Domain Adversarial Reinforcement Learning for Partial Domain Adaptation

Jin Chen, Xinxiao Wu, *Member, IEEE*, Lixin Duan, and Shenghua Gao, *Member, IEEE*

***Abstract*— Partial domain adaptation aims to transfer knowledge from a label-rich source domain to a label-scarce target domain (i.e., the target categories are a subset of the source ones), which relaxes the common assumption in traditional domain adaptation that the label space is fully shared across different domains. In this more general and practical scenario on partial domain adaptation, a major challenge is how to select source instances from the shared categories to ensure positive transfer for the target domain. To address this problem, we propose a domain adversarial reinforcement learning (DARL) framework to progressively select source instances to learn transferable features between domains by reducing the domain shift. Specifically, we employ a deep Q-learning to learn policies for an agent to make selection decisions by approximating the action-value function. Moreover, domain adversarial learning is introduced to learn a common feature subspace for the selected source instances and the target instances, and also to contribute to the reward calculation for the agent that is based on the relevance of the selected source instances with respect to the target domain. Extensive experiments on several benchmark data sets clearly demonstrate the superior performance of our proposed DARL over existing state-of-the-art methods for partial domain adaptation.**

***Index Terms*— Adversarial learning, partial domain adaptation, reinforcement learning.**

I. Introduction

In THE machine learning community, deep learning has achieved great success in a wide variety of tasks and applications, owing to abundant labeled training data. However, it is time consuming and labor intensive to collect and annotate a large number of data. Thus, domain adaptation is proposed to leverage the labeled data from a different but related domain (source domain) to boost the performance on an unlabeled but interested domain (target domain). It has been widely used in many research fields, such as natural language processing [1]–[5], object classification [6]–[8], action recognition [9], [10], person re-identification [11]–[15], video analysis [16]–[19], and so on. The core idea of domain adaptation is reducing the domain shift between the source and target domains [20].

To reduce the domain shift, many domain adaptation methods learn domain-invariant features by matching statistic moments of different domains [7], [8], [21], [22], utilizing domain adversarial networks [23]–[26], and integrating a batch normalization layer into networks [27]–[30]. However, all these methods assume that the label spaces between the source and target domains are the same. But in real applications, it is hard to hold the assumption of shared label space since sometimes there is no class label in the target domain and the classes of the target domain are unknown. To relax the shared label space assumption, a novel learning paradigm, called partial domain adaptation, is proposed where the label space of the target domain is a subspace of the label space of the source domain. This makes partial domain adaptation more general and practical with growing attention. The classes shared between the source and target domains are defined as shared classes, and the classes only in the source domain but not in the target domain are defined as outlier classes.

Due to the mismatch of label spaces between the source and the target domains in partial domain adaptation, directly aligning the data distributions between different domains [7], [31], [32] will lead to a negative transfer. To address this problem, several existing methods resort to improving the importance of source instances in the shared classes and reducing the importance of source instances in the outlier classes. For example, Zhang *et al.* [33] apply a two-domain classifier to learn the weights of source instances. In [34] and [35], the weights of source instances are determined by the class probability distribution of target instances based on their prediction scores obtained from the source classifier. Cao *et al.* [36] designs a weighting scheme to quantify the transferability of source instances based on the decision scores of a domain classifier.

Motivated by the successes of these works, in this article, we select source instances in the shared classes and use them as anchors to learn an adaptive classifier for the target domain. Since there is no available label in the target domain, it is nontrivial for us to perform the source instance selection. Unlike some existing works which select source instances based on pseudolabels [33]–[36], we formulate the selection

Manuscript received November 5, 2019; revised June 25, 2020 and September 17, 2020; accepted September 24, 2020. This work was supported in part by the Major Project for New Generation of AI under Grant 2018AAA0100400 and in part by the Natural Science Foundation of China under Grant 61673062, Grant 62072041, and Grant 61772118. *(Corresponding author: Xinxiao Wu.)*

Jin Chen and Xinxiao Wu are with the Beijing Laboratory of Intelligent Information Technology, Beijing Institute of Technology, Beijing 100081, China, and also with the School of Computer Science, Beijing Institute of Technology, Beijing 100081, China (e-mail: chen_jin@bit.edu.cn; wuxinxiao@bit.edu.cn).

Lixin Duan is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China, and also with the Big Data Research Center, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: lxduan@uestc.edu.cn).

Shenghua Gao is with the School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China (e-mail: gaoshh@shanghaitech.edu.cn).

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

Digital Object Identifier 10.1109/TNNLS.2020.3028078

of source instances as a Markov decision process and employ a reinforcement learning paradigm to automate the selection procedure. Specifically, selections are taken via a sequence of actions that are determined by both the immediate reward and future rewards. This makes the selections globally optimized and outperforms the selections based on pseudolabels at the instance level in existing works. Furthermore, with the superior exploration ability of reinforcement learning, the selection policies are searched in a wider solution space. We term our solution as domain adversarial reinforcement learning (DARL), which couples deep Q-learning with domain adversarial learning [31]. The deep Q-learning learns policies for selecting source instances in the shared classes via rewards that are defined by the relevance of source instances to the target domain. The domain adversarial learning learns the common feature space of the selected source instances and the target instances and simultaneously calculates the relevance of the selected source instances to the target domain for providing rewards to the agent.

Specifically, a deep Q-learning network (DQN) is built to approximate an action-value function to help the agent make selection policies, which takes agent state as input and outputs values of different agent actions. The actions are corresponding to source instances that are ready to be selected, and the states are represented by the feature vectors of those source instances. At each time step, according to the output values of the DQN, the agent takes action to select the corresponding source instance. The reward of this action is defined by the relevance of the selected source instance to the target domain that is measured by the data distribution difference between the selected source instance and the target domain in the common feature space. After executing this action, the current state is updated by filtering out the selected source instance. Then, the reward of this action and the updated state are fed back to the agent for updating the DQN and making the next selection.

After several selections, the selected source instances and the target instances are used for updating the domain adversarial learning network to learn the common feature subspace. The domain adversarial learning network consists of a feature extractor (generator) and a domain classifier (discriminator). The domain classifier aims to distinguish the source domain from the target domain, and the feature extractor tries to confuse the domain classifier to make the features as indistinguishable as possible. By taking advantage of such min–max game, the domain shift can be significantly reduced when the domain classifier is maximally confused. The decision score of the domain classifier reflects the data distribution difference between the selected source instances and target domain, which is served to measure the relevance of the selected source instance to the target domain. Furthermore, a classifier is trained on the selected source data and adapted well to the target domain, which takes the features in the common feature spaces as input and outputs the prediction of class label.

We propose an iterative optimization algorithm to jointly train a DQN and a domain adversarial learning network in an end-to-end manner. The domain adversarial learning network guides the agent to learn the right selection policy by providing rewards to avoid the negative transfer, and in return, the optimal source subset discovered by the Q-learning network enables the domain adversarial learning network to learn a common feature space for facilitating the positive transfer.

The main contributions are summarized as follows.

1) We propose a novel DARL framework for partial domain adaptation. With superior exploration ability of reinforcement learning and good performance of domain adversarial learning on the domain shift reduction, DARL is able to automatically select meaningful source instances from the shared categories and simultaneously learn transferable features between different domains.
2) We design a novel reward function based on domain adversarial learning, which guides the agent to learn suitable selection policies by considering the relevance of the selected source instances with respect to the target domain.
3) Evaluations on various benchmark data sets clearly show that DARL achieves superior results compared with existing state-of-the-art methods for partial domain adaptation.

II. Related Work

A. Partial Domain Adaptation

Existing partial domain adaptation methods focus on up-weighting source instances in the shared classes. Cao *et al.* [34] introduces multiple domain classifiers for fine-grained adaptation, where the class probabilities of each instance modeled by the source classifier are used as the weights for domain classifiers. Cao *et al.* [35] extend the domain adversarial network [31] by learning the weights of source classes and source instances, where the weights are computed with the class probability of target data predicted by the source classifier. In [33], two domain classifiers are introduced to learn the domain-invariant features by calculating the weights of source instances with the predicated domain scores. Cao *et al.* [36] introduced an Example Transfer Network (ETN) for partial domain adaptation, where a weighting scheme is designed to quantify the transferability of source examples.

Rather than learning weights of source instances in those methods, our method introduces reinforcement learning to select source instances in the shared classes according to sequential decisions of the agent.

