Deep Decision Tree Transfer Boosting

Shuhui Jiang, Haiyi Mao, Zhengming Ding, *Member, IEEE*, and Yun Fu, *Fellow, IEEE,*

***Abstract*— Instance transfer approaches consider source and target data together during the training process, and borrow examples from the source domain to augment the training data, when there is limited or no label in the target domain. Among them, boosting-based transfer learning methods (e.g., TrAdaBoost) are most widely used. When dealing with more complex data, we may consider the more complex hypotheses (e.g., a decision tree with deeper layers). However, with the fixed and high complexity of the hypotheses, TrAdaBoost and its variants may face the overfitting problems. Even worse, in the transfer learning scenario, a decision tree with deep layers may overfit different distribution data in the source domain. In this paper, we propose a new instance transfer learning method, i.e., Deep Decision Tree Transfer Boosting (DTrBoost), whose weights are learned and assigned to base learners by minimizing the data-dependent learning bounds across both source and target domains in terms of the Rademacher complexities. This guarantees that we can learn decision trees with deep layers without overfitting. The theorem proof and experimental results indicate the effectiveness of our proposed method.**

***Index Terms*— Decision tree, deep boosting (DeepBoost), instance transfer learning, transfer boosting.**

I. Introduction

TRANSFER learning is a hot research topic in neural networks and learning systems, which shows effective performance in visual categorization, face recognition, saliency detection, and so on [1]–[7]. Usually, the prediction performance of a learned model degrades if the number of training data is very limited. Transfer learning algorithms target at extracting the knowledge from one or more **source domains** and applying the knowledge to a **target domain** [1], [8]–[10]. Although the training data are more or less out-dated or in a different domain, some parts of the data could still be reused. For instance transfer approaches, both source and target data are considered during the training process [11]–[13]. A small amount of labeled **same-distributed** training data in the target domain are applied to vote the usefulness of each instance of the source domain. Since some of the source-domain training data might be under a different distribution from the target domain, they are called **diff-distributed** training data.

One major challenge of transfer learning lies in formulating an approach that makes full use of the available source-domain data. Instance transfer learning algorithms address this challenge by identifying the relevant instances that would be useful in learning a tuned classifier that classifies target data points correctly [14]–[18]. It is widely used in pedestrian detector [19], head pose classification [20], facial expression recognition [21], and so on. Among them, TrAdaBoost [14] is the most widely used method. In TrAdaBoost, AdaBoost [22] is applied to same-distributed training data to learn the base classifiers of the model. The weights of diff-distributed training instances, which are wrongly predicted due to the dissimilarity to the same-distributed instances, would be decreased to weaken their impacts.

However, for some difficult tasks in computer vision or image processing, simple boost stumps (e.g., one layer decision tree) may be not sufficient to model the data distribution to achieve high accuracy. Then, it attempts to apply a more complex hypothesis set, for example, decision trees with deep layers. However, in transfer learning scenario, it might not work, which is mainly because target-domain and source-domain samples are from different distributions. Directly adapting more complex base learners in TrAdaBoost, for example, the ensemble of deep layer decision trees with the same depth, may result in overfitting because it probably will overfit some diff-distributed data in the source domain. For example, gradient boosting methods [23] try to compute the optimal gradient direction to fit the training data to minimize the loss function. However, in transfer learning scenario, the optimal gradient directions of source- and target-domain data are different if their distributions are different. If we directly train a gradient tree with high depth to fit the whole data (i.e., both source- and target-domain data), the gradient tree may overfit some source-domain data with different distributions as the target-domain data.

To address the above-mentioned challenge, we propose a Deep Decision Tree Transfer Boosting (DTrBoost) approach, which is enlightened by DeepBoost [24], to improve the instance transfer learning when facing more complex data. DeepBoost proposes the learning bounds for convex ensembles of the base classifier set formulated in terms of the Rademacher complexities [25]. Like Vapnik–Chervonenkis (VC)-dimension, Rademacher complexity is a data-dependent

Manuscript received April 26, 2017; revised October 9, 2017, May 10, 2018, September 19, 2018 and February 4, 2019; accepted February 12, 2019. This work was supported in part by the NSF IIS Award under Grant 1651902 and in part by the U.S. Army Research Office Award under Grant W911NF-17-1-0367. *(Corresponding author: Shuhui Jiang.)*

S. Jiang is with the Department of Electrical and Computer Engineering, College of Engineering, Northeastern University, Boston, MA 02115 USA (e-mail: shjiang@ece.neu.edu).

H. Mao is with the Khoury College of Computer and Information Sciences, Northeastern University, Boston, MA 02115 USA (e-mail: mao.hai@husky.neu.edu).

Z. Ding is with the Department of Computer, Information and Technology, Indiana University—Purdue University Indianapolis, Indianapolis, IN 46202 USA (e-mail: zd2@iu.edu).

Y. Fu is with the Department of Electrical and Computer Engineering, College of Engineering, Northeastern University, Boston, MA 02115 USA, and also with the Khoury College of Computer and Information Sciences, Northeastern University, Boston, MA 02115 USA (e-mail: yunfu@ece.neu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2019.2901273

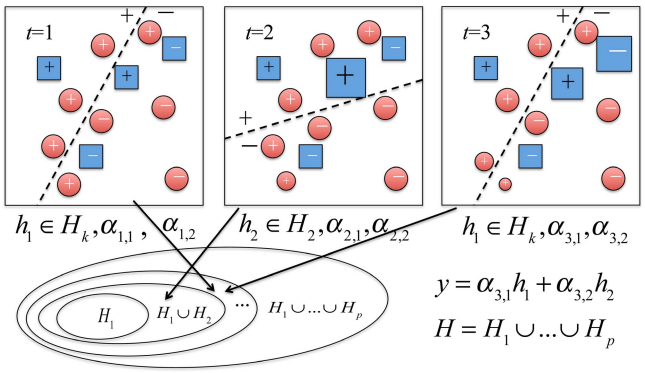


Fig. 1. Illustration of DTrBoost depicting the first three boost iterations. The scenario shows the source-domain data (circle points in red) transferring to target-domain data (square points in blue). The class label of each point is shown as (+/−). The size of the point indicates the weight of the instance. The dashed line indicates the hypothesis h_j, $j \in [1, \cdots, N]$. $\{h_1, \cdots, h_j, \cdots, h_N\}$ is a set of different hypotheses, where $N = 2$ in this example. H is composed by the union of p disjoint families $H_1, \cdots, H_p$ ordered by an increasing complexity. In this example, $h_1 \in H_k$ and $h_2 \in H_2$. DTrBoost updates the weights of all h_j denoted as $\alpha_{t,j}$ at each iteration t. After each iteration, DTrBoost increases the weights of wrongly classified target-domain data and decreases the weights of wrongly classified source-domain data.

estimate of complexity of the function class. It quantifies how much half of the samples can be representative of the other half, also quantifies the extent to which some functions in the function class could be correlated with a noise sequence. The penalty term guarantees that we can select base learners without overfitting. DTrBoost ensembles different complexity hypotheses families learning from both source-domain data and target-domain data at different iterations, and allocates more weights on hypotheses drawn from less complexity hypotheses families to avoid overfitting. We minimize the difference between the derivatives of loss functions of gradient boost in whole data and target-domain data, with the penalty on the complex term to avoid overfitting. Through different iterations during optimization, as shown in Fig. 1, DTrBoost gradually updates the weights of diff-distributed samples in the source domain and also updates the weights of all different hypotheses $\{h_1, \cdots, h_j, \cdots, h_N\}$. After all T iterations, DTrBoost ensembles the N $(0 < N \leq T)$ different hypotheses with the weights learned from $[T/2+1, T]$ as the final classifier. Finally, we summarize the contributions and novelties of our paper as follows.

1) We propose a novel DTrBoost for the transfer learning scenario. Compared with existing boosting-based transfer learning methods in terms of complexity-fixed hypotheses (e.g., fixed number of decision tree depth), DTrBoost adds the complexity penalty on the hypotheses, which helps to avoid overfitting when applying more complex hypotheses (e.g., a deeper decision tree) to deal with more complex cases.
2) We analyze our framework theoretically in terms of the convergence property. We prove that the derivative difference of gradient trees between whole data and target-domain data will be small enough after $T/2$ iterations.

II. Related Work

In this section, we first briefly describe the related works of ensemble learning methods. Second, we introduce the related works of transfer learning methods, especially the instance transfer learning ones.

A. Ensemble Learning

Ensemble learning methods (e.g., boosting, bagging, Bayesian averaging, and stacking) combine several learning algorithms to create one with higher prediction accuracy [22], [26]–[28].

Ensemble methods are widely applied, in practice, due to the significant improvement in terms of performance [29]–[32]. Ditzler *et al.* [29] developed an ensemble of online linear models using bagging and boosting for online feature selection. Li *et al.* [30] integrated L_p-norm into the bag scores to localize the witness instances for multiinstance classification. Ensemble algorithms have also been introduced in semisupervised classification (SSC) with improved generalization performance when compared to single classifiers. For example, Soares *et al.* [31] proposed a cluster-based boosting method for multiclass SSC. They integrated cluster-based regularization into boosting to avoid generating decision boundaries in high-density regions [31]. Zhang *et al.* [32] proposed a multiobjective deep belief networks ensemble method for estimating the remaining useful life.

In particular, AdaBoost, short for Adaptive Boosting is based on a rich theoretical analysis, which guarantees the performance in terms of the margins of the training samples [33], [34]. AdaBoost and its variants have been widely applied to regression and classification problems. Bjurgert *et al.* [35] used AdaBoot to estimate dynamical systems and explored the connection between AdaBoost and system identification. Qi *et al.* [36] applied ensemble learning strategy for the learning problem with label proportions (LLPs). They proposed an Adaboost-based loss function according to different weights for LLP and named as Adaboost-LLP. Adaboost-LLP exploits extra weight information and ensembles multiple weak classifiers into a strong one.

