2018 16th International Conference on Frontiers in Handwriting Recognition

Deep Transfer Mapping for Unsupervised Writer Adaptation

Hong-Ming Yang[1,2], Xu-Yao Zhang[1,2], Fei Yin[1,2], Jun Sun[4], Cheng-Lin Liu[1,2,3]
[1]NLPR, Institute of Automation, Chinese Academy of Sciences, Beijing, P.R. China
[2]University of Chinese Academy of Sciences, Beijing, P.R. China
[3]CAS Center for Excellence of Brain Science and Intelligence Technology, Beijing, P.R. China
Email: {hongming.yang, xyz, fyin, liucl}@nlpr.ia.ac.cn
[4]Fujitsu Research & Development Center, Beijing, P.R. China
Email: sunjun@cn.fujitsu.com

***Abstract*—Convolutional neural network (CNN) has achieved great success in handwriting recognition. However, it relies on large set of labeled data in training and its performance will deteriorate when the data distribution varies. To solve this problem, traditional methods usually consider adaptation of the single top layer of CNN. To better reduce the distribution discrepancy, in this paper, we consider adaptation of all layers of CNN including both convolutional and full layers. Four variations of transformations are designed based on different assumptions about the space relations for adaptation of convolutional layers. In order to make adaptation of multiple layers, we propose to cascade the transformations of different layers to conduct adaptation in a deep manner, and therefore this method is denoted as deep transfer mapping (DTM). DTM can capture the information from different layers and minimize the data divergence under different information abstract levels, thus it is more powerful and flexible for domain adaptation. Experiments on the online Chinese handwriting dataset (OLHWDB) demonstrate the efficiency and effectiveness of the proposed method for unsupervised writer adaptation.**

I. Introduction

Handwriting recognition is an important task in pattern recognition. In past years, various methods have been proposed to handle this problem [1]. Recently, benefited from the fast development of deep learning, both the convolutional neural network (CNN) [2] and recurrent neural network (RNN) [3] have been successfully applied in online and offline handwriting recognition. Despite the high accuracy, most methods still suffer from the changing distributions of the test data. In real world, the test samples usually come from various writers (with different writing styles), together with different writing tools (different pens or electronic writing devices), which will make the handwriting data in the real world have large differences with the training data. In such situations, the classifiers trained on the fixed training sets can not achieve satisfactory performance during the test process.

To deal with the performance degeneration under the condition of different data distributions on the test domain (target domain), various methods have been proposed for adapting the base classifier to different writers or different distributions of the target domain data, which is known as writer adaptation. There are three main category of writer adaptation methods for CNNs. The first one is the fine-tuning based methods,

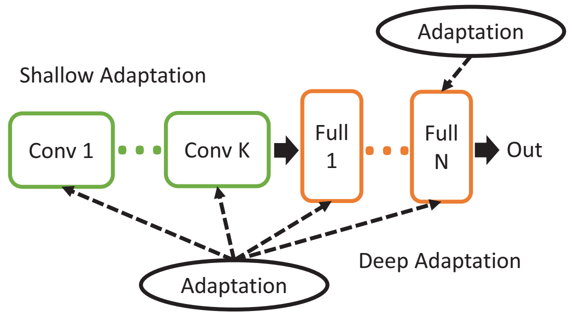

Fig. 1. Traditional methods usually only consider adaptation on the single top layer. This paper considers deep adaptation of all layers for a convolutional neural network.

i.e., use some labeled target domain data to fine-tune the top layers of the CNN, which has been trained on the training data (source domain data). The second category of writer adaptation methods is mainly based on representation learning. In these methods, different loss functions are designed for learning domain-invariant representations in the second-top layer of the CNN. Under such representations, the differences between the source and target domains are minimized and the classifiers trained on the source domain can be directly applied in the target domain. The last kind of writer adaptation methods mainly concentrates on learning projections between the source and target domains. They usually model the projection learning process as an optimization problem and by solving this problem, we can get an ideal projection. After applying this projection to the source or target domain data, the resulted data will share the same domain in which we can directly train classifiers and make predictions. However, as shown in Fig. 1, most of these methods consider adaptation only on the top single fully-connected layer, while ignoring the other layers and therefore perform adaptation only on a shallow level.