B. Reinforcement Learning

Reinforcement learning [37] has achieved significant success in many fields such as robot controlling [38] and machine gaming [39], [40]. In computer vision, reinforcement learning also has wide applications, including video captioning [41]–[45], action recognition [46], [47], object tracking [48]–[51], object detection [52]–[56], and one shot learning [57]. Wang *et al.* [43] designs a novel hierarchical reinforcement learning network, where a high-level agent learns

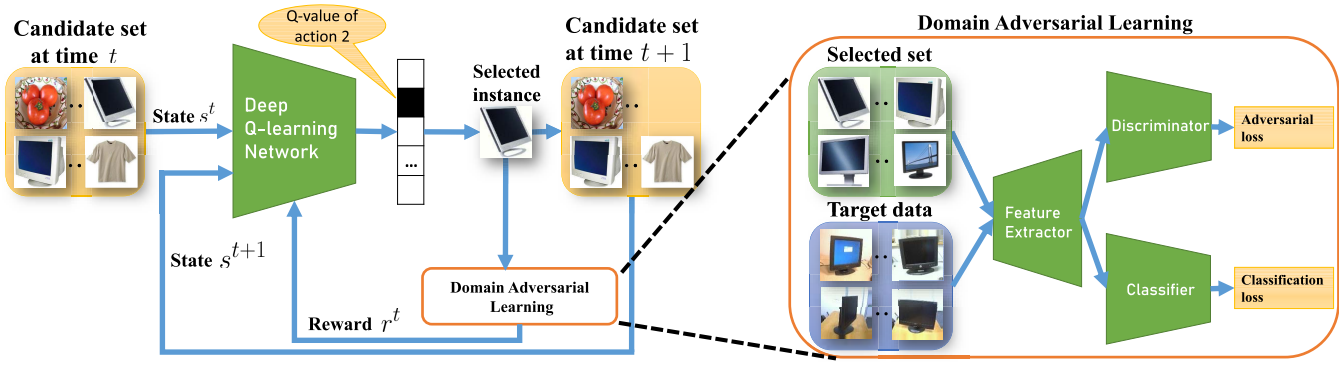


Fig. 1. Framework of our DARL. Deep Q-learning is used to learn policies for selecting source instances in the shared classes. A DQN approximates the action-value function, and then the agent selects one instance according to the estimated Q value. The reward is determined by how relevant the source instances are to the target domain measured by the domain adversarial learning. After n selections, (i.e., obtaining a selected set with n source instances), both the selected set and the target data set are used to learn domain-invariant features via domain adversarial learning.

to design subgoals, and a low-level agent learns to fulfill the subgoal. In [49], each object is considered as an agent, and the optimal tracked results are exploited by the collaborative interactions of different agents via the decision network. Pirinen and Sminchisescu [56] design a reinforcement learning-based region proposal network to leverage class-specific information and context information for multiple object detection. Dong and Xing [57] introduces a policy network to estimate the similarity of the input target image to all source images to optimize the sampling policy in one-shot learning.

Different from the aforementioned works, we apply reinforcement learning to partial domain adaptation, where the agent takes actions to select source instances in the shared classes for improving positive transfer.

C. Domain Adversarial Learning

Generative adversarial networks (GANs) [58] have been widely used in domain adaptation [23]–[26], [59] and are usually implemented by training a domain classifier (discriminator) and a feature extractor (generator) in an adversarial manner, denoted as domain adversarial learning. Specifically, in [24], a feature extractor and a domain predictor play a max–min game to learn domain-invariant features between the source and target domains. Generate To Adapt (GTA) [26] proposes a GANs-based method to learn an embedding that is robust to the domain shift. Adversarial Discriminative Domain Adaptation (ADDA) [23] first learns a discriminative feature representation with the labeled source data and then maps the target data to the same space by a domain adversarial loss. Pei *et al.* [25] designed multiple domain classifiers to capture multimode structures for avoiding the mismatch of categories. In [59], the attention mechanism is utilized to learn the weights of multiple regions of images with multiple region-level domain classifiers.

In this work, we use adversarial learning to train a novel domain classifier that not only captures the domain information but also preserves the class information of source domain, due to that the classes in source and target domains are different and the domain classifier should take into account of the class information.

III. Domain Adversarial Reinforcement Learning

A. Problem Definition

For the partial domain adaptation in an unsupervised scenario, we are given a labeled source domain $\mathcal{D}_s = \{(x_i^s, y_i^s)|_{i=1}^{N_s}\}$ drawn i.i.d from the source distribution $p(x)$ with $y_i^s \in \mathcal{Y}_s$ and an unlabeled target domain $\mathcal{D}_t = \{x_j^t|_{j=1}^{N_t}\}$ drawn i.i.d from the target distribution $q(x)$. N_s and N_t are the numbers of instances in the source and target domains, respectively. The target class label space $\mathcal{Y}_t$ is a subspace of the source class label space $\mathcal{Y}_s$, i.e., $\mathcal{Y}_t \subset \mathcal{Y}_s$. The classes in $\mathcal{Y}_s$ but not in $\mathcal{Y}_t$ are denoted as outlier classes and source instances in the outlier classes are denoted as outlier source instances for short. The common classes in $\mathcal{Y}_s$ and $\mathcal{Y}_t$ are denoted as shared classes. The data distributions of source and target domains are different, i.e., $p(x) \neq q(x)$.

The task of partial domain adaptation is to reduce the difference between $p(x)$ and $q(x)$. Due to the mismatch of $\mathcal{Y}_s$ and $\mathcal{Y}_t$, directly matching $p(x)$ with $q(x)$ is easily prone to a negative transfer. Thus, we should filter out outlier source instances and then reduce the domain shift between the optimal subset of the source domain and the target domain. This problem is formulated as learning an optimized subset $\hat{\mathcal{D}}_s$ of the source domain $\mathcal{D}_s$, expressed as

$$\begin{aligned} &\hat{\mathcal{D}}_s = G(\mathcal{D}_s, \mathcal{D}_t) \\ &\text{s.t. } \hat{\mathcal{D}}_s \subset \mathcal{D}_s \end{aligned} \tag{1}$$

where G represents a selection function that selects source instances with the class labels $y_i^s \in \mathcal{Y}_t$ to construct the optimal subset of source domain $\hat{\mathcal{D}}_s$. With this in mind, we propose a DARL framework that couples reinforcement learning with domain adversarial learning. The reinforcement learning is applied to learn a selection function G for obtaining $\hat{\mathcal{D}}_s$ and the domain adversarial learning is utilized to reduce the domain shift between $\hat{\mathcal{D}}_s$ and $\mathcal{D}_t$ simultaneously.

B. Network Overview

The architecture of our DARL is shown in Fig. 1, which includes a DQN and a domain adversarial learning network. DQN approximates the action-value function by learning

a mapping from agent state to action value, and the action value is determined by the immediate reward of the action and the future rewards. The domain adversarial learning network aims to reduce the domain shift between the optimal subset of the source domain and the target domain, which consisting of a feature extractor, a domain classifier, and a classifier. The reward for the agent is computed by the domain classifier in the domain adversarial learning network. DQN and the domain adversarial learning network are optimized in an end-to-end learning manner. The reward of the agent in DQN comes from the output of the domain classifier in the domain adversarial learning network, while the selected instances by the agent are used to train the domain adversarial learning network in return.

C. Deep Q-Learning

The deep Q-learning is applied to learn policies for selecting source instances in the shared classes. We define a candidate set $\mathcal{D}_c$ that consists of source instances to be selected and is initialized as the randomly sampled instances from the source domain, and a selected set $\mathcal{D}_e$ that is constructed by the selected source instances and initialized to empty. At time step t, the agent takes an action a_t according to the Q value $Q(\boldsymbol{s}_t, a)$ estimated by DQN with the state $\boldsymbol{s}_t$ as input. The action a_t is equivalent to selecting the corresponding instance from the candidate set $\mathcal{D}_c$ and moving it to the selected set $\mathcal{D}_e$. The reward R_t of action a_t and the next state s_{t+1} are sent to the agent for the next selection. This is one selection process of the agent. In each episode of deep Q-learning, the agent makes several selections until it reaches the terminal state on the candidate set.

1) State: At the initial stage of an episode, given the candidate set $\mathcal{D}_c = \{(x_i^c, y_i^c)|_{i=1}^{N_c}\}$ with N_c instances and the initial selected set $\mathcal{D}_e = \varnothing$, the initial state $\boldsymbol{s}_0$ is constructed by the feature vectors of instances in $\mathcal{D}_c$, represented by $\boldsymbol{s}_0 = [F(x_1^c), \ldots, F(x_{N_c}^c)] \in \mathbb{R}^{d \times N_c}$, where $F(x_i^c)$ denotes the d-dimensional feature vector of instance x_i^c extracted by the feature extractor F in the domain adversarial learning network. After taking an action, the corresponding instance in $\mathcal{D}_c$ is moved from $\mathcal{D}_c$ to $\mathcal{D}_e$. Thus, the size of state is changed from $d \times N_c$ to $d \times (N_c - 1)$. In order to keep the size of the state constant, we replace the selected instance with a zero-valued feature vector.

2) Action: An action is defined as selecting an instance from the candidate set $\mathcal{D}_c$. At each time step, the agent takes an action from the action set $A = \{a_1, a_2, \ldots, a_{N_c}\}$, where a_i means that selecting the ith instance in $\mathcal{D}_c$ and then moving it to $\mathcal{D}_e$. The number of actions is the same as the number of instances in $\mathcal{D}_c$, i.e., N_c. The optimal action taken by the agent at time step t is learned by

$$a_t = \max_a Q(\boldsymbol{s}_t, a) \tag{2}$$

where s_t indicates the state at time step t and the Q value $Q(s_t, a)$ is the accumulated rewards of taking the action a. The action a with the max-value of $Q(s_t, a)$ is taken by the agent, denoted as a_t. DQN estimates $Q(s_t, a)$, which uses s_t as input and outputs a $|A|$-dimensional vector that represents the Q-values of $|A|$ actions.