However, AdaBoost and its variants [16], [37] may face the overfitting problem with more complex data. Deep boosting (DeepBoost) decomposes a set of base classifiers into subfamilies according to, for example, the depth of the decision tree. It adopts new data-dependent learning bounds for convex ensemble expressed in terms of the Rademacher complexities of the subfamilies [24].

B. Transfer Learning

In recent years, transfer learning is widely used in neural networks and learning systems [1]–[7]. For example, Zhang *et al.* [3] applied a deep learning model with deep intersaliency mining and intrasaliency prior transfer for cosaliency detection among multiple related images. Sequence transfer learning methods have also attracted a lot of attentions due to the large amounts of text and video data from social media such as Twitter and Facebook [6].

Transfer learning approaches fall into three categories: inductive, transductive, and unsupervised transfer learning [8].

Inductive transfer learning focuses on the scenario that the target task is different from the source task, while two domains can be either the same or not. The tasks of source and target domain of the transductive transfer learning are the same while two domains are different. In unsupervised transfer learning approaches, the target-domain learning tasks are unsupervised, such as dimensionality reduction, clustering, and density estimation.

Instance transfer learning belongs to the transductive transfer learning setting. Instance transfer approaches take both source and target data into consideration during the training process [8]. TrAdaBoost [14] paved the way of instance transfer learning, which adopts AdaBoost [22] algorithm as a best-fit instance transfer learner. We will provide more detailed introduction of TrAdaBoost in Section III-B. TrAdaBoost is extended to many transfer learning tasks such as regression transfer [37] and multisource learning [17], [38]. For example, TransferBoost [17] calculates the error difference between only the target task or the integrate of the target and each source task as the aggregate transfer term for each source task. The weights of instances that belong to a positive transferable source task to the target task are boosted. Recently, Huang *et al.* [39] proposed a topic-related TrAdaBoost for cross-domain sentiment classification. They first extracted the topic distribution for each document. Then, they constructed the new representation for each document by appending the topic distribution to the original model.

A few efforts have been done to improve the performance of TrAdaBoost by modifying the weight updating strategy. For example, in cost-sensitive boosting approaches [17], a fixed cost is incorporated, via AdaCost [15], to the source weight updating strategy. We precalculated the distributions between source- and target-domain distributions with probability estimates as the cost. Recently, Ryu *et al.* [40] also applied the cost-sensitive boosting approach to predict the cross-project defect. In dynamic-TrAdaBoost, a dynamic factor was incorporated to meet the intended design of both AdaBoost and the "weighted majority algorithm" [16]. However, none of them focused on the overfitting problems when the hypotheses become more complex, for example, deeper decision trees. TrAdaBoost and the extension methods are with complexity-fixed hypotheses, which may bring overfitting problems when facing more complex data.

Recently, deep transfer learning algorithms generalize deep structures into the transfer learning scenario, in order to reduce the domain discrepancy and improve the transferability of feature representation [41]–[47]. For example, Zhou *et al.* [46] proposed a deep learning transfer hashing framework that incorporates transfer learning and hashing to address the data sparsity problem in hashing. Ding and Fu [47] investigated low-rank coding at top task-specific layers and proposed a deep transfer low-rank coding neural network framework. However, although those works can mitigate the marginal distribution divergence between the target and the source domain, they may fail to uncover the intrinsic classwise structure of two domains. Meanwhile, a huge number of labeled data are usually needed to estimate a mass of parameters in deep convolutional neural network framework.

Specifically, our method is in the line of instance transfer learning. Compared with TrAdaBoost and the extension methods with complexity-fixed hypotheses, DTrBoost adds the complexity penalty on the hypotheses. It means a hypothesis with the higher complexity (e.g., a deeper decision tree) is penalized and allocated lower weights. DTrBoost ensembles different complexity hypotheses families learning from both source-domain data and target-domain data, and allocates more weights on hypotheses from less complexity hypotheses families to avoid overfitting.

III. Proposed Method

In this section, first we give the problem formulation. Second, we introduce the preliminary knowledge: Boosting-based transfer learning (i.e., TrAdaBoost) [14] and DeepBoost [24]. Third, we describe our DTrBoost algorithm in detail. The theorem proof of the convergence property is also provided.

A. Problem Formulation

For the instance transfer learning scenario, let $X = X_s \cup X_d$ be the instance space, where X_s is the target-domain instance space and X_d is the source-domain instance space, which is with different distributions as the target domain. $Y = \{1, -1\}$ denotes the set of category labels. A concept is a Boolean function c, mapping from X to Y. The test data set is with the same distribution as the target domain, denoted as U. We partition the training data set $L \subseteq \{X \times Y\}$ into two labeled sets L_d and L_s. L_d denotes the diff-distributed training data that $L_d = \{(x_i^d, c(x_i^d))\}$, where $x_i^d \in X_d (i = 1, \cdots, n)$. L_s denotes the same-distributed training data that $L_s = \{(x_j^s, c(x_j^s))\}$, where $x_j^s \in X_s (i = 1, \cdots, m)$. $L = \{(x_i, c(x_i))\}$, where x_i is defined as

$$\begin{cases} x_i = x_i^d, & i = 1, \cdots, n \\ x_i^s, & i = n+1, \ldots, n+m. \end{cases}.$$

The problem is that, given a small number of labeled target-domain training data L_s and a large number of labeled source-domain training data L_d, we aim to learn a Boolean function c from X to Y to minimize the prediction error on the unlabeled test data U.

B. Boosting-Based Transfer Learning

In this section, we briefly describe the algorithm of TrAdaBoost [14]. TrAdaBoost paved the way of boosting-based transfer learning algorithms [15], [37], [38], [48], and it is one of the most representative methods in the transfer learning scenario [8].

The input of TrAdaBoost consists of labeled source- and target-domain training data L_d and L_s; a base "learner" such as a decision tree with one or two stumps; the maximum number of iterations T. TrAdaBoost iteratively trains a set of base classifiers based on the weighted target and source training samples. At the end of each boosting iteration, we increase

the weights of misclassified target instances and decrease the weights of misclassified source instances. The weights of correctly classified instances are unchanged. The weight of each hypothesis depends on the training error of the target domain in each iteration. Thus, the impacts of diff-distributed instances in the source domain gradually decrease. After all T iterations, TrAdaBoost ensembles the weighted hypotheses learned from $[T/2+1, T]$ as the final classifier. This updating mechanism is adapted from the AdaBoost [22] algorithm. The algorithm of TrAdaboost is given in Algorithm 1 and more detailed descriptions could be referred in [14].

C. Deep Boosting

Our work is enlightened by DeepBoost [24]. Here, we briefly introduce the basic idea and the objective function of DeepBoost.

Assume that p disjoint families $H_1, ..., H_p$ are decomposed from a set of base classifiers H, mapping from X to $\mathbb{R}$, ordered by an increasing complexity. Each family H_k, $k \in [1, p]$ can be a set of functions based on monomials of degree k or a set of decision trees of depth k.

The core of the DeepBoost is a capacity-conscious criterion for the hypotheses h_j selection. A learning bound for the convex ensembles formulated using the Rademacher complexities [25] of the subfamilies, and the weights are assigned to each subfamily. It is beneficial and guaranteed that DeepBoost seeks $\boldsymbol{\alpha}$ to minimize the learning bound, which is

$$G(\boldsymbol{\alpha}) = \frac{1}{m}\sum_{i=1}^{m} 1_{y_i \sum_{t=1}^{T} \alpha_t h_t(x_i) \le \rho} + \frac{4}{\rho}\sum_{t=1}^{T} \alpha_t r_t \tag{1}$$

where x_i denotes the target domain instance, $i \in \{1, ..., m\}$. y_i denotes the label of x_i. T denotes the number of iterations. h_t denotes the selected hypothesis at tth iteration. α_t denotes the weight assigned to the corresponding h_t. Let $\rho > 0$ while $\sum_{t=1}^{T} \alpha_t h_t(x_i) \le \rho$ is the ρ-margin error, and $r_t = \Re_m(H_{d(h_t)})$ is the standard Rademacher complexity. $d(h_t)$ is an index function that maps h_t to its hypotheses subfamily H_k. More descriptions of DeepBoost could be referred to [24].

D. Deep Decision Tree Transfer Boosting

TrAdaBoost and its variants may face overfitting problem when it comes to more complex data. In the transfer learning scenario, even worse, a more complex hypothesis set is likely to not only overfit the target-domain training samples but also the useless source-domain training samples, which are lying in different distributions. In order to solve this problem, we propose a DTrBoost algorithm.

We first introduce the hypotheses generation using decision tree. It is treated as a process of parameter estimation. We mainly follow the idea of the gradient boost tree to learn the parameters of hypotheses [23]. The objective function of gradient boost [23] of both the target domain and the source domain is as

$$\min_{\alpha_t \ge 0} \frac{1}{n+m}\sum_{i=1}^{n+m} \Phi(1 - y_i \sum_{t=1}^{T} \alpha_t h_t(x_i, a_t)) \tag{2}$$

Algorithm 1 TrAdaBoost [14]

INPUT: The labeled data set L_d, L_s. A base learning algorithm **learner**. The maximum number of iterations T.