How to efficiently and effectively make adaptation on convolutional layers is not well-studied. In this paper, we propose four types of transformations particularly designed for adaptation of convolutional layers, according to different assumptions on the forms of the transformation. Afterwards, as shown in Fig. 1, we further propose to conduct adaptation on multiple layers of the CNN, including the convolutional

978-1-5386-5875-8/18/$31.00 ©2018 IEEE
DOI 10.1109/ICFHR-2018.2018.00035

layers and fully-connected layers, not merely on the top layers, and we call this method as deep transfer mapping (DTM). DTM performs domain adaptation in a deep manner, thus it can capture more information and proceed adaptation under different information abstract levels, which can further improve the performance of the adaptation. In particular, our proposed methods are totally unsupervised, we do not need any labeled samples in the target domain, which makes our methods more general and applicable in the reality.

The rest of this paper is organized as following. Section II introduces the related works; Section III gives a brief review for the style transfer mapping (STM) method and presents our proposed adaptation methods for the convolutional layers. In section IV, we introduce the deep transfer mapping (DTM) method for unsupervised writer adaptation. After that, the experimental results are presented in Section V and the conclusions are given in Section VI.

II. Related works

Previously, Zhang and Liu [4] proposed learning a linear transformation to project the writer-specific data onto a style-free space for writer adaptation, which was further expanded and applied to CNN for unsupervised adaptation [2]. Feng et al. [5] used a nonlinear transformation of Gaussian progress regression to replace the linear transformation presented in [4], which further improved the transfer power of the projection. Du et al. [6] contributed a new criterion to learn the projection for supervised writer adaptation. For writer adaptation methods based on CNNs, Tang et al. [7] used some labeled target domain data to fine-tune the CNN learned on the source domain for semi-supervised adaptation. Tang et al. [8] first fine-tune the trained CNN, then combined the multiple kernel maximum mean discrepancies (MK-MMD) loss [9] with the traditional cross entropy loss to further train the CNN for semi-supervised adaptation.

Writer adaption is a specific problem of domain adaptation. The multiple kernel maximum mean discrepancies (MK-MMD) loss proposed in [9] can be used for unsupervised adaptation of CNN. Ghifary et al. [10] added an auto-encoder pipeline to the traditional CNN for reconstructing both the source and target domain data, which can ensure the encoder to learn shared features between source and target domains for unsupervised adaptation. Tzeng et al. [11] designed a domain confusion loss based on a domain classifier to train the CNN to learn domain-invariant features. Besides, they also proposed a soft label based method to further fine-tune the learned CNN. Both of these methods are effective for semi-supervised adaptation. Ganin et al. [12] proposed a domain adversarial loss, which is also based on a domain classifier, to learn domain-invariant features with a CNN for unsupervised domain adaptation.

However, most of these methods perform adaptation only on a shallow level, i.e., on the top single fully-connected layer, and the deep transfer together with the convolutional layers had not been stressed in previous works.

III. Adaptation on convolutional layers

Most previous methods perform adaptation only on the fully-connected layers, which ignore the adaptation property of the convolutional layers. In this paper, we propose a transfer mapping based method for adaptation on convolutional layers. In this section, we first give a brief review of the style transfer mapping method for adaptation, then we introduce our proposed adaptation method for the convolutional layers.

A. Style transfer mapping in CNN

Let $D_s = \{(x_i^s, y_i^s)_{i=1}^{n_s}\}$ and $D_t = \{(x_i^t)_{i=1}^{n_t}\}$ denote the source and target domain data respectively. In the reality, the distribution on the source domain p_s is different with the distribution on the target domain p_t, i.e., $p_s \neq p_t$, which significantly influences the generalization performance of the learned classifier on D_t.

To deal with this problem, a domain adaptation method called style transfer mapping (STM) was proposed in [4], [2]. Firstly, a base CNN is trained on D_s and let ϕ denotes the map from the input to the second-top layer output of the learned CNN (CNN feature extractor). For the data in D_s and D_t, we can extract their features with ϕ, then we have:

$$\widehat{D_s} = \{(\phi(x_i^s), y_i^s)_{i=1}^{n_s}\}, \quad \widehat{D_t} = \{(\phi(x_i^t))_{i=1}^{n_t}\}. \tag{1}$$