3) Reward: The reward is the feedback of the corresponding action taken by the agent. It guides the agent to make selection decisions. Since the source instances in the shared classes should be more relevant to the target domain than the source instances in the outlier classes, we use the relevance of source instances to the target domain to design the reward.

When the agent takes action a_t to move the candidate instance x to the selected set $\mathcal{D}_e$, the reward of the action a_t is computed by

$$R_t = \begin{cases} +1, & \text{if } \varphi(x) > \tau \\ -1, & \text{otherwise} \end{cases} \tag{3}$$

where $\varphi(x)$ is a metric function that measures the relevance of instance x to the target domain and will be detailed in Section III-E. The more relevant the instance x is to the target domain, the higher the value of $\varphi(x)$ becomes. We adopt a binary reward, i.e., $+1$ and -1, which has been widely used in reinforcement learning for various tasks [48], [60], since a binary reward can help the agent clearly distinguish good or bad actions and provide more explicit guidance than directly using the relevance measure as a reward. If directly using the relevance measure, the relevance difference between different instances is too small to confuse the agent about which actions are good and which actions are bad. If $\varphi(x)$ is higher than a threshold τ, then the reward for the agent will be $+1$ otherwise, the reward will be -1. When the reward is -1, the agent reaches the terminal state, stops the selection on the current candidate set and begins a new selection on the next candidate set.

4) Objective: Based on the definitions of the state, action, and reward, the objective function of DQN is given by

$$\mathcal{L}_q = \mathbb{E}_{\boldsymbol{s}_t, a_t}\left[(V(\boldsymbol{s}_t) - Q(\boldsymbol{s}_t, a_t))^2\right] \tag{4}$$

where $V(\boldsymbol{s}_t) - Q(\boldsymbol{s}_t, a_t)$ is the temporal difference error. $V(\boldsymbol{s}_t)$ is the target value of $Q(\boldsymbol{s}_t, a_t)$, estimated by

$$V(\boldsymbol{s}_t) = \mathbb{E}_{\boldsymbol{s}_{t+1}}\left[R_t + \gamma \max_{a_{t+1}} Q(\boldsymbol{s}_{t+1}, a_{t+1}|\boldsymbol{s}_t, a_t)\right] \tag{5}$$

where the first term R_t is the immediate reward of taking the action a_t under the state s_t, computed by (3), and the second term is the future reward estimated by the current DQN with the next state $\boldsymbol{s}_{t+1}$ as input. If the current state $\boldsymbol{s}_t$ is terminal state, i.e., $R_t < 0$, the second term is 0.

D. Domain Adversarial Learning

The goal of domain adversarial learning is to learn common feature space of the selected source instances and target domain for reducing the domain shift, which is achieved by the adversarial learning procedure of a domain classifier D and a feature extractor F. The domain classifier D is trained to distinguish the source domain from the target domain, and the feature extractor F is trained to confuse the domain classifier D. Thus, the adversarial loss of domain adversarial

learning is summarized as the minimax form

$$\min_F \max_D \mathcal{L}_d(F, D) = \mathbb{E}_{x\sim p(x)} \log(D(F(x))) + \mathbb{E}_{x\sim q(x)} \log(1 - D(F(x))). \quad (6)$$

With the fixed F, the domain classifier D learns an optimal bound of the true domain distribution by maximizing the adversarial loss $\mathcal{L}_d(F, D)$. With the fixed D, the feature extractor F learns feature representations as indistinguishable as possible across domains by minimizing the adversarial loss $\mathcal{L}_d(F, D)$.

An optimal domain classifier does not only distinguish the source domain from the target domain but also identifies the class label of source instances. To this end, a $K + 1$-way classifier is introduced as the domain classifier D, where K is the number of source classes $K = |\mathcal{Y}_s|$. The first K ways model the class distribution, and the last way models the domain distribution. We use one-hot encoding to represent the class label of each instance x and add an element to represent whether x is from the target domain or not. The output of D is a $K + 1$-dimensional vector, where the first K elements represent the class label of x in the one-hot encoding form, and the last element denotes the probability of the instance coming from the target domain. The adversarial training between the domain classifier D and the feature extractor F is implemented via assigning different labels to the source and target instances when updating D or updating F. We present the implementation of adversarial training between D and F in detail as follows.

When optimizing D with the fixed F, the adversarial loss becomes

$$\min_D \mathcal{L}_y(F, D) = \mathbb{E}_{x_i^s\sim p(x)} H\big(D\big(F\big(x_i^s\big)\big), \tilde{\mathbf{y}}_d^s\big) + \mathbb{E}_{x_j^t\sim q(x)} H\big(D\big(F\big(x_j^t\big)\big), \tilde{\mathbf{y}}_d^t\big) \quad (7)$$

where $H(\cdot, \cdot)$ is a cross entropy loss. The label $\tilde{\mathbf{y}}_d^s$ of source instance x_i^s and the label $\tilde{\mathbf{y}}_d^t$ of target instance x_j^t are defined by

$$\tilde{\mathbf{y}}_d^s = \left[\underbrace{0, \ldots, 0, 1}_{l}, \overbrace{0, \ldots, 0, 0}^{K}, 0\right], \quad \big(x_i^s, y_i^s\big) \in \mathcal{D}_s,\ y_i^s = l$$
$$\tilde{\mathbf{y}}_d^t = \left[\overbrace{0, \cdots\cdots\cdots\cdots, 0}^{K}, 1\right], \quad x_j^t \in \mathcal{D}_t \quad (8)$$

where y_i^s is the class label of source instance x_i^s. The first K elements of $\tilde{\mathbf{y}}_d^s$ are the class label of source instance x_i^s in the one-hot encoding form. The last elements of $\tilde{\mathbf{y}}_d^s$ and $\tilde{\mathbf{y}}_d^t$ are 0 and 1, respectively, denoting that x_i^s comes from the source domain and x_j^t comes from the target domain. We expect that the domain classifier D can classify labeled source instances and assign unlabeled target instances into the target domain. Thus, when optimizing D by (7), $\tilde{\mathbf{y}}_d^s$ contains the source class information while $\tilde{\mathbf{y}}_d^t$ does not, as defined in (8).

When optimizing F with fixed D, the adversarial loss becomes

$$\min_F \mathcal{L}_y(F, D) = \mathbb{E}_{x_i^s\sim p(x)} H\big(D\big(F\big(x_i^s\big)\big), \tilde{\mathbf{y}}_f^s\big) + \mathbb{E}_{x_j^t\sim q(x)} H\big(D\big(F\big(x_j^t\big)\big), \tilde{\mathbf{y}}_f^t\big). \quad (9)$$

The label $\tilde{\mathbf{y}}_f^s$ of source instance x_i^s and the label $\tilde{\mathbf{y}}_f^t$ of target instance x_j^t are defined by

$$\tilde{\mathbf{y}}_f^s = \left[\overbrace{0, \cdots\cdots\cdots\cdots, 0}^{K}, 1\right], \quad \big(x_i^s, y_i^s\big) \in \mathcal{D}_s$$
$$\tilde{\mathbf{y}}_f^t = \left[\underbrace{\overbrace{0, \ldots, 0, 1}^{K}, 0, \ldots, 0, 0}_{l}, 0\right], \quad x_j^t \in \mathcal{D}_t,\ \hat{y}_j^t = l \quad (10)$$

where $\hat{y}_j^t$ is the pseudoclass label of the target instance x_j^t predicted by the classifier C. The feature extractor F aims to confuse D, i.e., makes D classify target instances into K source classes and assign source instances into the target domain. Hence, the last element of $\tilde{\mathbf{y}}_f^s$ is 1, which denotes that D should classify the source instance x_i^s as coming from the target domain. The first K elements of target instance label $\tilde{\mathbf{y}}_f^t$ are the pseudoclass label of target instance x_j^t in the one-hot encoding form since F expects D to classify the target instance x_j^t into K source classes. Considering the intent of F, when optimizing F by (9), $\tilde{\mathbf{y}}_f^t$ contains the target class information, while $\tilde{\mathbf{y}}_f^s$ does not, as defined in (10).

With the transferable features learned by the adversarial training of D and F, an adaptive classifier C is trained by minimizing the following source risk:

$$\min_{F,C} \mathcal{L}_c(F, C) = \mathbb{E}_{x_i^s\sim p(x)} \left[-\sum_{k=1}^{K} \mathbb{1}_{k=y_i^s} \log C\big(F\big(x_i^s\big)\big)\right] \quad (11)$$

where y_i^s is the class label of instance x_i^s, and K is the number of source classes, i.e., $K = |\mathcal{Y}_s|$. $\mathbb{1}_{k=y_i^s}$ means that if $k = y_i^s$, the value of $\mathbb{1}_{k=y_i^s}$ is 1 and otherwise is 0.