1: Initialize the source domain sample and target domain sample weight $\boldsymbol{w}^1$.
2: **for** t=1,2,...,T **do**
3: Set $\boldsymbol{p}^t = \boldsymbol{w}^t / (\sum_{i=1}^{n+m} w_i^t)$.
4: Call **learner** and get hypothesis $h_t : X \to Y$ based on both source domain samples and target domain samples.
5: Calculate the error ε_t of h_t on target domain data.
6: Set $\beta_t = \varepsilon_t/(1-\varepsilon_t)$ and $\beta = 1/(1+2\ln n/N^{(1/2)})$. Note that, ε_t is required to be less than 1/2.
7: Update the new weight vector:

$$\begin{cases} w_i^{t+1} = w_i^t \beta^{|h_t(x_i)-c(x_i)|}, & 1 \le i \le n. \\ w_i^t \beta_t^{-|h_t(x_i)-c(x_i)|}, & n+1 \le i \le n+m. \end{cases}$$

8: **end for**

ONPUT: the hypothesis

$$\begin{cases} h_f(x) = 1, & \prod_{t=\lceil T/2 \rceil}^{T} \beta_t^{-h_t(x)} \ge \prod_{t=\lceil T/2 \rceil}^{T} \beta_t^{-1/2} \\ 0, & \text{otherwise.} \end{cases}$$

where x_i denotes the input instance and $i \in \{1, \cdots, n, \cdots, n+m\}$. y_i denotes the label of x_i. $t \in \{1, 2, \cdots, T\}$ denotes the index of iteration. T denotes the total number of iterations. $h_t(x_i, a_t)$ denotes the hypothesis in the iteration t, where a_t is the model parameters of h_t. α_t denotes the weight of h_t. Let Φ be a nonincreasing convex function. For example, in AdaBoost, Φ is selected as the exponential function.

In tth iteration, gradient boosting chooses a new function $h_t(x_i, a_t)$ to be the most parallel to the negative gradient $\{d_t^w(x_i)\}_{i=1}^{m+n}$ along the observed data

$$d_t^w(x) = \left[\frac{\partial \Phi(y, f(x))}{\partial f(x)}\right]_{f(x_i)=f_{t-1}(x_i)} \tag{3}$$

where $f_t \leftarrow f_{t-1} + \alpha_t h_t(x, a_t)$. Here, d_t^w denotes the gradient calculated with both source and target domains. w notifies whole data including the source domain and the target domain.

However, as discussed above, in the source domain, some data are useless and with different distributions as the target domain. Thus, the gradient directions are different between whole data and data in the target domain. We could not directly apply the hypotheses learned in the whole data for target-domain classification.

Our main idea is to decrease the weights of the diff-distributed samples in the source domain, so that the gradient direction would become very close when the weighted data in two domains are from the same distribution. Meanwhile, we consider the complexity penalty enlightened by DeepBoost to avoid overfitting.

Suppose $H = \{h_1, \cdots, h_j, \cdots, h_N\}$ are N different hypotheses. e_j is the direction of the jth hypothesis. F_w is the loss function of gradient boosting of data in both the source

and the target domain, and F_s is the loss function of gradient boosting of data in the target domain as

$$F_w^t = \frac{1}{n+m}\sum_{i=1}^{n+m}\Phi(1-y_i\sum_{j=1}^{N}\alpha_{t,j}h_j(x_i,a_j)) + \sum_{j=1}^{N}(\lambda r_t+\beta)\alpha_{t,j} \quad (4)$$

$$F_s^t = \frac{1}{m}\sum_{i=1}^{m}\Phi(1-y_i\sum_{j=1}^{N}\alpha_{t,j}h_j(x_i,a_j)) + \sum_{j=1}^{N}(\lambda r_t+\beta)\alpha_{t,j} \quad (5)$$

where $\alpha_{t,j}$ denotes the weight of hypothesis j in the tth iteration. The second part in (4) and (5) is the regularization of the hypothesis Rademacher complexity, where λ and β are the parameters. $r_t = \Re_m(H_{d(h_t)})$ is the standard Rademacher complexity. This term is introduced in DeepBoost but not for transfer learning scenario [24]. $d(h_t)$ denotes the index of the hypothesis set which h_t belong to, and that is $h_t \in H_{d(h_t)}$.

Our objective function is to minimize the derivative of F_w^t and F_s^t in direction e_j is as

$$\min_{\alpha_{t,j}} G = \min_{\alpha_{t,j}} \left| F_w^{'t}(\alpha_{t-1,j}, e_j) - F_s^{'t}(\alpha_{t-1,j}, e_j) \right| \quad (6)$$

where $F_w^{'t}(\alpha_{t-1,j}, e_j)$ and $F_s^{'t}(\alpha_{t-1,j}, e_j)$ are the derivative of F_w^t and F_s^t in direction e_j, respectively. By minimizing the derivative of F_w^t and F_s^t, the weights of the diff-distributed samples in the source domain would be deceased at each iteration. Thus, the distributions of the weighted samples in source and target domain gradually get similar to each other. When learning the weights of hypotheses, by adding the complexity penalty, the objective function can punish more of complex hypotheses. Thus, the hypotheses families with less complexity would be allocated more weights to avoid overfitting.

E. Optimization

Our optimization mainly consists of three parts. The first is to learn a series of hypotheses with different complexities (e.g., decision trees with different depths) on both target-domain data and source-domain data. The second is to select the local best hypothesis from different complexity hypotheses families only based on target-domain data, and update the weights of the best hypothesis set. The third is to update weights of both source and target domains.

1) Learning Hypothesis: We apply both n source-domain data L_d and m target-domain data L_s to learn hypothesis as (4). We take the derivative of (4) in direction e_j in iteration t toward all the weighted samples according to [24]

$$\begin{aligned} d_{t,j}^w &= F_w^{'t}(\alpha_{t-1,j}, e_j) \\ &= -\frac{1}{n+m}\sum_{i=1}^{n+m} y_i h_j(x_i)\Phi'(1-y_i f_{t-1}(x_i)) + \alpha_{t-1,j}\Lambda_t^w \\ &= -\frac{1}{n+m}\sum_{i=1}^{n+m} y_i h_j(x_i) D_t^w(i) S_t^w + \alpha_{t-1,j}\Lambda_t^w \\ &= \left(2\epsilon_{t,j}^w - 1\right)\frac{S_t^w}{n+m} + \alpha_{t-1,j}\Lambda_t^w \end{aligned} \quad (7)$$

where $d_{t,j}^w$ is the derivative of F_w toward direction e_j in the tth iteration. Λ_t is calculated as $\Lambda_t = \lambda r_t + \beta$. $\lambda r_t + \beta$ is the complexity penalty in (4). S_t^w is the normalizer of the weights of L_s and L_d in iteration t [24]. $D_t^w(i)$ denotes the sample weight of each x_i. $\epsilon_{t,j}^w$ denotes the estimated error.

After calculating the gradient $d_{t,j}^w$, gradient boost tree chooses the new function $h_j(x_i, a_j)$ incrementally to be the most correlated with $-d_{t,j}^w$

$$a_j = \arg\min_{a_j} \sum_{i=1}^{n+m}\left[-d_j^w(x_i) - h_j(x_i, a_j)\right]^2. \quad (8)$$

Thus, we obtain the hypothesis set $H = \{h_1, \cdots, h_j, \cdots, h_N\}$ after t-iterations, $N \le t \le T$.

2) Search and Update Local Best Hypothesis: In order to minimize the derivative of F_w and F_s, after getting $H = \{h_1, \cdots, h_j, \cdots, h_N\}$, we find and update the local best hypothesis h_k among H in each iteration.

First, we calculate the derivative of F_s in each direction e_j of h_j toward weighted target-domain samples with the same strategy as (7) as

$$\begin{aligned} d_{t,j}^s &= F_s^{'t}(\alpha_{t-1,j}, e_j) \\ &= -\frac{1}{m}\sum_{i=1}^{m} y_i h_j(x_i)\Phi'(1-y_i f_{t-1}(x_i)) + \alpha_{t-1,j}\Lambda_t^s \\ &= -\frac{1}{m}\sum_{i=1}^{m} y_i h_j(x_i) D_t^s(i) S_t^s + \alpha_{t-1,j}\Lambda_t^s \\ &= \left(2\epsilon_{t,j}^s - 1\right)\frac{S_t^s}{m} + \alpha_{t-1,j}\Lambda_t^s \end{aligned} \quad (9)$$

where $d_{t,j}^s$ is the derivative of F_s toward direction e_j in the tth iteration. Similar as (7), S_t^s is the normalizer of the weights of L_s in iteration t. $D_t^s(i)$ denotes the sample weight. $\epsilon_{t,j}^s$ denotes the estimated error. λ is the complexity penalty.

Then, we choose h_j with the largest $d_{t,j}^s$ as the local best learner, denoted as h_k. In this way, we find the direction e_j that is most parallel to the gradients of F_s of the weighted target-domain samples.

The basic idea of updating the weights of hypothesis is that, in each iteration t, we only update the weight $\alpha_{t,k}$ of local best hypothesis h_k and keeps other $\alpha_{t,j}$ $(j \ne k)$ unchanged. This idea could be formalized as $\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} + \eta_t e_k$, where $\boldsymbol{\alpha}_t$ is the set of $\alpha_{t,j}$, $\boldsymbol{\alpha}_t = \{\alpha_{1,t}, ..., \alpha_{N,t}\}$. e_k denotes that direction of h_k. η_t could be calculated as [24].

3) Update Weights of Training Samples: We introduce the strategy of updating the weights of both target-domain training samples L_s and source-domain training samples L_d. Target-domain training data L_s are updated according to the derivative of target-domain data [24]

$$D_{t+1}^s(i) \leftarrow \frac{\Phi'(1-y_i\sum_{j=1}^{N}\alpha_{t,j}h_j(x_i))}{S_{t+1}^s}, \quad n+1 \le i \le n+m. \quad (10)$$

The source-domain training data L_d are updated as

$$D_{t+1}^d(i) \leftarrow D_t^d(i)\gamma^{|(\sum_{j=1}^N \alpha_{t,j}h_j(x_i))-c(x_i)|}, \quad 1 \le i \le n \quad (11)$$

where $\gamma = \exp[-0.5 * \log(1 + 2\ln n/T^{(1/2)})]$ and $c(x_i)$ is the class label of x_i [14].

