

[4] Bohan Li, Liangrui Peng, and Jingning Ji. Historical chinese character recognition method based on style transfer mapping. In *2014 11th IAPR International Workshop on Document Analysis Systems*, pages 96–100. IEEE, 2014.

[5] Jixiong Feng, Liangrui Peng, and Franck Lebourgeois. Gaussian process style transfer mapping for historical chinese character recognition. In *SPIE/IS&T Electronic Imaging*, pages 94020D–94020D, 2015.

[6] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[7] Jun Du, Jian-Fang Zhai, Jin-Shui Hu, Bo Zhu, Si Wei, and Li-Rong Dai. Writer adaptive feature extraction based on convolutional neural networks for online handwritten chinese character recognition. In *2015 13th International Conference on Document Analysis and Recognition*, pages 841–845. IEEE, 2015.

[8] Xu-Yao Zhang, Yoshua Bengio, and Cheng-Lin Liu. Online and offline handwritten chinese character recognition: A comprehensive study and new benchmark. *Pattern Recognition, accepted*, 2016.

[9] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive Modeling*, 5(3):1, 1988.

[10] Cheng-Lin Liu, Fei Yin, Da-Han Wang, and Qiu-Feng Wang. Online and offline handwritten chinese character recognition: benchmarking on new databases. *Pattern Recognition*, 46(1):155–162, 2013.

[11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *AISTATS*, volume 15, pages 315–323, 2011.

[12] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[13] Boris Teodorovich Polyak. Gradient methods for solving equations and inequalities. *USSR Computational Mathematics and Mathematical Physics*, 4(6):17–32, 1964.

[14] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *AISTATS*, volume 2, pages 562–570, 2015.

[15] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[16] Fumitaka Kimura, Kenji Takashina, Shinji Tsuruoka, and Yasuji Miyake. Modified quadratic discriminant functions and the application to chinese character recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (1):149–153, 1987.

[17] Cheng-Lin Liu, Fei Yin, Da-Han Wang, and Qiu-Feng Wang. Casia online and offline chinese handwriting databases. In *Proceedings of the 11th International Conference on Document Analysis and Recognition*, pages 37–41. IEEE, 2011.