The overall optimization problem of domain adversarial learning is defined as

$$\min_{F,C,D} \mathcal{L}(F, C, D) = \mathcal{L}_c(F, C) + \mathcal{L}_y(F, D). \quad (12)$$

The feature extractor F and the domain classifier D are trained in an adversarial manner with different label values of instances by minimizing the adversarial loss $\mathcal{L}_y(F, D)$.

E. Relevance Metric

The relevance metric function $\varphi(x)$ measures the relevance of an instance x to the target domain, which is based on the domain classifier D and the classifier C.

1) Instance-Level Relevance Measured by D: If the source instance is likely to be assigned into the target domain by the domain classifier D, the relevance of this instance to the target domain is high. The last element of the output of D is denoted as $D(\cdot)_d$. The higher the $D(F(x_i^s))_d$ is, the more relevant the source instance x_i^s is to the target domain, and the more likely x_i^s belongs to the shared classes.

2) Class-Level Relevance Measured by C: Since the target classes and the outlier classes have no overlap, the target data have a low probability of being assigned to the outlier classes. Therefore, we use the predicted class distribution of the target data to compute the relevance of source classes to the target domain, denoted as $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_K] \in \mathbb{R}^K$, where μ_l represents the relevance of the lth source class to the target domain. The higher the μ_l is, the more relevant the lth source class is to the target domain, and the more likely the lth source class is in the shared classes. We compute $\boldsymbol{\mu}$ by

$$\boldsymbol{\mu} = \frac{1}{N_t} \sum_{j=0}^{N_t} C\left(F\left(x_j^t\right)\right), \quad x_j^t \in \mathcal{D}_t \tag{13}$$

and normalize it by $\boldsymbol{\mu} = \boldsymbol{\mu}/(\max(\boldsymbol{\mu}))$.

The instance-level and the class-level relevance represent the relevance of the instance x_i^s to the target domain from different aspects. The bigger the values of the instance-level relevance $D(F(x_i^s))_d$ and the class-level relevance $\mu_{y_i^s}$ are the more relevant x_i^s with class label y_i^s is to the target domain. Thus, it is a natural way to compute the product of $D(F(x_i^s))_d$ and $\mu_{y_i^s}$ to evaluate the relevance of x_i^s to the target domain. The relevance metric function $\varphi(x_i^s)$ is given by

$$\varphi\left(x_i^s\right) = \mu_{y_i^s} D\left(F\left(x_i^s\right)\right)_d. \tag{14}$$

F. Training

The ϵ-greedy strategy [61] is used to balance the exploration and exploitation of the agent. During the training at time t, the policy described by (2) is refined as

$$a_t = \begin{cases} \max_a Q(s_t, a), & \text{if } \lambda \geq \epsilon \\ a', & \text{otherwise} \end{cases} \tag{15}$$

where a' represents an action randomly selected from the action set A, λ is a random number drawn from [0, 1], and ϵ represents the probability of performing exploration, decaying with iterations. When $\lambda \geq \epsilon$, the agent takes an action by the action policies described as (2) to maximize the accumulated rewards. Otherwise, the agent performs exploration by taking a random action a' from the action set A, which can expand the solution space and avoid falling into a locally optimal solution.

The training of DQN requires data to be independent and identically distributed. However, the samples obtained in the training process are strongly correlated sequentially and do not satisfy the independent and identically distributed condition. Therefore, an experience replay strategy [62] is exploited to break the correlation between samples by storing-resampling and makes the network converge faster and more stable. Therefore, the agent can learn from various experiences in the long run.

The training of DQN has three stages: observing, exploring and training. During the observing stage, the experience of agent (s_t, a_t, s_{t+1}, r_t) is stored into an experience pool M and DQN is not updated due to lack of experience. During the exploring state, DQN is updated by (4) using randomly sampled experiences from the experience pool. In this stage, the agent explores environments to widen the solution space with a probability ϵ and ϵ decays from an initial high to a final low value. During the training stage, the agent explores environments with a low probability of ϵ.

The training process of the DARL is summarized in Algorithm 1.

Algorithm 1 DARL

Input: The source domain $\mathcal{D}_s$ and target domain $\mathcal{D}_t$
Output: The optimal F, C.
Pre-train
1: F and C with $\mathcal{D}_s$ by Eq.(11);
2: Initialize the experience pool $M = \varnothing$;
3: **while** not converge **do**
4: Initialize $\mathcal{D}_c$, $\mathcal{D}_e$ and generate the state $\boldsymbol{s}_0$ with $\mathcal{D}_c$;
5: **repeat**
6: Take an action a_t using the policy Eq.(15);
7: Compute the reward R_t of a_t by Eq.(3);
8: Update $\mathcal{D}_c$, $\mathcal{D}_e$ and state;
9: Insert experience $(\boldsymbol{s}_t, a_t, \boldsymbol{s}_{t+1}, R_t)$ into M;
10: **if** In *exploring* and *training* stage **then**
11: Sample experiences from M to update deep Q-learning network by Eq.(4);
12: **end if**
13: **until** Terminated, *i.e.,* $R_t < 0$
14: Update C, F, D with $\mathcal{D}_e$ and $\mathcal{D}_t$ by Eq.(12).
15: **end while**

G. Remarks

Most existing methods of partial domain adaptation utilize pseudo labels to weigh source instances in a straightforward manner [33]–[36]. Compared with the pseudolabeling methods, our DARL formulates the selections as a Markov decision process and applies the reinforcement learning paradigm to automatically learn policies for selecting source instances. The advantages of using reinforcement learning are as follows. On the one hand, reinforcement learning does not only make use of the prediction information but also explores in a wider space to find better solutions. Since the agent can take actions of small Q-values with a certain probability, i.e., *the exploration ability of reinforcement learning*, DARL can widen the solution space and jump out the local optimum, while the pseudolabeling methods only rely on the relevance of source instances to the target domain, and thus, easily trapping in the local optimum. For example, for the instances with low relevance but belonging to the shared classes, DARL can search them in a wide space owing to the superior exploration ability of the agent, while the pseudolabeling method will directly filter them out due to the low relevance. On the other hand, the selection strategy in the DARL is optimized via a sequential decision process at the set level with the guidance of the accumulated rewards; therefore, it can be more accurate compared with selecting based on pseudolabels at the instance level. In other words, pseudolabeling-based methods only consider the immediate rewards, i.e., the relevance of the selected instance to the target domain, when making the selection decision. For example, for the instances with high

relevance but not belonging to the shared classes, DARL will not select them according to future low rewards, while pseudolabeling methods will select them according to the immediate high rewards.

IV. Experiments

To evaluate the effectiveness of our method, we conduct 38 tasks of partial domain adaptation on eight publicly available data sets with three base networks.

A. Data Sets

The **Office + Caltech-10** data set [63] has four domains: Amazon, DSLR, Webcam, and Caltech, where the domains of Amazon, DSLR, and Webcam come from Office-31 [64] data set, and the Caltech domain comes from the Caltech-256 [65] data set. This data set has ten classes, which are the common classes between the Office-31 [64] and Caltech-256 [65] data sets. Hence, the Amazon, DSLR, Webcam, and Caltech domains in the Office + Caltech-10 data set are denoted as A10, D10, W10, and C10, respectively. The first five common classes of Amazon, DSLR, Webcam, and Caltech are denoted as A5, D5, W5, and C5, respectively. Following [33], one domain from A10, D10, W10, and C10 are used as the source domain, and one domain from A5, D5, W5, and C5 are used as the target domain. Therefore, there are 12 transfer tasks: C10 $\rightarrow$ A5, C10 $\rightarrow$ W5, C10 $\rightarrow$ D5, A10 $\rightarrow$ C5, A10 $\rightarrow$ W5, A10 $\rightarrow$ D5, W10 $\rightarrow$ C5, W10 $\rightarrow$ A5, W10 $\rightarrow$ D5, D10 $\rightarrow$ C5, D10 $\rightarrow$ A5, and D10 $\rightarrow$ W5.

The **Office-31** data set [64] has 31 classes of 4652 images, including three different domains: Amazon (A31), DSLR (D31), and Webcam (W31). The Amazon domain contains 2817 images sampled from online merchants (www.amazon.com), and each class has 90 images on average. The DSLR domain contains 498 high-resolution images taken by a digital SLR camera. The Webcam domain contains 795 low-resolution images taken by a Web camera. Following [34], one domain from A31, D31, and W31 in the Office-31 data set are used as the source domain, and one domain from A10, D10, and W10 in the Office + Caltech-10 data set are used as the target domain. Hence, there are six transfer tasks: A31 $\rightarrow$ W10, D31 $\rightarrow$ W10, W31 $\rightarrow$ D10, A31 $\rightarrow$ D10, D31 $\rightarrow$ A10, and W31 $\rightarrow$ A10.