4) Ensemble Local Best Hypotheses: After T iterations, DTrBoost ensembles all the N local best hypotheses together as $f = \sum_{j=1}^N (\alpha_{T,j} - \alpha_{T/2,j})h_j$. f is the final output of the algorithm. The weight of the tth hypothesis is $(\alpha_{T,j} - \alpha_{T/2,j})$, meaning we only consider the weight $\alpha_{t,j}$, $t \in [2/T + 1, T]$. We exclude the weights learned from the first $T/2$ iterations. Because in the first certain number of iterations (i.e., $T/2$), the weights of diff-distributed data in the source domain have not been decreased to a very small number. It will largely hold back learning the same gradient direction comparing to the gradient direction only learning from the target domain. In the theoretical analysis, we will prove that after $T/2$ iterations, the differences of the gradient direction in the whole data and the target-domain data would be a very small number.

Algorithm 2 DTrBoost

INPUT:
(1) Source and target domain data L_d, L_s,
(2) Maximum depth of decision tree,
(3) Maximum number of iterations T,

1: Initialize the weights of source domain samples D_1^d, and target domain samples D_1^s.
2: **for** $t = 1, 2, \cdots, T$ **do**
3: Calculate the derivative $d_{t,j}^w$ as Eq. (7) and learn h_t.
4: Calculate the number of different hypotheses N.
5: **for** $j = 1, 2, \cdots, N$ **do**
6: Calculate the derivate $d_{t,j}^s$ of Eq. (9).
7: **end for**
8: $k = \arg\max |d_j^s|_{j\in[1,N]}$
9: Update $\boldsymbol{\alpha}_t$ by $\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} + \eta_t \boldsymbol{e}_k$.
10: Update target domain data weight D_{t+1}^s as Eq. (10).
11: Update source domain data weight D_{t+1}^d as Eq. (11).
12: **end for**

ONPUT: $f = \sum_{j=1}^N (\alpha_{T,j} - \alpha_{T/2,j})h_j$

Algorithm 2 presents the algorithm of DTrBoost. The inputs of the algorithm are the labeled source-domain (diff-distributed) training data L_d and labeled target-domain (same-distributed) training data L_s. The output of DTrBoost is $f = \sum_{j=1}^N (\alpha_{T,j} - \alpha_{T/2,j})h_j$, where N is the number of different hypotheses, $N \le T$.

5) Discussion: Here, we analyze the relationship between our DTrBoost and TrAdaBoost. When $\lambda = 0$ and $\beta = 0$, the complexity term of the bound of DTrBoost and the control of the sum of the mixture weights are ignored. It coincides with TrAdaBoost with precisely the same direction.

IV. Theoretical Analysis of DTrBoost

In this section, we theoretically analyze our framework in terms of the convergence property. We prove that after several iterations, the difference between $F_w^{'t}$ and $F_s^{'t}$ in (6) is within a certain small number.

Lemma1: In the source domain $\forall\epsilon$, $\exists T_0$, $t \ge T_0$, $S_t^d \le \exp(-(1-\gamma)\iota)$, where ι is a small number and convergent after ($t \ge T_0$). We follow [14] and set T_0 as $T/2$. γ is a weight updated root and $\gamma = \exp(-0.5 \times \log(1 + (2\ln n/T)^{(1/2)})$.

Theorem2: $\exists T_0, t \ge T_0, |F_w^{'t}(\alpha_{t-1,j}, e_j) - F_s^{'t}(\alpha_{t-1,j}, e_j)| \le (2\epsilon_{t,j}^s)(S_t^d/m) \le (2\epsilon_{t,j}^s)(\exp(-(1-\gamma)\iota)/m)$, where $F_w^{'t}(\alpha_{t-1,j}, e_j)$ is the derivative of the loss function toward the jth direction on whole data in the tth iteration, and $F_s^{'t}(\alpha_{t-1,j}, e_j)$ is the derivative of the loss function toward the jth direction on target-domain data in the tth iteration. $(2\epsilon_{t,j}^s)(\exp(-(1-\gamma)\iota)/m)$ is a certain small number.

Theorem 2 substantiates that after $T/2$ iterations, the derivative of the loss function between whole data and target-domain data toward the same direction is within a certain small value. As described above, the difference of the derivative is mainly caused by the difference of the distribution of source and target-domain data. During the optimization, the weights of the diff-distributed source-domain data are gradually decreasing and loosing impact. After $T/2$ iterations, the difference of the derivative is within a certain small number. It shows that DTrBoost is able to decrease the impact of diff-distributed source-domain data, and augment the knowledge learning from same-distributed source-domain data.

The proof of Lemma 1 and Theorem 2 could be found in the Appendix.

V. Experiment

In this section, first at all, we introduce the data set and experimental settings. Second, we present the experimental results of both our method and the state-of-the-art methods. Third, we discuss the impact of the depth of the decision tree and the overfitting problem.

A. Data Sets

To evaluate the effectiveness of DTrBoost, we conduct the experiments on seven data sets, including three image data sets and four UCI data sets,[1] following the data sets used by our comparison methods. Table I summarizes the description of seven data sets.

Office+Caltech and CMU PIE are the two benchmark transfer learning data sets. Office+Caltech has four domains: Caltech-256 (C), Amazon (A), Webcam (W), and digital single-lens reflex camera (DSLR) (D). It is released in [49]. The office data set contains 4,652 images, 31 categories and three domains, Amazon, Webcam, and DSLR. Caltech-256 contains 30 607 images and 256 categories. It is a standard object recognition database. For Office+Caltech data set, we apply two kinds of feature representations to conduct the experiments: speeded-up robust features (SURF) and convolution neural network (CNN). For better comparison with previously reported performance, we first use the SURF descriptors provided in [49]. SURF descriptors are encoded with the codebook which is from a subset of Amazon

[1] https://archive.ics.uci.edu/ml/data_sets.html

TABLE I
DATA SET DESCRIPTION. WE SHOW THE TYPE OF THE TASK (TYPE), NUMBER OF SAMPLES (#SAMPLE), FEATURE DESCRIPTOR (DESCRIPTOR), FEATURE DIMENSIONS (#FEATURE) OF EACH DATA SET

Dataset	Type	#Sample	Descriptor	#Feature
Office +Caltech	Object	2,533	SURF	800
			CNN	4,096
PIE	Face	11,554	Raw pixel	1,024
VLCSI	Object	18,070	CNN	4,096
Mushroom	Life	8,416	Attributes	22
OCR	Computer	5,620	Attributes	64
Ionosphere	Physical	351	Attributes	34
BreastCancer	Life	699	Attributes	10

images. The histograms are with 800-bin and standardized by z-score. Following [50], we also apply DeCaf[6] features. DeCaf is a convolutional network framework which is trained on imageNet [51], [52]. DeCaf[6] features are the activations of the sixth fully connected layer of the network and with 4096-dimension features.

PIE, which stands for "Pose, Illumination, Expression," is a benchmark face database. Each subset of PIE is with different poses and we follow [53] and choose PIE05 (left pose), PIE07 (upward pose), PIE09 (downward pose), PIE27 (frontal pose), and PIE29 (right pose) as five domains in the experiments. In each domain, face images are taken with different illuminations and expression conditions. In CMU PIE, the faces are cropped into the size of 32×32 and adopt the gray-scale raw pixel value as the input, which leads to 1024-dimension features.

In the experiments, we pick one domain as the target domain, the others as the source domain. In CUM PIE, the PIE05 means we take PIE05 as the target domain and combine the rest as the source domain. As we deal with the binary classification problem, we average the pairwise binary classification accuracy.

VLCSI [54], [55] consists of 18 070 images from PASCAL VOC2007 (V), LabelMe (L), Caltech-101 (C), SUN09 (S), and ImageNet (I) data sets, each of which represents one domain. C and I are the object-centric data sets, while V, L, and S are scene-centric. The five domains share five object categories: "bird," "car," "chair," "dog," and "person."

For four UCI data sets, we apply the original attributes of the data as feature representation. As these four UCI data sets are not originally used for transfer learning purpose, we follow TrAdaboost [14] to use KL-divergence (Kullback and Leibler, 1951) as the criterion to separate the data set into two domains of different distributions. For example, for Mushroom data set, we split the data set based on the feature "stalk-shape" following TrAdaboost. Table II summarizes the KL-divergence and size of source-domain training data L_d, and target-domain training and test data $L_s \cup U$.

TABLE II
DESCRIPTION OF UCI DATA SETS MUSHROOM, OCR17, OCR49, IONOSPHERE, AND BREAST CANCER. FOR EACH DATA SET, WE PRESENT THE KL-DIVERGENCE AND THE SIZE OF SOURCE-DOMAIN TRAINING DATA L_d, AND TARGET-DOMAIN TRAINING AND TEST DATA $L_s \cup U$

Dataset	KL-divergence	Size	
		$\lvert L_d\rvert$	$\lvert L_s \cup U\rvert$
Mushroom	0.5086	4,864	3,552
OCR17	23.01	776	358
OCR49	24.88	362	769
Ionosphere	16.36	204	147
BreastCancer	5.808	292	407

B. Compared Methods

We compare our method with three state-of-the-art instance transfer learning methods: boosting for transfer learning (TrAdaBoost), cost-sensitive boosting (Cost-TrAdaBoost), adaptive boosting for transfer learning using dynamic updates (Dynamic-TrAdaBoost). Although there are other state-of-the-art related works [4], [37]–[39], they are not working on the same scenario as ours.

In order to show the effectiveness of transfer learning, we also compare boosting based methods but without transfer learning: AdaBoost and DeepAdaBoost (i.e., the AdaBoost setting for DeepBoost). For transfer learning methods, both the target- and source-domain training data are used. For these two nontransfer learning methods, only target-domain training data are used.

The brief introduction of five methods is listed as follows.