Because $p_s \neq p_t$, naturally we have $\widehat{p_s} \neq \widehat{p_t}$. To minimize the discrepancy between $\widehat{p_s}$ and $\widehat{p_t}$, a linear transformation is learned to project the data from $\widehat{D_t}$ to $\widehat{D_s}$. The learning of the linear transformation is modeled as:

$$\min_{A \in \mathbf{R}^{d \times d}, b \in \mathbf{R}^d} \sum_{i=1}^{N_t} f_i \| A\phi(x_i^t) + b - t_i \|_2^2 + \beta \| A - I \|_F^2 + \gamma \| b \|_2^2, \tag{2}$$

where A and b denote the parameters of the linear transformation, f_i denotes the transformation confidence and t_i denotes the target of x_i^t in the transformation, which is computed by:

$$t_i = c_{\hat{y}_i}, \tag{3}$$

where $\hat{y}_i$ is the predicted pseudo label of sample x_i^t by the trained CNN, and c_y represents the mean of a specific class y computed on $\widehat{D_s}$, i.e.,

$$c_y = \frac{1}{n_y} \sum_{i=1}^{n_y} \phi(x_{yi}^s). \tag{4}$$

The object in equation 2 has a closed solution, which is shown in [4]. By applying the learned transformation on $\widehat{D_t}$, we have:

$$\widetilde{D_t} = \{(A\phi(x_i^t) + b)_{i=1}^{n_t}\}. \tag{5}$$

After projection, the resulted $\widetilde{D_t}$ will have smaller discrepancy with $\widehat{D_s}$, thus the linear discriminant performed by the softmax layer of the trained CNN will have better performance on $\widetilde{D_t}$. Note that the linear transformation can be embedded into the CNN by insert an adaptation layer after the second-top layer, the activation of the adaptation layer must be identical and its weight and bias must be assigned with the solved A and b respectively.

After the adaption, we will have more accurate pseudo labels and more reasonable classification confidences, thus we can get better f_i and t_i, then we can repeat the process introduced above for better performance. Besides, STM is totally unsupervised, we do not need any labeled samples in the target domain.

B. Adaptation methods on convolutional layers

For a specific convolutional layer l, let p_s^l and p_t^l denote the distributions for the output of source and target domain data on layer l respectively. In the domain divergence case, we have $p_s \neq p_t$, thus $p_s^l \neq p_t^l$. Inspired by the style transfer mapping method introduced in section III-A, we propose to learn a linear transformation to project the outputs of layer l for decreasing the differences between the domains. However, the output of the convolutional layer l for an input sample is a three-dimensional tensor, which is more difficult to handle compared with the one-dimensional vector outputted by the fully-connected layer. In our work, we analyze different space relations in the output of layer l and propose different forms of linear transformations to project the output of target domain data for adaptation. Meanwhile, we also provide the corresponding learning methods for the different kinds of linear transformations. We mainly consider four kinds of space relations in the output of layer l: fully associated (FA), partly associated (PA), weakly independent (WI), and strongly independent (SI). For each of these situations, we design relevant domain adaptation method based on different forms of the designed mapping.

1) Fully associated adaptation (FAA): For layer l, we denote its output for an input sample x_i as:

$$o_i = \{d_{cjk}\}_{c=1,j=1,k=1}^{c=C,j=H,k=W}, \tag{6}$$

where c, j, k denote the index of the channels, rows of the feature map, columns of the feature map in the output of layer l respectively. Layer l has C feature maps and resulted size of the feature maps is $H \times W$ ($H = W$ in most cases). In the fully associated (FA) case, we consider all space positions of (c, j, k) in the output o_i are related to each other. Therefore, after the linear projection, each position (c', k', j') in the resulted o_i' should be related to the values of all space positions in o_i. To realize this object, we first expand the three-dimensional o_i to a long vector v_i and the dimension of v_i is CHW. For the resulted vector v_i, we use a couple of transformation matrix and bias:

$$A \in \mathbb{R}^{CHW \times CHW}, \quad b \in \mathbb{R}^{CHW} \tag{7}$$

to project it for adaptation. After projection, we have:

$$v_i' = Av_i + b. \tag{8}$$

For each item $(v_i')_j$ in v_i', we have:

$$(v_i')_j = \sum_{k=1}^{CHW} A_{jk}(v_i)_k + b_j. \tag{9}$$

From equation 9, we can see that each position in v_i' is related to all positions in the o_i, which conforms to the fully associated assumption. After projection, we reshape v_i' to a three-dimensional tensor with size $C \times H \times W$, then it can continue to forward through the CNN.

To learn the parameters A, b in the linear transformation, we adopt the same strategy as STM by solving the optimization problem defined in equation 2, which has a closed and exact solution.