The **Office-Home** data set [66] is an object recognition data set and has around 15 500 images. These images come from four domains: Artistic, Clipart, Product, and Real-World, and each domain has images of 65 classes. We denote the Artistic, Clipart, Product, and Real-World domains as Ar-65, Cl-65, Pr-65, and Rw-65, respectively. The first 25 classes in alphabetical order of the four domains are denoted as Ar-25, Cl-25, Pr-25, and Rw-25, respectively. Following [36], we use one domain from Ar-65, Cl-65, Pr-65, and Rw-65 as the source domain and one domain from Ar-25, Cl-25, Pr-25, and Rw-25 as the target domain. Therefore, there are 12 transfer tasks: Ar-65$\rightarrow$ Cl-25, Ar-65 $\rightarrow$ Pr-25, Ar-65 $\rightarrow$ Rw-25, Cl-65 $\rightarrow$ Ar-25, Cl-65 $\rightarrow$ Pr-25, Cl-65 $\rightarrow$ Rw-25, Pr-65 $\rightarrow$ Ar-25, Pr-65 $\rightarrow$ Cl-25, Pr-65 $\rightarrow$ Rw-25, Rw-65 $\rightarrow$ Ar-25, Rw-65 $\rightarrow$ Cl-25, and Rw-65 $\rightarrow$ Pr-25.

The **Caltech-Office** data set is constructed by the Caltech-256 and Office-31 data sets, which is a large scale data set. The Caltech-256 data set [65] is a standard data set for object recognition and consists of 256 classes of objects with 30 607 images. Each class has at least 80 images that are sampled from Google and PicSearch. The Caltech-256 data set with 256 classes is used as the source domain, denoted as C256, and one domain from A10, D10, and W10 in the Office + Caltech-10 data set are used as the target domain. Thus, there are three transfer tasks: C256 $\rightarrow$ W10, C256 $\rightarrow$ A10, and C256 $\rightarrow$ D10.

The **VisDA2017** data set [67] is a large-scale data set for domain adaptation, which has two domains: the synthetic domain and the real domain. The two domains have over 280 000 images with 12 classes in total, denoted as S12 and R12. The first six classes of the Synthetic and Real domains are denoted as S6 and R6. Following [35], we use one domain from S12 and R12 as the source domain and one domain from S6 and R6 as the target domain. Therefore, there are two transfer tasks: S12 $\rightarrow$ R6 and R12 $\rightarrow$ S6.

The digit data sets consist of three data sets: the **MNIST** data set [69], the **SVHN** data set [70], and the **USPS** data set [71]. The MNIST data set consists of handwritten digit images of size 28×28. The SVHN data set is a real-world image data set obtained from house numbers in Google Street View images. Images in the SVHN data set are scaled to 32×32 pixels. The USPS data set includes handwritten digit images of size 16×16. All data sets have 10 classes (0–9), denoted as M10, S10, and U10. The first five classes (0–4) of the MNIST and USPS data sets are denoted as M5 and U5. Following [23], [31], we construct three common domain adaptation transfer tasks: S10 $\rightarrow$ M5, M10 $\rightarrow$ U5, and U10 $\rightarrow$ M5.

B. Baseline Methods

We compare our method with the existing state-of-the-art methods, including partial domain adaptation methods (i.e., Selective Adversarial Network (SAN) [34], Importance Weighted Adversarial Network (IWAN) [33], Partial Adversarial Domain Adaptation (PADA) [35], ETN [36], and Reinforced Transfer Network (RTNet) [72]) and standard domain adaptation methods with the shared label space assumption (i.e., Deep Adaptation Network (DAN) [7], Domain Adversarial Neural Network (DANN) [31], Residual Transfer Network (RTN) [8], and ADDA [23]).

C. Implementation Details

We conduct experiments with three base networks: AlexNet, ResNet-50 and LeNet. With AlexNet as the base network, we fine-tune the AlexNet model pretrained on the ImageNet data set, following [34]. Concretely, the feature extractor F is obtained by removing the *fc8* layer of AlexNet and adding a bottleneck layer with 256 units on *fc7*. The classifier C is built on the bottleneck layer with one *fc* layer (256 $\rightarrow$ the number of source classes). We fine-tune the *conv5*, *fc6*, and *fc7* layers of F, and train the bottleneck layer of F and the classifier C. The bottleneck layer of F and C are trained from scratch, whose learning rate is set to be $10\times$ that of the other

TABLE I
ACCURACY (%) ON THE OFFICE + CALTECH-10 DATA SET WITH ALEXNET AS BASE NETWORK

Method	C10 →A5	C10 → W5	C10 → D5	A10 → C5	A10 → W5	A10 → D5
AlexNet [68]	94.65	90.37	97.06	85.79	81.48	95.59
DANN [31]	91.86	82.22	83.82	77.57	65.93	80.88
RTN [8]	91.86	93.33	80.88	80.99	69.63	70.59
ADDA [23]	93.15	94.07	97.06	85.27	87.41	89.71
IWAN [33]	94.22	97.78	98.53	89.90	87.41	88.24
DARL	**96.36**	**98.52**	**100.00**	**92.47**	**88.89**	**100.00**

Method	W10 → C5	W10 → A5	W10 → D5	D10 → C5	D10 → A5	D10 → W5	Avg
AlexNet [68]	76.37	87.79	**100.00**	80.99	89.94	97.04	89.76
DANN [31]	72.60	80.30	95.59	69.35	77.09	80.74	79.83
RTN [8]	59.08	74.73	**100.00**	59.08	70.02	91.11	78.44
ADDA [23]	86.82	92.08	**100.00**	89.90	93.79	98.52	92.31
IWAN [33]	90.24	95.29	**100.00**	91.61	94.43	98.52	93.85
DARL	**93.15**	**96.15**	**100.00**	**92.64**	**95.93**	**99.26**	**96.11**

layers [34]. The domain classifier D is built with three *fc* layers (1024 → 1024 → the number of source classes + 1). DQN has four *fc* layers (1024 → 512 → 256 → action number). With ResNet-50 as base network, we fine-tune the ResNet-50 model pretrained on the ImageNet data set, following [35]. The feature extractor F is obtained by removing the *fc* layer of ResNet-50 and adding a bottleneck layer with 256 units on *res5c*. We fine-tune the last block of ResNet-50 and train the bottleneck layer of F and the classifier C. The learning rates of the bottleneck layer of F and C are set to be 10 times of the other layers. The architectures of the domain classifier and DQN are the same as AlexNet. For digit image classification, we conduct an experiment in two settings. In the first setting, the original LeNet [69] is trained from scratch. All images are rescaled to 32×32 pixels, and the grayscale images are converted to the RGB images by copying the image channels. In the second setting, following [72], we conduct experiments on the modified LeNet (denoted as "LeNet-m") and convert all the RGB images to the grayscale images. The domain classifier D is built with three *fc* layers (500 → 500 → the number of source classes + 1). The architecture of DQN is the same as AlexNet and ResNet-50.

We implement our method with the TensorFlow framework. An AdamOptimizer [73] is used to optimize the whole network. The learning rates of our DQN and domain adversarial learning network are both set to 0.0001 with 0.9 and 0.5 as the momentum, respectively. The discount factor γ in (5) is set to 0.9. The size of the experience pool M is set to 2000. When updating DQN, we randomly sample 32 experiences from the experience pool M to compute gradients via (4). The size of the initial candidate set is set to 16. The batch size of the domain adversarial learning network is set to 32 with 16 source images and 16 target images. For transfer tasks on Office + Caltech-10 and Office-31 data sets, we train DARL with 2000 iterations. For transfer tasks on Caltech-Office, VisDA2017, Office-Home, and digit data sets, we train DARL with 5000 iterations. The exploring probability ϵ decays from 1 to 0 at the exploring stage of DQN. The threshold τ is set to 0.3, 0.1, and 0.3 for AlexNet, ResNet-50, and LeNet as base network, respectively.

D. Results

For fair comparison, we directly use the reported results of all the compared methods in their original articles and implement our method with the same base network as that in the compared methods.

1) Results on the Office + Caltech-10 Data Set: Table I reports the classification accuracy of our DARL and other domain adaptation methods on the Office + Caltech-10 data set. From the results, we have the following observations.

1) DARL outperforms all the compared methods on all the transfer tasks, clearly demonstrating the benefit of reinforcement learning on selecting the right source instances for partial domain adaptation. In particular, DARL achieves higher classification accuracy on the hard tasks, such as C10 → W5, A10 → C5, and W10 → C5, where the distributions of the source and target domains are largely different.
2) The performances of the standard domain adaptation methods with the shared label space assumption degrade when the source and target label spaces are different. For example, DANN and RTN work worse than AlexNet, which further proves the existence of the negative transfer caused by the mismatch of the label spaces.
3) DARL and ADDA achieve much better results than DANN and perform more robust to the different label spaces. The possible reason is that DARL and ADDA both preserve the class information of the source domain well when performing the domain adversarial learning, while the class information in DANN is poorly preserved since the feature extractor is trained with classification and adversarial loss at the same time. This further implies that the class information in partial domain adaptation is important and should be preserved during learning domain-invariant features.