1) *TrAdaBoost [14]:* TrAdaBoost is the most popular boosting-based transfer learning algorithm, which extends boosting-based learning algorithms [22] to the transfer learning scenario. The description of TrAdaBoost is presented in Section III-B "boosting-based transfer learning."
2) *Cost-TrAdaBoost [15]:* Ashok *et al.* extended the cost-sensitive boosting method [48] to transfer learning scenario for concept drift. In Cost-TrAdaBoost, a fixed cost is incorporated into the source-domain weight updating strategy. This cost is precalculated according to the relevance between source and target distributions by probability estimates. Recently, Ryu *et al* [40] also applied the cost-sensitive boosting approach for cross-project defect prediction.
3) *Dynamic-TrAdaBoost [16]:* Al-Stouhi and Reddy [16] extended TrAdaBoost by incorporating a dynamic factor in order to meet the intended design of both AdaBoost and the "weighted majority algorithm."
4) *AdaBoost [22]:* AdaBoost ensembles a base classifier hypothesis set. In each iteration, Adaboost increases the weight of wrongly classified samples and decreases the weight for correctly classified samples.
5) *DeepAdaBoost [24]:* DeepBoost investigates a capacity-conscious criterion for the hypotheses selection. A complexity penalty is added to prevent overfitting. In this paper, we adopt the exponential function-based DeepBoost and name it as DeepAdaBoost.

TABLE III

ERROR RATES UNDER OFFICE+CALTECH-256 DATA SET USING SURF DESCRIPTORS AND CMU PIE DATA SET USING THE RAW PIXEL DESCRIPTOR UNDER THE RATIO OF 0.05. H_1 AND H_2 DENOTE H_1^{stumps} AND H_2^{stumps}, RESPECTIVELY. THE BEST AND SECOND BEST PERFORMANCES ARE WITH BOLD FONTS AND UNDERLINE, RESPECTIVELY

Dataset	AdaBoost		Deep AdaBoost	TrAdaBoost		Cost-TrAdaBoost		Dynamic-TrAdaBoost		Ours
	H_1	H_2		H_1	H_2	H_1	H_2	H_1	H_2	
C,D,W->A	0.1824	0.2340	0.1622	0.1482	0.1527	<u>0.1436</u>	0.1448	0.1743	0.2100	**0.1366**
A,D,W->C	0.3200	0.3462	0.3143	0.3123	<u>0.3018</u>	0.3123	<u>0.3018</u>	0.3123	0.3123	**0.2942**
A,C,W->D	0.2431	0.2333	0.2235	0.2152	0.2064	0.2011	<u>0.1972</u>	0.2023	0.1988	**0.1924**
A,C,D->W	0.2846	0.2982	0.2755	0.1850	<u>0.1750</u>	0.1850	<u>0.1750</u>	0.1850	<u>0.1750</u>	**0.1635**
PIE05	0.2875	0.3214	0.3040	0.1279	0.1326	<u>0.1135</u>	0.1287	0.1224	0.1326	**0.1048**
PIE27	0.2832	0.2945	0.2384	0.1357	0.1542	0.1224	0.1233	0.1246	<u>0.1183</u>	**0.0781**

TABLE IV

ERROR RATES UNDER MUSHROOM, OCR17&19, IONOSPHERE, AND BREAST CANCER DATA SETS UNDER THE RATIO OF 0.1. H_1 AND H_2 DENOTE H_1^{stumps} AND H_2^{stumps}, RESPECTIVELY. THE BEST AND SECOND BEST PERFORMANCES ARE SHOWN WITH BOLD FONTS AND UNDERLINE, RESPECTIVELY

Dataset	AdaBoost		Deep AdaBoost	TrAdaBoost		Cost-TrAdaBoost		Dynamic-TrAdaBoost		Ours
	H_1	H_2		H_1	H_2	H_1	H_2	H_1	H_2	
Mushroom	0.0071	0.0071	0.0091	0.0044	0.0087	0.0044	0.0076	0.0049	<u>0.0041</u>	**0.0037**
OCR17	0.0513	0.0613	0.0267	0.0247	0.0253	<u>0.0238</u>	0.0267	0.0247	0.0453	**0.0187**
OCR49	0.0642	0.0636	0.0451	<u>0.0447</u>	0.0451	<u>0.0447</u>	0.0451	<u>0.0447</u>	0.0562	**0.0327**
Ionosphere	0.3912	0.2800	0.3520	0.2400	0.3520	0.2268	0.2132	**0.1925**	0.2631	<u>0.2075</u>
BreastCancer	0.1476	0.1360	0.1148	<u>0.1060</u>	0.1152	<u>0.1060</u>	0.1246	0.1100	0.1589	**0.0960**

TABLE V

ERROR RATES UNDER OFFICE+CALTECH-256 DATA SET USING CNN DESCRIPTORS UNDER THE RATIO OF 0.05. H_1 AND H_2 DENOTE H_1^{stumps} AND H_2^{stumps}, RESPECTIVELY. THE BEST AND SECOND BEST PERFORMANCES ARE SHOWN WITH BOLD FONTS AND UNDERLINE, RESPECTIVELY

Dataset	AdaBoost		Deep AdaBoost	TrAdaBoost		Cost-TrAdaBoost		Dynamic-TrAdaBoost		DAN	Ours
	H_1	H_2		H_1	H_2	H_1	H_2	H_1	H_2		
C,D,W->A	0.0724	0.0711	0.0545	0.0614	0.0526	0.0614	0.0614	0.0614	0.0526	<u>0.0441</u>	**0.0418**
A,D,W->C	0.1558	0.1632	0.1346	0.0847	0.0831	0.0851	0.0847	0.0847	0.0845	<u>0.0823</u>	**0.0786**
A,C,W->D	0.0421	0.0313	0.0316	0.0286	0.0274	0.0254	0.0241	0.0245	0.0245	<u>0.0223</u>	**0.0165**
A,C,D->W	0.0331	0.0331	0.0274	0.0258	0.0234	<u>0.0232</u>	<u>0.0232</u>	0.0252	0.0243	0.0252	**0.0192**

6) *DTrAdaBoost(Ours):* This method contains the full pipeline of our method. We adopt the exponential function-based DTrBoost and name it as DeepTrAdaBoost.

C. Experimental Settings

We use decision tree as the basic learner for all the methods. As AdaBoost, TrAdaBoost, Cost-TrAdaBoost, and Dynamic-TrAdaBoost are with the fixed depth of decision tree, we follow the depth setting (i.e., 1 or 2 depth) in this paper for better comparison, named H_1^{stumps} and H_2^{stumps}. Note that in existing instance transfer learning approaches, deep decision tree is not considered.

To fairly compare our method (depth adaptive) and existing instance transfer learning methods (depth fixed), in Tables III and IV, we set the maximum depth as 2. We aim at evaluating whether our method could adaptively select the depth between 1 and 2, and outperform existing methods under their depth settings. Then, in Section V-E, we discuss each method under deeper layer settings and show the effectiveness of our method toward the overfitting problem.

We set the number of iterations as 100 and the loss of boosting with the exponential function for all the methods. We test the ratio of the number of target- and source-domain training data from 0.01 to 0.5 as [14]. We conduct the experiments under each setting for 10 times and report and averaged error rates.

D. Experimental Result

We compare our method with both nontransfer learning methods, AdaBoost and DeepAdaBoost, and instance transfer learning methods, TrAdaBoost, Cost-TrAdaBoost, and Dynamic-TrAdaBoost. In Section V-D1, we show results based on low-level feature under Office+Caltech-256 and CMU PIE data sets given in Table III and under Mushroom, OCR17&19, Ionosphere, and Breast Cancer data sets given in Table IV. In Section V-D2, we show the results based on deep learning-based feature. We show the results based on

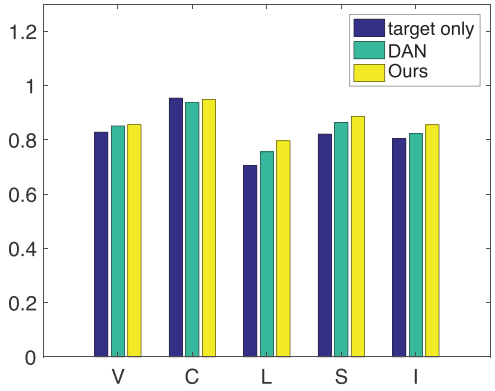


Fig. 2. Accuracy of classification results by three methods "target only," "DAN [44]," and "ours." "Target only" means we only apply the target-domain data. The x-axis presents the name of the target domain. For example, "V" presents that "V" is the target domain. The rest "C," "L," "S," and "I" are combined as the source domain. The y-axis presents the accuracy of each method.

TABLE VI

ERROR RATES UNDER THE DEPTH OF DECISION TREE FROM 1 TO 5 IN PIE05 DATA SET AT THE RATIO OF 0.05

Depth	1	2	3	4	5
DT	0.4052	0.3612	0.3892	0.4038	0.4037
AdaBoost	0.2875	0.3214	0.3214	0.3216	0.3216
DeepAdaBoost	0.3045	0.3040	0.3040	0.3035	0.3035
TrAdaBoost	0.1279	0.1326	0.1336	0.1340	0.1345
Cost	**0.1135**	0.1287	0.1283	0.1293	0.1290
Dynamic	0.1224	0.1326	0.1324	0.1326	0.1328
Ours	0.1228	**0.1048**	**0.1045**	**0.1043**	**0.1042**

CNN feature under Office+Caltech-256 given in Table IV and under VLCSI data set [54], [55] in Fig. 2. We further compare with one typical deep transfer learning method, i.e., deep adaptation networks (DAN) [44] in Office+Caltech-256 and VLCSI data set.

For four nondeep methods, AdaBoost, TrAdaBoost, Cost-TrAdaBoost, and Dynamic-TrAdaBoost, we present the results under the depth of decision tree as 1 (H_1^{stumps}) and 2 (H_1^{stumps}). As discussed in experimental settings, in order to fairly compare the performance of deep and nondeep methods, we set the maximum depth of decision tree of DeepAdaBoost and our method as 2.

The ratio of the number of training samples in the target and the source domain given in Tables III and V and Fig. 2 is 0.05 and in Table IV is 0.1. λ and β in (4) and (5) are set in the range of $\lambda \in \{10^{-i} : i = 0, \cdots, 7\}$ and $\beta \in \{2^{-i} : i = 0, \cdots, 13\}$ according to cross validation.