The fully associated (FA) is the most complicated case, it considers the relations between all the positions in o_i. Meanwhile, the corresponding FAA is also the most flexible and powerful adaptation method. However, in FAA, we need to learn a lot of parameters, with the number of $CHW(CHW+1)$. This greatly effects the speed and storage efficiency of FAA. In particular, for the bottom layers, where the outputs do not transit (or get through only few) pooling layers, the resulted feature map size ($H \times W$) is very large, making the number of the parameters even larger, which greatly limits the application of FAA in these layers.

2) Partly associated adaptation (PAA): The assumption in FA is very strong, to relax the constraint for the space relations in o_i, we propose a weaker assumption: partly associated (PA). In PA, we consider the positions belong to the same feature maps are all related to each other, but the different feature maps are mutually independent. Due to the independence of the feature maps, we should learn the transformation for each feature map respectively. For a specific feature map m_c, taking into account the associations of all positions within it, we use the same strategy as FAA to learn the linear transformation. Specifically, We first expand the two-dimensional feature map m_c into a vector v_c with the length of $H \times W$, then we use a pair of transformation matrix and bias:

$$A \in \mathbb{R}^{HW \times HW}, \quad b \in \mathbb{R}^{HW} \tag{10}$$

to project it for adaptation. As discussed in FAA, such operations can ensure the space relations for all positions within the feature map m_c. After projection, we should reshape the resulted vector v_c' to two-dimensional feature map with size $H \times W$ for further feedforward through the CNN. Similarly, the parameters A, b can also be learned via solving the optimization problem defined in equation 2, we only need to change the output from $\phi(x_i)$ to $(v_c)_i$. This process should be proceeded for C times separately for learning the transformations for each feature map independently.

Compared with FA, the constraints in PA are weaker and the number of parameters is also smaller, with only $CHW(HW+1)$ parameters. Even though the PAA is not as powerful and flexible as FAA, it needs less computations and storage spaces, which makes it applicable in most situations.

3) Weakly independent adaptation (WIA): Both the FA and PA consider the relations between the positions but in different level. FA consider global relations and PA consider relations only within the same feature map. In this subsection, we further relax the constraints and consider an extreme situation: weakly independent (WI). In WI, we assume that all positions of (c, j, k) in o_i are independent with each other and there does not exist any space relations in o_i. Under this assumption, the linear transformation should be learned and applied for each position (c, j, k) respectively. For a specific position (c_0, j_0, k_0) in o_i, its value is a real number, thus the relevant linear transforation only includes two parameters: a scale factor a and an offset parameter b. The projection is conducted as:

$$(o'_i)_{c_0,j_0,k_0} = a(o_i)_{c_0,j_0,k_0} + b. \tag{11}$$

To learn the parameters a, b, we adopt the same strategy as STM but modify the object function in equation 2, the resulted optimization object is:

$$\min_{a,b\in\mathbf{R}} \sum_{i=1}^{N_t} f_i(a(o_i^t)_{c_0,j_0,k_0} + b - (t_i)_{c_0,j_0,k_0})^2 + \beta(a-1)^2 + \gamma b^2. \tag{12}$$

In equation 12, o_i^t represents the output of the layer l for sample $x_i^t \in D_t$, f_i and t_i have same meanings as described in section III-A. To learn the transformations for all positions in o_i, we should repeat this process for CHW times. However, different positions in o_i share the same form of solution for object 12, thus their parameters can be learned in a parallel way, which ensure the computation efficiency of the weakly independent adaptation (WIA) method.

Based on FA and PA, WI further simplifies the constraints of the space relations in o_i and considers adaptation for each position (c, j, k) separately. The number of parameter in WIA is $2CHW$, which is far smaller than the parameters in FA and PA. In WIA, the one-dimensional linear function limits the power and flexibility of the adaptation, but it adds some implicit generalization for the adaptation, which can prevent the transformation from over-fitting. Meanwhile, compared with FAA and PAA, WIA consumes less computation resources and storage spaces, making it more applicable in large scale CNNs with large size of inputs.

4) Strong independent adaptation (SIA): In the batch normalization (BN) [13] layer, after normalizing the data, a one-dimensional linear function is followed to further project the data. For data within the same feature map, they share the same linear function, i.e., share the same scale factor a and offset factor b. Inspired by this, we further simplify the linear transformation in WIA and force the positions in the same feature maps share a same transformation. In other words, we only need to learn one linear transformation (with two parameters) for each feature map in the output of layer l. In this situation, the positions within the same feature maps are viewed equally and different feature maps are independent with each other, we call this situation as strong independent (SI).