2) Results on the Office-31 Data Set: Table II shows the classification accuracy of different methods on the Office-31 data set with AlexNet as the base network. Our DARL achieves state-of-the-art or comparable results on six transfer tasks. Transfer tasks on the Office-31 data set are more challenging than those on the Office + Caltech-10 data set since the number of outlier classes of the Office-31 data set is larger than that of the Office + Caltech-10 data set. In this situation, the standard domain adaptation methods do not improve or even hurt the performance of the target domain. For example, DANN is 15.78% lower than AlexNet on the Office-31 data set and 9.93% lower than AlexNet on the

TABLE II
ACCURACY(%) ON THE OFFICE-31 DATA SET WITH ALEXNET AS BASE NETWORK

Method	A31 → W10	D31→ W10	W31 → D10	A31 → D10	D31 → A10	W31 →A10	Avg
AlexNet [68]	59.32	96.27	98.73	73.25	70.77	66.08	77.40
DAN [7]	56.52	71.86	86.78	51.86	50.42	52.29	61.62
DANN [31]	56.95	75.59	89.17	57.32	57.62	63.15	66.64
RTN [8]	66.78	86.77	99.36	70.06	73.52	76.41	78.82
ADDA [23]	70.68	96.44	98.65	72.90	74.26	75.56	81.42
IWAN [33]	76.27	98.98	**100.00**	78.98	89.46	81.73	87.57
SAN [34]	**80.02**	98.64	**100.00**	81.28	80.58	83.09	87.27
DARL	77.97	**100.00**	**100.00**	**83.44**	**93.01**	**87.47**	**90.32**

TABLE III
ACCURACY(%) ON THE OFFICE-31 DATA SET WITH RESNET-50 AS BASE NETWORK

Method	A31 → W10	D31→ W10	W31 → D10	A31 → D10	D31 → A10	W31 →A10	Avg
ResNet-50 [74]	76.61	94.58	98.09	84.08	72.86	75.37	83.60
DAN [7]	59.32	73.90	90.45	61.78	74.95	67.64	71.34
DANN [31]	73.56	96.27	98.73	81.53	82.78	86.12	86.50
RTN [8]	78.98	93.22	85.35	77.07	89.25	89.46	85.56
ADDA [23]	75.67	95.38	99.85	83.41	83.62	84.25	87.03
IWAN [33]	89.15	99.32	99.36	90.45	95.62	94.26	94.69
SAN [34]	93.90	99.32	99.36	94.27	94.15	88.73	94.96
PADA [35]	86.54	99.32	**100.00**	82.17	92.69	**95.41**	92.69
ETN [36]	94.52	**100.00**	**100.00**	95.03	**96.21**	94.64	96.73
DARL	**94.58**	99.66	**100.00**	**98.73**	94.57	94.26	**96.97**

TABLE IV
ACCURACY (%) ON THE OFFICE-HOME DATA SET WITH RESNET-50 AS BASE NETWORK

Method	Ar-65 → Cl-25	Ar-65 → Pr-25	Ar-65 → Rw-25	Cl-65 → Ar-25	Cl-65→ Pr-25	Cl-65 → Rw-25
ResNet-50 [74]	44.00	62.80	74.27	55.37	54.23	61.40
DANN [31]	43.76	67.90	77.47	63.73	58.99	67.59
ADDA [23]	45.23	68.79	79.21	64.56	60.01	68.29
RTN [8]	49.31	57.70	80.07	63.54	63.47	73.38
IWAN [33]	53.94	54.45	78.12	61.31	47.95	63.32
SAN [34]	44.42	68.68	74.60	67.49	64.99	77.80
PADA [35]	51.95	67.00	78.74	52.16	53.78	59.03
ETN [36]	**59.24**	77.03	79.54	62.92	65.73	75.01
DARL	55.31	**80.73**	**86.36**	**67.93**	**66.16**	**78.52**

Method	Pr-65 → Ar-25	Pr-65 → Cl-25	Pr-65 → Rw-25	Rw-65 → Ar-25	Rw-65 → Cl-25	Rw-65 → Pr-25	Avg
ResNet-50 [74]	56.29	38.69	75.54	63.09	42.81	74.62	58.59
DANN [31]	56.84	37.07	76.37	69.15	44.30	77.48	61.72
ADDA [23]	57.56	38.89	77.45	70.28	45.23	78.32	62.82
RTN [8]	65.11	41.73	75.32	63.18	43.57	80.50	63.07
IWAN [33]	54.17	52.02	81.28	76.46	56.75	82.90	63.56
SAN [34]	59.78	44.72	80.07	72.18	50.21	78.66	65.30
PADA [35]	52.61	43.22	78.79	73.73	56.60	77.09	62.06
ETN [36]	68.29	**55.37**	84.37	75.72	**57.66**	84.54	70.45
DARL	**68.74**	50.93	**87.74**	**79.45**	57.19	**85.60**	**72.06**

Office + Caltech-10 data set, which shows that the negative transfer is more serious for the Office-31 data set. For partial domain adaptation methods, SAN and IWAN gain 10.17% and 9.87% than AlexNet, respectively, and our DARL outperforms SAN and IWAN, clearly validating the benefit of modeling the source instance selection as a Markov decision process.

To evaluate the performance of our DARL on different base networks, we conduct experiments on the Office-31 data set with ResNet-50 as the base network. The classification accuracies of different methods are shown in Table III. With ResNet-50 as base network, the performance on the Office-31 data set approaches saturation, and the potential improvement is limited due to the small domain gap and the large intra-category discrepancy. In this situation, DARL also achieves better results than the state-of-the-art method (ETN), further validating the effectiveness of DARL. For both AlexNet and ResNet-50 as base networks, DARL consistently achieves better results than other methods and performs superior ability in generalization.

3) Results on the Office-Home Data Set: Table IV shows the classification accuracy of our DARL and other compared methods on the Office-Home data set with ResNet-50 as the base network. For transfer tasks on the Office-Home data set, there are 40 outlier source classes, and the negative transfer is more serious than that on the Office + Caltech-10 and Office-31 data sets. From the results, with the serious negative transfer, our DARL still outperforms all compared methods and achieves an average improvement of 1.61% over the state-of-the-art method (ETN), which demonstrates the excellent learning ability of the agent for exploring selection

TABLE V
ACCURACY (%) ON THE CALTECH-OFFICE DATA SET WITH ALEXNET AS BASE NETWORK

Method	C256 → W10	C256→A10	C256 →D10	Avg
AlexNet [68]	62.37	78.39	65.61	68.79
DAN [7]	42.37	70.75	47.04	53.39
DANN [31]	54.57	72.86	57.96	61.80
RTN [8]	71.02	81.32	62.35	71.56
ADDA [23]	73.66	78.35	74.80	75.60
IWAN [33]	86.10	82.25	84.08	84.14
SAN [34]	**88.33**	83.82	85.35	85.83
DARL	88.14	**92.59**	**91.72**	**90.82**

TABLE VI
ACCURACY (%) ON THE VISDA2017 DATA SET WITH RESNET-50 AS BASE NETWORK

Method	R12→S6	S12→R6	Avg
ResNet-50 [74]	64.28	45.26	54.77
DAN [7]	68.35	47.60	57.98
DANN [31]	73.84	51.01	62.43
RTN [8]	72.93	50.04	61.49
PADA [35]	76.50	53.53	65.01
DARL	**79.94**	**67.77**	**73.86**

policies and the effectiveness of our DARL for handling more challenging partial domain adaptation.

4) Results on the Caltech-Office Data Set: Table V shows the classification accuracy of compared methods and our DARL on the Caltech-Office data set. The three transfer tasks on this data set are more difficult than those on the Office + Caltech-10 and Office-31 data sets since the proportion of outlier classes in the total classes on the Caltech-Office data set is higher than that on the others. For those difficult transfer tasks, DARL also performs well and outperforms SAN and IWAN with the gains of 4.65% and 6.34% on average, respectively, verifying that our DARL is excellent in handling more challenging partial domain adaptation.

5) Results on the VisDA2017 Data Set: Table VI reports the classification accuracy of our DARL and other domain adaptation methods on the VisDA2017 data set with ResNet-50 as the base network. Transfer tasks on VisDA2017 are more challenging than others since the large scale of data set and the considerable domain shift between synthetic data and real data. In this situation, our DARL still outperforms all compared methods and achieves 19.09% improvement over ResNet-50 on average. Compared with PADA [35], DARL achieves 8.85% improvement on average. The reasons are as follows: 1) DARL optimizes the selection decision on the set level, while PADA optimizes that on the instance level. In other words, DARL makes selection decisions according to accumulated rewards, while PADA only considers the immediate rewards when making decisions. 2) DARL utilizes both domain prediction probabilities and class probabilities to measure the relevance of instances to the target domain, while PADA only applies class probabilities to weigh source data.

6) Results on the Digit Data Sets: Table VII shows the classification accuracies of DARL and the compared methods on the digit data sets. The first and second parts show the results with LeNet and LeNet-m as the base network, respectively. From results, we can observe that DARL outperforms all compared domain adaption methods with the same base network, clearly demonstrating the effectiveness of our method.