1) Low-Level Feature-Based Results: Tables III and IV show the error rate of each method using low-level or raw feature descriptors. First, we could see that transfer learning methods achieve a much higher performance than nontransfer learning methods. In all the data sets, TrAdaBoost gets the lower error rate than AdaBoost, and DTrAdaBoost (ours) gets lower error rate than DeepAdaBoost. It shows the effectiveness of transfer learning methods compared to nontransfer learning methods with limited target-domain training data.

Second, when comparing DTrAdaBoost (ours) with other three instance transfer learning methods, in Table III, DTrAdaBoost achieves the lowest error rate on all data sets of both H_1^{stumps} and H_2^{stumps}. In Table IV, DTrAdaBoost achieves the lowest error rate on all the settings, except for Inonosphere data set where our performance achieves the second best, which 1.5% higher error rate than Dynamic-TrAdaBoost with base hypotheses H_2^{stumps}. We analyze that it is mainly because the overfitting problem in the transfer scenario in Ionosphere is not serious under H_2^{stumps}. We could see that in three transfer learning methods TrAdaBoost, Cost-TrAdaBoost, and Dynamic-TrAdaBoost, the error rates under H_2^{stumps} are all lower than H_1^{stumps}. Thus, the DTrAdaBoost does not show significant advantages on preventing the overfitting in this setting. Furthermore, the Dynamic-TrAdaBoost does show effective performance under Ionosphere data set.

We could also observe in Tables III and IV that H_2^{stumps} does not necessarily achieve the lower error rate than H_1^{stumps}, which demonstrates that with fixed depth, a deeper hypothesis may perform worse than a lower hypothesis, which may be caused by overfitting. However, as DeepAdaBoost and DTrAdaBoost (ours) could adaptively adjust the depth of hypotheses while training, DeepAdaBoost achieves better performances than AdaBoost and our method achieves better performances than other three boosting-based transfer learning methods of both H_1^{stumps} and H_2^{stumps}.

2) Deep Learning-Based Results: We also conduct experiments based on the deep learning feature descriptor. We adopt the CNN DeCAF[6] features as inputs, with the representations' dimensionality of 4096. We further compare with one typical deep transfer learning method, i.e., DAN [44] in Office+Caltech-256 and VLCSI data set [54], [55], and given in Table V and Fig. 2. Since DAN is an 560 end-to-end deep model, we directly train on the raw images.

Table V shows the error rate of nontransfer learning methods, AdaBoost and DeepAdaBoost, and instance-based transfer learning methods, TrAdaBoost, Cost-TrAdaBoost, and Dynamic-TrAdaBoost, ours using CNN feature descriptor, and deep transfer learning model DAN. First, the results show that transfer learning-based models (TrAdaBoost, Cost-TrAdaBoost, Dynamic-TrAdaBoost, DAN, and ours) achieve lower error rates than nontransfer learning-based methods (AdaBoost and DeepAdaBoost). Second, the results demonstrate that our method achieves the lowest error rate among all the instance transfer learning methods. These two observations are the same as in Tables III and IV, which are based on low-level-based feature descriptors. Third, when comparing with DAN, our method outperforms DAN over all the four scenarios, which further shows the effectiveness of our method. Fourth, when comparing results given in Tables III and V, CNN feature-based results outperform SURF based results, which show the effectiveness of applying deep learning-based features.

Fig. 2 shows the accuracy of classification results by "target only," "DAN [44]," and "Ours." Specifically, "target only" means we only apply labeled target-domain data to recognize the test target data. The x-axis presents the name of the target domain, and the rest four domains are combined as the source

TABLE VII

ERROR RATES UNDER THE DEPTH OF DECISION TREE FROM 1 TO 5 IN OFFICE+CALTECH-256 DATA SET A,D,W→C SCENARIO USING CNN FEATURES AT THE RATIO OF 0.05

Depth	1	2	3	4	5
DT	0.4324	0.4168	0.4264	0.4278	0.4276
AdaBoost	0.3140	0.3140	0.3145	0.3267	0.3215
DeepAdaBoost	0.3154	0.3096	0.3075	0.3063	0.3065
TrAdaBoost	0.3060	0.3011	0.3068	0.3116	0.3115
Cost	0.3012	0.3012	0.3022	0.3107	0.3108
Dynamic	0.3042	0.3052	0.3050	0.3068	0.3066
Ours	**0.2984**	**0.2832**	**0.2829**	**0.2827**	**0.2827**

TABLE VIII

ERROR RATES UNDER THE DEPTH OF DECISION TREE FROM 1 TO 5 IN MUSHROOM DATA SET AT THE RATIO OF 0.1

Depth	1	2	3	4	5
DT	0.2059	0.0900	0.0423	0.0294	0.0313
AdaBoost	0.0071	0.0071	0.0061	0.0064	0.0064
TrAdaBoost	0.0091	0.0044	0.0048	0.0055	0.0770
Cost	0.0087	0.0044	0.0050	0.0057	0.0067
Dynamic	0.0076	0.0049	0.0049	0.0055	0.0063
DeepAdaBoost	**0.0044**	0.0041	0.0054	0.0059	0.0042
Ours	0.0072	**0.0037**	**0.0037**	**0.0037**	**0.0034**

domain. We average the pairwise binary classification accuracy as the results and show them in Fig. 2. We observe from the results that our model could beat DAN for all the cases, which demonstrate the effectiveness of our model. The observation also demonstrates that an end-to-end deep model is not always a good strategy for transfer learning, since a pretrained deep model already conducts some transferability.

E. Discussion of Depth and Overfitting

In this section, we discuss the performance under different depth settings and the overfitting problem. First, we show the performance of different methods under depth from 1 to 5 to demonstrate the overfitting problem of depth-fixed methods (i.e., TrAdaBoost, Cost-TrAdaBoost, and Dynamic-TrAdaBoost), and the advantage that our method could avoid overfitting. Second, we present the average depth (avg. depth) and average number (avg. no) of the final hypotheses after 100 iterations.

Tables VI–VIII present the error rates of all comparisons under PIE05, Office+Caltech-256, and Mushroom data sets, under the settings that "Depth" equals 1–5. For depth-fixed methods (i.e., AdaBoost, TrAdaBoost, Cost-TrAdaBoost, and Dynamic-TrAdaBoost), "Depth" means the layers of the decision tree. For depth-adaptive methods (i.e., DeepAdaBoost and DTrBoost), "Depth" means the maximum layers of the decision tree. Note that the actual depth of depth-adaptive methods is learned adaptively, and usually smaller than the maximum number.

We could see that the error rate of the depth-fixed methods increases when the "Depth" increases. It shows that when directly using a more complex basic learner (i.e., a deeper layer decision tree) to fit the training data, the model is easily getting overfitting. It is corresponding to our motivation to solve the overfitting problem when using a deeper decision tree in the transfer learning scenario.

However, for depth-adaptive methods (i.e., DeepAdaBoost and Ours), when the "Depth" increases, the error rate keeps decreasing, meaning facing a less serious overfitting problem. We analyze that it is because the model could penalize the complexity of the decision tree in the objective function. Thus, the actual layers of the decision tree are learned adaptively to avoid the model to be too complex and overfitting.

TABLE IX

AVERAGE DEPTH AND AVERAGE NUMBER OF THE FINAL HYPOTHESES AFTER 100 ITERATIONS

	Max. depth	=2	Max. depth	=3
	Avg. depth	Avg. no.	Avg. depth	Avg. no.
Mushroom	1.278	35	2.352	42
OCR17	1.537	48	2.467	45
OCR49	1.367	38	2.327	37
Ionosphere	1.342	32	2.316	42
BreastCancer	1.325	42	2.341	44
C,D,W→A	1.532	46	2.437	42
A,D,W→C	1.487	38	2.735	44
A,C,W→D	1.668	49	2.652	47
A,C,D→W	1.527	41	2.742	45
PIE05	1.728	37	2.643	43
PIE27	1.634	43	2.631	44

Table IX shows the average depth (avg. depth) and the average number (avg. no) of the final hypotheses after 100 iterations. Note that SURF feature is used in this table for Office+Caltech-256 data set. For methods with fixed depth and the fixed number of trees (i.e., AdaBoost, TrAdaBoost, Cost-TrAdaBoost, and Dynamic-TrAdaBoost), the depth is set as the maximum depth shown as "Max. depth," and the number of trees is set as the number of iterations, which is 100. It demonstrates that our method could adaptively adjust the depth of the hypothesis. It also demonstrates that instead of adding a new hypothesis in each iteration, our method compares the newly learned hypothesis with all the previously learned hypotheses, and increases the weight for the hypothesis with the largest gradient.

F. Discussion of the Ratio in Transfer Learning

In this section, we discuss the impact of the ratio of number of target-domain samples to source-domain samples. We compare both nontransfer learning methods Adaboost and DeepAdaBoost, and instance transfer learning methods TrAdaboost, Cost-TrAdaBoost, Dynamic-TrAdaBoost with our method. AdaBoost, TrAdaBoost, Cost-TrAdaBoost, Dynamic-TrAdaBoost are with base hypotheses H_2^{stumps}.

Fig. 3 shows the curves of error rates under the ratio from 0.01 to 0.1 in Mushroom and Ionosphere data sets. AdaBoost, TrAdaBoost, Cost-TrAdaBoost, Dynamic-TrAdaBoost are with H_2^{stumps}. We also conduct experiments with the ratio from

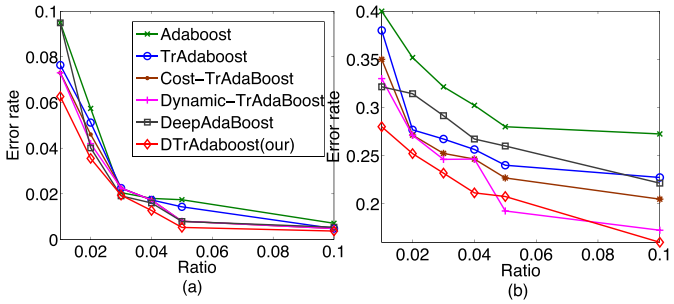


Fig. 3. Error rates under the ratio from 0.01 to 0.1 in (a) Mushroom and (b) ionosphere data sets. The ratio means the ratio of target-domain training data to source-domain training data. The legend in (b) is the same as (a).