In SI, for a specific feature map c_0, in order to learn the corresponding transformation, we use the same strategy as STM but further modify the optimization object in equation 2. Due to the shared parameters for each feature map, the object

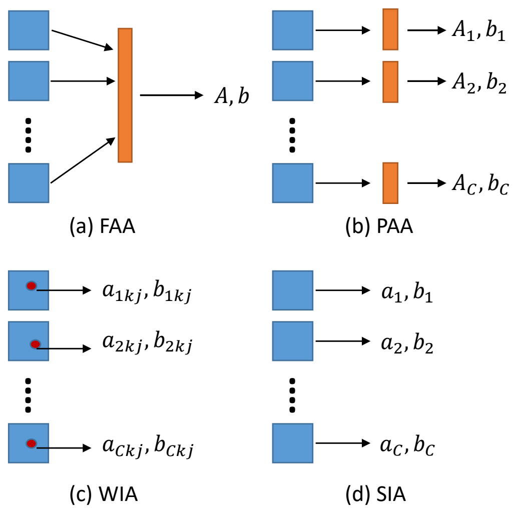

Fig. 2. Different adaptation methods for convolutional layers.

function is re-defined as:

$$\min_{a,b\in\mathbf{R}} \sum_{i=1}^{N_t}\sum_{j=1}^{H}\sum_{k=1}^{W} f_i(a(o_i^t)_{c_0,j,k} + b - (t_i)_{c_0,j,k})^2 + \beta(a-1)^2 + \gamma b^2. \tag{13}$$

This optimization problem is also convex and it has a closed form solution. Similar to WIA, the solutions for different feature maps can also be computed parallelly from the data.

SIA only has $2C$ parameters in the transformation, it is the simplest case. Though decreased in the transfer power and flexibility, the increased computation and storage efficiency further enhance its practicability in real applications.

A graphic description of these domain adaptation methods can be seen in Fig. 2.

IV. Deep Transfer Mapping for Adaptation

It is well known that different CNN layers abstract the information in different levels. The information in the top layers are highly abstracted and they include more semantic representations. However, in the bottom layers, the information have not been deeply abstracted and they are more general. In most previous methods, the adaptations are performed only on the top layers (on one layer in most case), they only capture the abstract information but ignore the different information in the bottom layers. In order to handle the information comprehensively and minimize the domain discrepancy under different abstract levels, we propose to perform adaptation in multiple layers of the CNN with different depth. We use the STM for adaptation in the fully-connected layers and apply the methods proposed in section III-B for adaptation in the convolutional layers. We conduct adaptation in a deep manner and we call this method as deep transfer mapping (DTM).

Specifically, we first choose several layers from CNN (has been trained on the source domain) and insert one adaptation layer after each of them. The adaptation layers act as the linear transformation in the adaptation process and they are initialized as identical mappings. In particular, the parameters of the adaptation layers are not updated according to the corresponding gradients, but assigned with the solutions of the corresponding optimization problems, as described in section III. We perform adaptation and update the corresponding adaptation layer from bottom layer to top layer orderly, and when we update one adaptation layer, we keep the other adaptation layers fixed.

To choose more appropriate layers for adaptation, we first investigate the adaptation property for each layer in the CNN. To do this, we perform adaptation after each layer of the CNN separately and study the corresponding adaptation performance. For layers which contribute significant improvements in performance after adaptation, we consider they have better adaptation property and conduct adaptation after them.

Compared with the previous methods which perform adaptation only on one top layer, the DTM has two main advantages: (1) DTM cascades multiple disjointed transformations from different locations of the CNN for projection, hence it is more powerful for flexibly aligning the distributions between the domains; (2) DTM captures more comprehensive information and minimize the discrepancy of the distributions under different abstract levels. Therefore, DTM is a logical and suitable method for domain adaptation, which can further improve the performance of the base CNN on the target domain.