TABLE VII
ACCURACY (%) ON THE DIGIT DATA SETS

Base Net	Method	S10→M5	M10→U5	U10→M5	Avg
LeNet	LeNet [69]	64.22	70.88	88.49	74.53
	DARL	**73.57**	**75.97**	**92.34**	**80.63**
LeNet-m	LeNet-m	82.75	-	90.06	86.41
	PADA [35]	90.40	-	97.40	93.90
	ETN [36]	93.60	-	96.50	95.05
	RTNet [72]	97.20	-	98.50	97.85
	DARL	**97.40**	**-**	**98.86**	**98.13**

E. Ablation Studies

1) Effect of Reinforcement Learning: To go deeper with the effect of reinforcement learning, we compare our method with two variations: without reinforcement learning (denoted as DA) and replacing reinforcement learning with selecting source instances directly based on the pseudo domain and class labels (denoted as DA-PLs). In DA-PLs, source instances are directly selected by the relevance metric function defined in (14). Specifically, the source instances of $\varphi(x_i^s) > \tau$ are selected to update the domain adversarial learning network, and we set the same threshold τ for DA-PLs as DARL. We show the results of DA and DA-PLs on all the data sets in Tables VIII–XI, respectively.

It is interesting to observe that:

1) Compared with DA, our DARL outperforms DA in all the transfer tasks, especially on the difficult tasks, such as D31 → A10 and C256 → D10. For example, our DARL achieves an improvement of 21.77% on average over DA on the Caltech-Office data set. The possible reason is that DA matches all the source data with the target domain where the negative transfer caused by outlier classes hurts the performance on the target domain, while DARL refines the source domain via selection actions taken by the agent and only matches the optimal subset of source domain with the target domain for facilitating the positive transfer.
2) DA performs better than some standard domain adaptation methods with the shared label space assumption. For example, DA gains a 7.25% improvement over DANN on the Caltech-Office data set, probably due to that the domain classifier of DA retains the class information of the source domain whereas the domain classifier of DANN does not consider the class information.
3) Our DARL works better than DA-PLs on all transfer tasks, demonstrating that reinforcement learning is more powerful than selecting with pseudo labels directly. There are two possible reasons: 1) the learned selection in our DARL is a sequential decision process since the output of DQN represents both the immediate and future rewards, while the selection in DA-PLs only considers the immediate rewards. For the instances with high relevance but not belonging to the shared classes,

TABLE VIII

ACCURACY (%) OF ABLATION EXPERIMENTS ON THE OFFICE + CALTECH-10 DATA SET WITH ALEXNET AS BASE NETWORK

Method	C10 →A5	C10 → W5	C10 → D5	A10 → C5	A10 → W5	A10 → D5
DA	94.86	97.04	100.00	86.13	88.15	97.06
DA-PLs	95.29	91.85	98.53	90.24	85.93	95.59
DARL-stanD	95.93	97.78	100.00	90.07	88.89	98.53
DARL	96.36	98.52	100.00	92.47	88.89	100.00

Method	W10 → C5	W10 → A5	W10 → D5	D10 → C5	D10 → A5	D10 → W5	Avg
DA	85.96	93.36	100.00	85.79	89.08	99.26	93.06
DA-PLs	91.27	95.29	100.00	80.99	89.94	98.52	92.79
DARL-stanD	92.47	95.93	100.00	91.27	89.94	98.52	95.94
DARL	93.15	96.15	100.00	92.64	95.93	99.26	96.11

TABLE IX

ACCURACY (%) OF ABLATION EXPERIMENTS ON THE OFFICE-31 DATA SET WITH ALEXNET AS BASE NETWORK

Method	A31 → W10	D31→ W10	W31 → D10	A31 → D10	D31 → A10	W31 →A10	Avg
DA	60.00	97.63	98.09	75.08	81.52	78.50	81.92
DA-PLs	67.46	98.89	99.36	73.98	90.71	81.94	85.39
DARL-stanD	77.63	99.32	99.36	80.25	84.34	79.33	86.71
DARL	77.97	100.00	100.00	83.44	93.01	87.47	90.32

TABLE X

ACCURACY (%) OF ABLATION EXPERIMENTS ON THE OFFICE-31 DATA SET WITH RESNET-50 AS BASE NETWORK

Method	A31 → W10	D31→ W10	W31 → D10	A31 → D10	D31 → A10	W31 →A10	Avg
DA	87.80	96.72	100.00	91.08	79.75	78.81	87.43
DA-PLs	81.69	98.31	100.00	95.94	91.96	81.11	90.33
DARL-stanD	93.90	98.64	100.00	92.99	84.24	83.30	92.18
DARL	94.58	99.66	100.00	98.73	94.57	94.26	96.97

TABLE XI

ACCURACY (%) OF ABLATION EXPERIMENTS ON THE CALTECH-OFFICE DATA SET WITH ALEXNET AS BASE NETWORK

Method	C256 → W10	C256→ A10	C256 → D10	Avg
DA	63.05	78.50	65.61	69.05
DA-PLs	83.73	92.28	87.26	87.76
DARL-stanD	85.76	90.19	89.81	88.57
DARL	88.14	92.59	91.72	90.82

DARL will not select them according to future rewards, while DA-PLs will select them according to the high immediate relevance. 2) Our DARL widens the solution space and jumps out the local optimum with superior exploration ability, while DA-PLs only relies on the relevance of source instances to the target domain, and thus, easily get trapped in a local optimum. For the instances with low relevance but belonging to the shared classes, our DARL can search them in a wide space, while DA-PLs will directly filter them out due to the low relevance.

2) Effect of Domain Classifier: In this article, we apply a domain classifier with a $K + 1$ way to consider both the class information and the domain information at the same time. To evaluate the effectiveness of this novel domain classifier, we compare our DARL with a variation: replacing our $K + 1$ way domain classifier with a standard domain classifier that only considers the domain information (denoted as DARL-stanD). We show the results of DARL-stanD on the Office + Caltech-10, Office-31, and Caltech-Office data sets in Tables VIII–XI, respectively. The improvement of DARL over DARL-stanD verifies that the class information is important in partial domain adaptation and should be preserved when performing domain adversarial learning.

F. Parameter Analysis

The threshold τ of reward function in (3) is an import factor. More experiments are conducted on the Office-31 data set to evaluate the effect of the threshold τ. We tune τ in range of $\{0.1, 0.2, 0.3, 0.4, 0.5\}$ and show the accuracy-threshold curves in Fig. 2(a), where the horizontal axis represents the value of τ in (3), the vertical axis represents the classification accuracy, and the different colors of curves denote different transfer tasks on the Office-31 data set. From the results, it is obvious that the accuracy of all transfer tasks first increases and then decreases as the threshold τ increases. Specifically, when τ is small, the accuracy is lower since some source instances in the outlier classes cannot be filtered out. When τ is large, the accuracy is also lower since some source instances in the shared classes are filtered out. When $\tau = 0.3$, DARL generally achieves best results for most transfer tasks. Moreover, it is worth noting that the trends of the accuracy-threshold curves for different transfer tasks are similar. Hence, we set $\tau = 0.3$ for other data sets using AlexNet as base network. For other base networks, we tune τ on one data set and apply this τ to other data sets.

G. Convergence Analysis

We study the test errors on the C256 → W10 transfer task on the Caltech-Office data set to evaluate the convergence

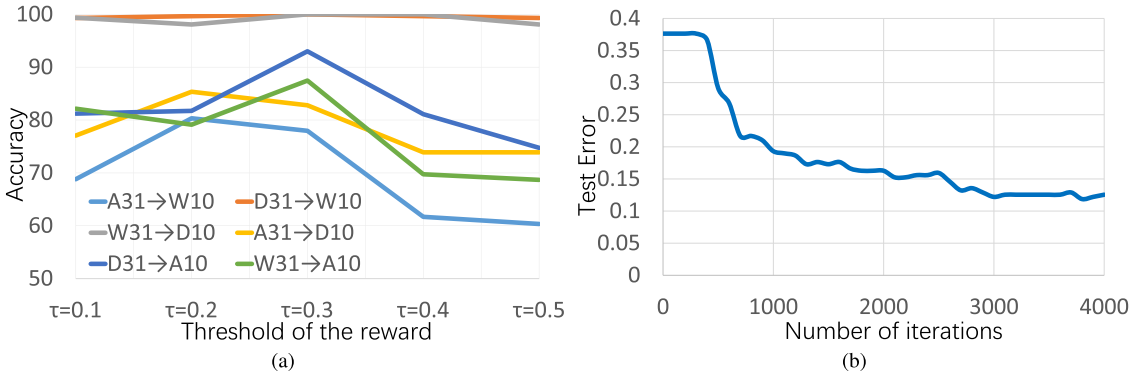


Fig. 2. Empirical analysis of DARL. (a) Shows the performances of different thresholds on the Office-31 data set. The horizontal axis represents the value of threshold τ in (3), and the vertical axis represents the classification accuracy on the target domain. We use six colors to denote six transfer tasks on the Office-31 data set. (b) Shows the test errors of the C256 → W10 transfer task at different training iterations on the Caltech-Office data set. The horizontal axis represents the number of training iterations, and the vertical axis represents the test error on the target domain (W10). (a) Accuracy versus threshold. (b) Test error versus iteration.