0.2 to 0.5 and we observe that there is no significant difference between transfer methods under the ratio from 0.2 to 0.5.

First, we could see that when the ratio is very small such as 0.01, all the transfer learning methods are with the lower error rates compared to nontransfer learning methods. It shows that with limited number of training data, nontransfer learning methods are failed to learn good hypotheses, which is known as the data sparsity problem.

Second, when the ratio increases, the error rates of all the methods decrease. Meanwhile, the difference of the error rate between transfer and nontransfer learning methods decreases. When the ratio is 0.1, the difference between transfer and nontransfer learning methods is very small. We also conduct experiments with the ratio from 0.2 to 0.5 and we observe that the difference between transfer methods is not significant under the ratio from 0.2 to 0.5. This phenomenon is similar as the observation in [14].

Third, in Fig. 3, we could see that under the ratio from 0.01 to 0.1, our method achieves the lowest error rate of all the instance transfer learning methods, and DeepAdaBoost performs better than AdaBoost. It shows the effectiveness of the "deep" strategy in both transfer and nontransfer learning tasks.

VI. Conclusion

In this paper, we proposed a novel instance transfer learning method, DTrBoost. Compared with existing boosting-based transfer learning methods of complexity-fixed hypotheses, DTrBoost ensembled different complexity hypotheses families and added the complexity penalty on the hypotheses. It allocated larger weights to the hypotheses drawn from less complexity families and smaller weights to the hypotheses drawn from higher complexity families. In this way, DTrBoost could avoid overfitting when applying more complex hypotheses (e.g., deep decision tree) to deal with more complex cases such as computer vision data. In each iteration, a depth adaptive hypothesis was learned with the complexity penalty from both the source-domain and the target-domain data and then DTrBoost choose the direction with the largest gradient to optimize the objective function. We theoretically analyzed our algorithm in terms of the convergence property. Experimental results demonstrated the effectiveness of our method. In the future, we plan to extend DTrBoost to multisource and multiclass scenarios.

Appendix A
Proof of Lemma 1 and Theorem 2

A. Proof of Lemma 1

Lemma 1 could be proved according to Lemma 1 in Adaboost [22] and Theorem 2 in TrAdaboost [14]. It means that after t iterations ($t \geq T_0$), the normalizer of the source domain becomes a very small number. We follow [14] and set T_0 as $T/2$. The proof of Lemma 1 is as following.

Proof:

As shown in Lemma 1 in Adaboost [22], for any sequence of loss vectors ℓ^1,..., ℓ^t, in each iteration t, we have

$$S_t^d \leq \exp\big(-(1-\gamma)\textstyle\sum_{\tau=1}^{\mathrm{t}}\sum_{\mathrm{i}}^{\mathrm{n}} \mathrm{p}_{\mathrm{i}}^{\tau}\cdot\ell_{\mathrm{i}}^{\tau}\big)$$

where S_t^d denotes the normalizer of the source domain in the tth iteration. ℓ_i^τ denotes the loss of the training instance x_i in the τth iteration and p_i^t denotes the weight of x_i. γ denotes a weight updated root and $\gamma = \exp(-0.5 \times \log(1 + (2\ln \mathrm{n}/\mathrm{T})^{(1/2)}$, and n denotes the number of source-domain data.

According to Theorem 2 in TrAdaboost [14]

$$\lim_{T\to\infty}\Big(\sum_{t=\lceil T/2\rceil}^{T}\sum_{i}^{n} p_i^t\cdot l_i^t\Big)/(T-\lceil T/2\rceil)=0$$

which means that after $t \geq T_0$ iteration (T_0 as $T/2$), $\sum_{\tau=1}^{t}\sum_{i}^{n} p_i^\tau\cdot\ell_i^\tau$ is convergent. Thus, we have $S_t^d \leq \exp(-(1-\gamma)\iota)$, where ι is a small number and convergent after ($t \geq T_0$). To this end, we have proved the Lemma 1.

B. Proof of Theorem 2

Here, we provide the proof of Theorem 2 that, $\exists T_0$, $t \geq T_0$, $|F_w^{'t}(\alpha_{t-1,j}, e_j) - F_s^{'t}(\alpha_{t-1,j}, e_j)| \leq (2\epsilon_{t,j}^s)(S_t^d/m) \leq (2\epsilon_{t,j}^s)(exp(-(1-\gamma)\iota)/m)$.

$F_w^{'t}(\alpha_{t-1,j}, e_j)$ and $F_s^{'t}(\alpha_{t-1,j}, e_j)$ have been described in (7) and (9) in Section III-E. To simplify, we write $|F_w^{'t}(\alpha_{t-1,j}, e_j) - F_s^{'t}(\alpha_{t-1,j}, e_j)|$ as $|F'_w - F'_s|$. The proof of Theorem 2 is as.

Proof:

We could easily calculate $|F'_w - F'_s|$ according to (7) and (9) as

$$\begin{aligned}|F'_w - F'_s| &= \Bigg|\frac{1}{m+n}\sum_{i=1}^{m+n} y_i h_j(x_i)\Phi'(1-y_i f_{t-1}(x_i)) \\ &\quad -\frac{1}{m}\sum_{i=n}^{m+n} y_i h_j(x_i)\Phi'(1-y_i f_{t-1}(x_i))\Bigg| \\ &= \Big|(2\varepsilon_{t,j}^w - 1)\frac{S_t^s + S_t^d}{m+n} - (2\varepsilon_{t,j}^s - 1)\frac{S_t^s}{m}\Big| \end{aligned} \quad (12)$$

where m and n denote the number of target- and source-domain data. y_i denotes the label of x_i. S_t^s and S_t^d denote the normalizer of target- and source-domain data, respectively. $\varepsilon_{t,j}^s$ and $\varepsilon_{t,j}^w$ denote the training error of target-domain data and whole data in the tth iteration and jth direction, respectively.

With the increase in iterations, the weights of the wrongly classified source-domain data are gradually decreased.

When $t \geq T_0$, we have $S_t^d \varepsilon_{t,j}^d \ll S_t^s \varepsilon_{t,j}^s$. Then, we have

$$\varepsilon_{t,j}^w = \frac{S_t^s \varepsilon_{t,j}^s + S_t^d \varepsilon_{t,j}^d}{S_t^s + S_t^d} \leq \frac{S_t^s \varepsilon_{t,j}^s + S_t^s \varepsilon_{t,j}^s}{S_t^s + S_t^d} \leq 2\varepsilon_{t,j}^s. \quad (13)$$

Thus, (14) would be written as

$$\begin{aligned} |F_w' - F_s'| &= \left| \left(4\varepsilon_{t,j}^s - 1\right) \frac{S_t^s + S_t^d}{m+n} - \left(2\varepsilon_{t,j}^s - 1\right) \frac{S_t^s}{m} \right| \\ &\leq \left| \left(2\varepsilon_{t,j}^s - 1\right) \frac{S_t^d + S_t^s}{m+n} - \left(2\varepsilon_{t,j}^s - 1\right) \frac{S_t^s}{m} \right. \\ &\quad \left. + 2\varepsilon_{t,j}^s \frac{S_t^d + S_t^s}{m+n} \right|. \end{aligned} \quad (14)$$

Since $m \ll n$, $S_t^s < n$, $S_t^d < n$, and according to the Lemma 1, we have

$$\begin{aligned} |F_w' - F_s'| &\leq \left| \left(2\varepsilon_{t,j}^s - 1\right) \left(\frac{S_t^d}{m}\right) + 2\varepsilon_{t,j}^s \right| \\ &\leq \left| \left(2\epsilon_{t,j}^s - 1\right) \frac{\exp(-(1-\gamma)\imath)}{m} + 2\varepsilon_{t,j}^s \right|. \end{aligned} \quad (15)$$

To this end, we have proved the Theorem 2.

REFERENCES

[1] L. Shao, F. Zhu, and X. Li, "Transfer learning for visual categorization: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 5, pp. 1019–1034, May 2014.

[2] Z. Ding, M. Shao, and Y. Fu, "Incomplete multisource transfer learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 2, pp. 310–323, Feb. 2018.

[3] D. Zhang, J. Han, J. Han, and L. Shao, "Cosaliency detection based on intrasaliency prior transfer and deep intersaliency mining," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 6, pp. 1163–1176, Jun. 2016.

[4] N. Segev, M. Harel, S. Mannor, K. Crammer, and R. El-Yaniv, "Learn on source, refine on target: A model transfer learning framework with random forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 9, pp. 1811–1824, Sep. 2017.

[5] Y. Lu, L. Chen, A. Saidi, E. Dellandrea, and Y. Wang, "Discriminative transfer learning using similarities and dissimilarities," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 3097–3110, Jul. 2018.

[6] S. Sun, H. Liu, J. Meng, C. L. P. Chen, and Y. Yang, "Substructural regularization with data-sensitive granularity for sequence transfer learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2545–2557, Jun. 2018.

[7] L. Yang, L. Jing, J. Yu, and M. K. Ng, "Learning transferred weights from co-occurrence data for heterogeneous transfer learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 11, pp. 2187–2200, Nov. 2016.

[8] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[9] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: Transfer learning from unlabeled data," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 759–766.

[10] D. Cook, K. D. Feuz, and N. C. Krishnan, "Transfer learning for activity recognition: A survey," *Knowl. Inf. Syst.*, vol. 36, no. 3, pp. 537–556, Sep. 2013.

[11] J. Jiang and C. Zhai, "Instance weighting for domain adaptation in NLP," in *Proc. ACL*, vol. 7, 2007, pp. 264–271.