V. Experiments

A. Datasets

We use the data from CASIA-OLHWDB1.0-1.2 [14] as the training dataset, and use the ICDAR-2013 online competition dataset [15] as the test dataset. The training and test datasets include 2,697,673 and 224,590 samples respectively, which fall into 3755 classes. The training and test data come from different writers, due to the large variability of writing styles between the writers, the data distribution on the training set is very different from the distribution on the test set, thus the datasets are very befitting to test the performance of different domain adaptation methods. In particular, the test data are contributed by 60 different writers, due to the divergence of the writing styles, we view each writer's data as an independent target domain. In other words, the test dataset includes 60 different target domains, and the base classifier learned on the training set must be adapted to each of these target domains respectively.

B. Base CNN classifier

We use CNN as the base classifier, to guarantee its performance, we adopt the same network structure and training method as [2]. Meanwhile, we also apply the same data pre-processing method used in [2]. The details of the CNN framework are presented in Table I. After training, the CNN can achieve the accuracy of 97.55% on the test dataset.

C. Experiments for different conv-layer adaptation methods

We propose four variations of the domain adaptation method for convolutional layers in section III-B. They are based on different assumptions about the space relations in the output together with different forms of transformations. To investigate their ability for adaptation, we apply these domain adaptation methods after the same convolutional layer respectively and observe their performance. The corresponding results are shown in Table II.

TABLE I
The Structure of the Base CNN Classifier.

Layer ID	Layer Type	Parameter	Pooling	Drop Rate
0	input	$8(32 \times 32)$	#	0.0
1	conv	$50(3 \times 3)$	#	0.0
2	conv	$100(3 \times 3)$	2×2	0.1
3	conv	$150(3 \times 3)$	#	0.1
4	conv	$200(3 \times 3)$	2×2	0.2
5	conv	$250(3 \times 3)$	#	0.2
6	conv	$300(3 \times 3)$	2×2	0.3
7	conv	$350(3 \times 3)$	#	0.3
8	conv	$400(3 \times 3)$	2×2	0.4
9	FC	900	#	0.5
10	FC	200	#	0.0
11	softmax	3755	#	#

TABLE II
Performance of Different Adaptation Methods After Layer8.

Methods	without	FAA	PAA	WIA	SIA
Test Acc (%)	97.55	97.91	97.71	97.69	97.62
ERR (%)	0	14.69	6.53	5.71	2.86

In Table II, "without" denotes the performance without adaptation and "ERR" represents the error reduction rate, it is computed by:

$$\frac{error_{before} - error_{after}}{error_{before}}, \tag{14}$$

which is frequently used in previous works ([2], [4]) to measure the performance of the adaptation. From Table II, we can see that all of these methods are effective for domain adaptation. After adaptation, the error rate of the base CNN on test dataset really decreases. In particular, we can see the performance of adaptation increases with the transformation ability of the projection used in the adaptation. FAA has the most powerful transformation, thus it achieves the best adaptation performance. However, this is at the cost of more consuming computation time and storage space, which may limit the application of FAA in practice.

D. Experiments for adaptation after different layers

Most previous methods only perform adaptation on the top layers of the CNN, but ignore the discrepancy of the distributions on other layers. Different from previous methods, in this experiment, we perform domain adaptation on each layer of the CNN respectively and investigate the corresponding adaptation property of the layer. To avoid the heavy burden of computation and storage arise in FAA and PAA, we use WIA to do adaptation in convolutional layers. For the fully-connected layers, we use the style transfer mapping (STM) method introduced in section III-A for adaptation. The test accuracies after adaptation are shown in Table III.

TABLE III
ADAPTATION PERFORMANCE FOR DIFFERENT LAYERS IN CNN.

Layer ID	1	2	3	4	5
Test Acc (%)	97.51	97.57	97.61	97.61	97.63
ERR (%)	-1.63	0.82	2.45	2.45	3.27
Layer ID	6	7	8	9	10
Test Acc (%)	97.67	97.67	97.69	97.85	97.91
ERR (%)	4.90	4.90	5.71	12.24	14.69

TABLE IV
ADAPTATION PERFORMANCE FOR DEEP TRANSFER MAPPING.

Layer ID	without	8	$8 \to 9$	$8 \to 9 \to 10$
Test Acc (%)	97.55	97.91	98.00	98.02
ERR (%)	0	14.69	18.37	19.18

From Table III, we can see that the adaptation are effective for most layers of the CNN. After adaptation, the test performance are really improved. Specifically, from bottom layers to top layers, with the increase of the depth, the adaptation performance also increase. As we mentioned in section IV, the information in the top layers are abstract and domain specific, thus the domain discrepancy on these layers are larger, hence the adaptation performance on these layers are better. In the bottom layers of the CNN, the learned information are more general, hence they are applicable across different domains and tasks. Domain divergence in these layers are not obvious, thus adaptation on these layers do not improve the performance a lot. The similar conclusions are also presented in [16].

E. Experiments for deep transfer mapping (DTM)

In this experiment, we perform adaptation on multiple layers of the base CNN to investigate the potential of proposed deep transfer mapping (DTM) for writer adaptation. We cascade three adaptation layers for adaptation, and they are located after layer 8, layer 9 and layer 10 respectively. For adaptation on layer 8 (convolutional), we adopt FAA for adaptation. For layer 9 and layer 10 (fully-connected), we apply the STM for adaptation. The corresponding results are shown in Table IV.

From Table IV, we can see that when perform adaptation only on layer 8, the error rate is significantly reduced, this again demonstrates the effectiveness of the proposed method for adaptation on the convolutional layer. Moreover, when cascade another adaptation layer after layer 9, we can further improve the performance, this demonstrates the effectiveness of the proposed DTM for writer adaptation. However, when we cascade more adaptation layers (after layer 10) for adaptation, the improvements are not obvious. In this situation, the transformations have been saturated, thus performing adaptation on more layers will not help a lot any more.

VI. CONCLUSION

In this paper, we propose a domain adaptation method with four variations for the convolutional layers. The kernel of this method is to learn and apply a linear transformation to project the outputs to decrease the discrepancy between the domains. For the multi-dimensional output in the convolutional layer, we present different assumptions about its space relations and design different kinds of transformations in this method. Based on this, we further propose the deep transfer mapping (DTM) method for unsupervised writer adaptation. DTM cascades multiple disjointed adaptation layers and perform adaptation in a deep manner. By conducting adaptation on multiple layers of the CNN, the DTM can capture the information from different layers and match the distributions under different abstract levels, which can further improve the performance of the adaptation. Corresponding experimental results demonstrate the efficiency and effectiveness of the proposed methods.

ACKNOWLEDGMENT

This work has been supported by the National Natural Science Foundation of China (NSFC) Grants 61721004, 61633021 and 61573355.

REFERENCES

[1] Cheng-Lin Liu, Ruwei Dai, and Baihua Xiao. Chinese character recognition: history, status and prospects. *Frontiers of Computer Science in China*, 1(2):126–136, 2007.
[2] Xu-Yao Zhang, Yoshua Bengio, and Cheng-Lin Liu. Online and offline handwritten Chinese character recognition: A comprehensive study and new benchmark. *Pattern Recognition*, 61:348–360, 2017.
[3] Xu-Yao. Zhang, Fei Yin, Yan-Ming Zhang, Cheng-Lin Liu, and Yoshua Bengio. Drawing and recognizing Chinese characters with recurrent neural network. *IEEE Trans. PAMI*, 40(4):849–862, 2018.
[4] Xu-Yao Zhang and Cheng-Lin Liu. Writer adaptation with style transfer mapping. *IEEE Trans. PAMI*, (7):1773–1787, 2013.
[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] 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 *ICDAR*, pages 841–845, 2015.
[7] Yejun Tang, Liangrui Peng, Qian Xu, Yanwei Wang, and Akio Furuhata. CNN based transfer learning for historical Chinese character recognition. In *DAS*, pages 25–29, 2016.
[8] Yejun Tang, Bing Wu, Liangrui Peng, and Changsong Liu. Semi-supervised transfer learning for convolutional neural network based Chinese character recognition. In *ICDAR*, pages 441–447, 2017.
[9] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. Learning transferable features with deep adaptation networks. In *ICML*, pages 97–105, 2015.
[10] Muhammad Ghifary, W. Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *ECCV*, pages 597–613, 2016.
[11] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *ICCV*, pages 4068–4076, 2015.
[12] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, pages 1180–1189, 2015.
[13] Sergey Ioffe and Christian Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.
[14] Cheng-Lin Liu, Fei Yin, and Da-Han Wang. CASIA online and offline Chinese handwriting databases. In *ICDAR*, pages 37–41, 2011.
[15] Fei Yin, Qiu-Feng Wang, Xu-Yao Zhang, and Cheng-Lin Liu. ICDAR 2013 Chinese handwriting recognition competition. In *ICDAR*, pages 1464–1470, 2013.
[16] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks. In *NIPS*, pages 3320–3328, 2014.