[12] X. Liao, Y. Xue, and L. Carin, "Logistic regression with an auxiliary data source," in *Proc. 22nd Int. Conf. Mach. Learn.*, 2005, pp. 505–512.

[13] S. Zhou, E. Smirnov, G. Schoenmakers, and R. Peeters, "Decision trees for instance transfer," in *Proc. Symp. Conformal Probabilistic Predict. Appl.* Cham, Switzerland: Springer, 2016, pp. 116–127.

[14] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 193–200.

[15] A. Venkatesan, N. C. Krishnan, and S. Panchanathan, "Cost-sensitive boosting for concept drift," in *Proc. Int. Workshop Handling Concept Drift Adapt. Inf. Syst.*, 2010, pp. 41–47.

[16] S. Al-Stouhi and C. K. Reddy, "Adaptive boosting for transfer learning using dynamic updates," in *Machine Learning and Knowledge Discovery in Databases*. Berlin, Germany: Springer, 2011, pp. 60–75.

[17] E. Eaton and M. desJardins, "Set-based boosting for instance-level transfer," in *Proc. IEEE Int. Conf. Data Mining Workshops*, Dec. 2009, pp. 422–428.

[18] C. Wang, Y. Wu, and Z. Liu, "Hierarchical boosting for transfer learning with multi-source," in *Proc. Int. Conf. Artif. Intell. Robot., Int. Conf. Automat., Control Robot. Eng.*, 2016, p. 15.

[19] J. Xu, D. Vázquez, S. Ramos, A. M. López, and D. Ponsa, "Adapting a pedestrian detector by boosting LDA exemplar classifiers," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2013, pp. 688–693.

[20] R. L. Vieriu *et al.*, "Boosting-based transfer learning for multi-view head-pose classification from surveillance videos," in *Proc. 20th Eur. Signal Process. Conf.*, Aug. 2012, pp. 649–653.

[21] B. Jiang and K. Jia, "Semi-supervised facial expression recognition algorithm on the condition of multi-pose," *J. Inf. Hiding Multimedia Signal Process.*, vol. 4, no. 3, pp. 138–146, 2013.

[22] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[23] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, 2001.

[24] C. Cortes, M. Mohri, and U. Syed, "Deep boosting," in *Proc. 31th Int. Conf. Mach. Learn.*, 2014, pp. 1179–1187.

[25] P. L. Bartlett and S. Mendelson, "Rademacher and Gaussian complexities: Risk bounds and structural results," *J. Mach. Learn. Res.*, vol. 3, pp. 463–482, Mar. 2003.

[26] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.

[27] P. Smyth and D. Wolpert, "Linearly combining density estimators via stacking," *Mach. Learn.*, vol. 36, no. 1, pp. 59–83, Jul. 1999.

[28] R. E. Schapire, "The boosting approach to machine learning: An overview," in *Nonlinear Estimation and Classification*. Springer, 2003, pp. 149–171.

[29] G. Ditzler, J. LaBarck, J. Ritchie, G. Rosen, and R. Polikar, "Extensions to online feature selection using bagging and boosting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 4504–4509, Sep. 2018.

[30] Y. Li, S. Wang, Q. Tian, and X. Ding, "A boosting approach to exploit instance correlations for multi-instance classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 12, pp. 2740–2747, Dec. 2016.

[31] R. G. F. Soares, H. Chen, and X. Yao, "Efficient cluster-based boosting for semisupervised classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5667–5680, Nov. 2018.

[32] C. Zhang, P. Lim, A. K. Qin, and K. C. Tan, "Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2306–2318, Oct. 2017.

[33] P. Bartlett, Y. Freund, W. S. Lee, and R. E. Schapire, "Boosting the margin: A new explanation for the effectiveness of voting methods," *Ann. Statist.*, vol. 26, no. 5, pp. 1651–1686, 1998.

[34] V. Koltchinskii and D. Panchenko, "Empirical margin distributions and bounding the generalization error of combined classifiers," *Ann. Statist.*, vol. 30, no. 1, pp. 1–50, 2002.

[35] J. Bjurgert, P. E. Valenzuela, and C. R. Rojas, "On adaptive boosting for system identification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 4510–4514, Sep. 2018.

[36] Z. Qi, F. Meng, Y. Tian, L. Niu, Y. Shi, and P. Zhang, "Adaboost-LLP: A boosting method for learning with label proportions," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 8, pp. 3548–3559, Aug. 2018.

[37] D. Pardoe and P. Stone, "Boosting for regression transfer," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 863–870.

[38] Y. Yao and G. Doretto, "Boosting for transfer learning with multiple sources," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 1855–1862.

[39] X. Huang, Y. Rao, H. Xie, T.-L. Wong, and F. L. Wang, "Cross-domain sentiment classification via topic-related TrAdaBoost," in *Proc. AAAI*, 2017, pp. 4939–4940.

[40] D. Ryu, J.-I. Jang, and J. Baik, "A transfer cost-sensitive boosting approach for cross-project defect prediction," *Softw. Qual. J.*, vol. 25, no. 1, pp. 235–272, 2017.

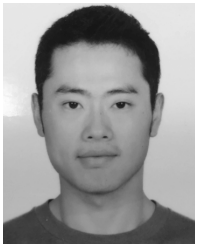
[41] J. Hu, J. Lu, and Y.-P. Tan, "Deep transfer metric learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 325–333.

[42] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3320–3328.
[43] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 1180–1189.
[44] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 97–105.
[45] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Simultaneous deep transfer across domains and tasks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 4068–4076.
[46] J. T. Zhou, H. Zhao, X. Peng, M. Fang, Z. Qin, and R. S. M. Goh, "Transfer hashing: From shallow to deep," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 6191–6201, Dec. 2018.
[47] Z. Ding and Y. Fu, "Deep transfer low-rank coding for cross-domain learning," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published. [Online]. Available: https://ieeexplore.ieee.org/document/8513988
[48] Y. Sun, M. S. Kamel, A. K. C. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognit.*, vol. 40, no. 12, pp. 3358–3378, 2007.
[49] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2066–2073.
[50] J. Zhang, W. Li, and P. Ogunbona. (2017). "Joint geometrical and statistical alignment for visual domain adaptation." [Online]. Available: https://arxiv.org/abs/1705.05498
[51] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 675–678.
[52] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[53] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 2200–2207.
[54] M. Ghifary, W. B. Kleijn, M. Zhang, and D. Balduzzi, "Domain generalization for object recognition with multi-task autoencoders," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 2551–2559.
[55] Z. Ding and Y. Fu, "Deep domain generalization with structured low-rank constraint," *IEEE Trans. Image Process.*, vol. 27, no. 1, pp. 304–313, Jan. 2018.

Shuhui Jiang received the B.S. and M.S. degrees in Xi'an Jiaotong University, Xi'an, China, in 2007 and 2011, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA in 2018.

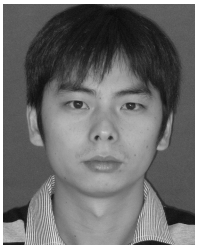
She was a Research Intern with the Adobe Research Lab, San Jose, CA, USA, in 2016. Her current research interests include machine learning, multimedia, and computer vision.

Dr. Jiang was a recipient of the Dean's Fellowship of Northeastern University in 2014 and the Best Paper Candidate (ACM MM 2017). She has served as a reviewer for IEEE journals such as the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS and the IEEE TRANSACTIONS ON MULTIMEDIA.



Haiyi Mao received the B.S. degree in computer science from Xidian University, Xi'an, China, in 2012, and the M.Sc. degree in computer science from Northeastern University, Boston, MA, USA.

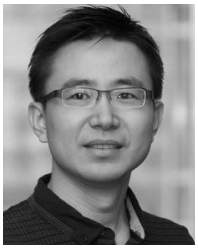
His current research interests include machine learning, deep learning, and data mining.



Zhengming Ding (S'14–M'18) received the B.Eng. degree in information security and the M.Eng. degree in computer software and theory from the University of Electronic Science and Technology of China, Chengdu, China, in 2010 and 2013, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA, in 2018.

He has been a Faculty Member with the Department of Computer, Information and Technology, Indiana University—Purdue University Indianapolis, Indianapolis, IN, USA, since 2018. His current research interests include transfer learning, multiview learning, and deep learning.

Dr. Ding was a recipient of the National Institute of Justice Fellowship from 2016 to 2018, the Best Paper Award (SPIE 2016), and the Best Paper Candidate (ACM MM 2017). He is an Associate Editor of the *Journal of Electronic Imaging*.



Yun Fu (S'07–M'08–SM'11–F'19) received the B.Eng. degree in information engineering and the M.Eng. degree in pattern recognition and intelligence systems from Xi'an Jiaotong University, Xi'an, China, respectively, and the M.S. degree in statistics and the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign, Urbana and Champaign, IL, USA, respectively.

He has been an Interdisciplinary Faculty Member with the College of Engineering and the Khoury College of Computer and Information Sciences, Northeastern University, Boston, MA, USA, since 2012. His current research interests include machine learning, computational intelligence, big data mining, computer vision, pattern recognition, and cyber-physical systems.

Dr. Fu is a fellow of IAPR, OSA, and SPIE. He is a Lifetime Distinguished Member of ACM, a Lifetime Member of AAAI and Institute of Mathematical Statistics, a member of ACM Future of Computing Academy, Global Young Academy, AAAS, INNS, and a Beckman Graduate Fellow from 2007 to 2008. He was a recipient of the seven Prestigious Young Investigator Awards from NAE, ONR, ARO, IEEE, INNS, UIUC, Grainger Foundation, nine Best Paper Awards from IEEE, IAPR, SPIE, and SIAM, and many major Industrial Research Awards from Google, Samsung, and Adobe. He serves as an Associate Editor, the Chair, a PC Member, and a reviewer for many top journals and international conferences/workshops. He is currently an Associate Editor of the IEEE and TNNLS.