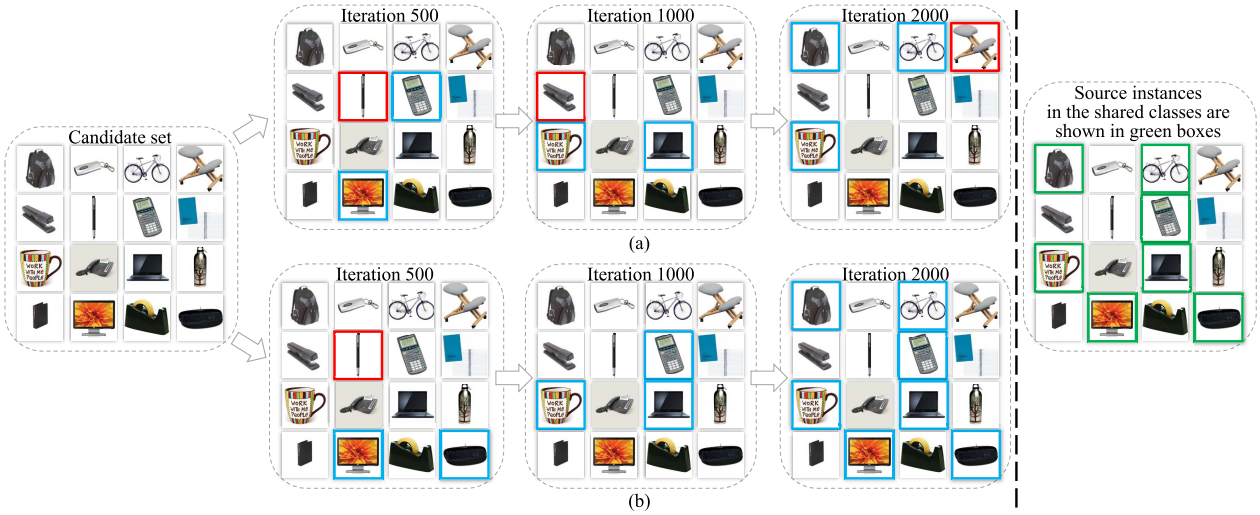


Fig. 3. Example of the selection process of DA-PLs and DARL on the A31 → D10 setting. The left image represents source instances in a candidate set, and the goal is to select outsource instances in the shared classes that are shown in green boxes of the right image. The source instances in the blue and red boxes represent the selected instances. Blue boxes indicate correctly selected instances, and red boxes indicate wrongly selected instances. (a) Source instances selected by DA-PLs. (b) Source instances selected by DARL.

performance of DARL. The curve of accuracy and test error on the target domain is shown in Fig. 2(b), where the horizontal axis represents the number of training iterations, and the vertical axis represents the test error on the target domain (W10). From the results, it can be observed that DARL can gradually converge to a low test error.

H. Qualitative Evaluation

To go deeper with the effectiveness of reinforcement learning for sample selection, we visualize the selection processes of DA-PLs and DARL on the A31 → D10 setting in Fig. 3, where we use blue boxes to denote correctly selected source instances and red boxes to denote wrongly selected source instances. From the results, we find that DARL works better than DA-PLs for selecting source instances in the shared classes. At the iteration of 500, the performances of DA-PLs and DARL are comparable. At the iteration of 1000, with the accumulation of experience, DARL can select more source instances in the shared classes than DA-PLs. At the iteration of 2000, DARL selects out all source instances in the shared classes, clearly demonstrating the effectiveness of reinforcement learning in sample selection.

I. Feature Visualization

Fig. 4 visualizes the features from the bottleneck layer in the feature extractor F of our DARL and AlexNet on the Caltech-Office data set. Due to a large number of source classes, we randomly sample ten outlier classes and ten shared classes from the source domain for each task. The visualization results of AlexNet and our DARL are shown in Fig. 4(a) and (b), respectively. In Fig. 4(a) and (b), the green samples represent source instances in the shared classes, the red samples represent source instances in the outlier classes, and the blue samples denote the target instances. From Fig. 4, we have the following observations. First, the distributions of the blue samples and the green samples are different in Fig. 4(a), demonstrating the existence of domain shift between the source and target domains. As shown in Fig. 4(b),

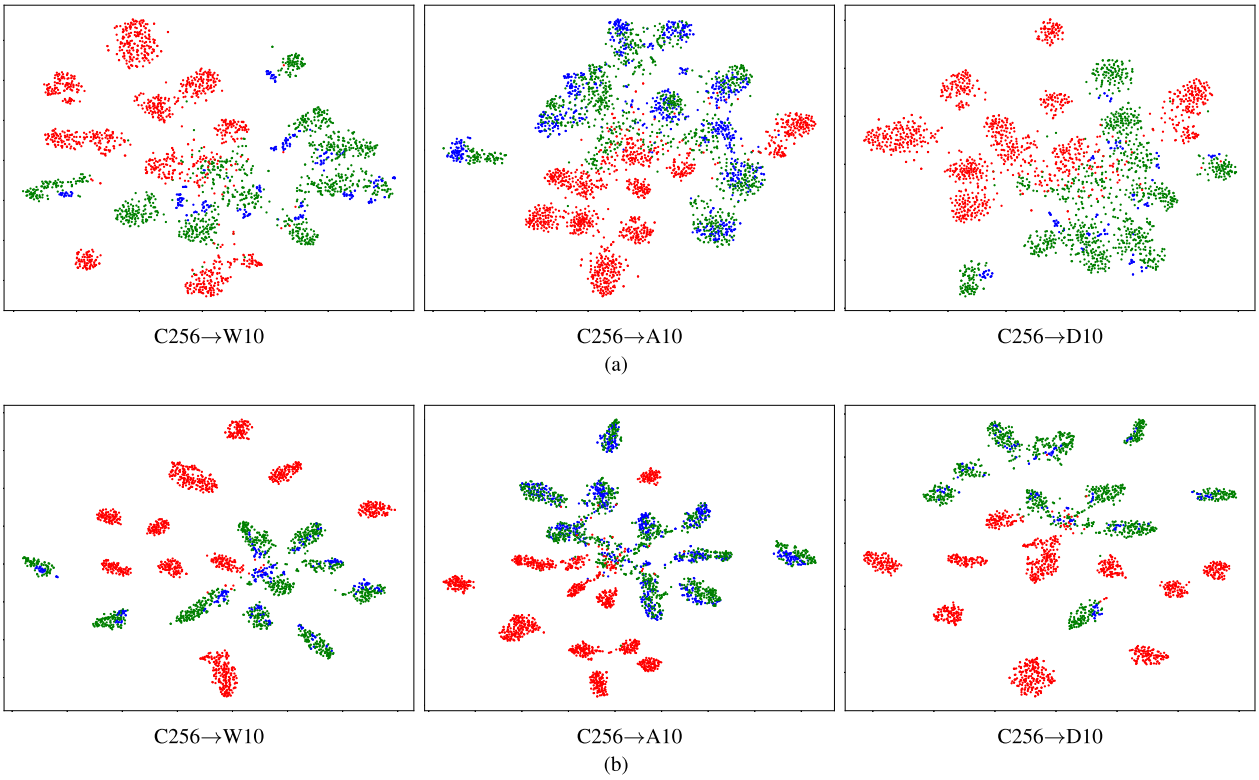


Fig. 4. Learned feature visualization of (a) AlexNet and our (b) DARL on the Caltech-Office data set using t-SNE. We use the green samples to represent the source instances in the shared classes, the red samples to represent the source instances in the outlier classes, and the blue samples to denote the target instances, respectively.

TABLE XII
COMPUTATION TIME (SECONDS) OF DARL ON DIFFIERENT DATA SETS WITH DIFFERENT BASE NETWORKS

Method	Office+Caltech-10	Office-31		Caltech-Office	Office-Home	VisDA2017	digits	Avg
	AlexNet	AlexNet	ResNet-50	AlexNet	ResNet-50	ResNet-50	LeNet	
Training time	655.35	701.48	1892.70	4163.38	5010.86	9553.92	337.57	3187.89
Test time	1.05	1.66	6.02	2.61	20.59	976.12	33.77	148.83

the blue samples are almost aligned with the green samples. This means that our DARL can effectively align the source instances in the shared classes with the target domain. Second, compared with the samples learned by AlexNet in Fig. 4(a), the samples of our DARL have smaller intraclass distance and larger interclass distance, which indicates that features learned by DARL are more discriminative than features extracted with AlexNet, probably due to that when performing domain adversarial learning, the class information of the source domain is enhanced with our $K + 1$ way domain classifier.

J. Efficiency Evaluation

We report the average training and test times of DARL on all evaluated data sets with different base networks in Table XII. All computations are performed on a computer with a single NVIDIA GeForce GTX Titan X GPU, an Intel Core i7-5930 K 3.50-GHz CPU, and a 64-GB RAM. Required memory of using ResNet-50, AlexNet, and LeNet as the base network is 6480, 2460, 923 Mb, respectively. For transfer tasks on the Office + Caltech-10 and Office-31 data sets, our DARL requires 2000 iterations for training. For transfer tasks on the Caltech-Office, VisDA2017, Office-Home, and digit data sets, our DARL requires 5000 iterations for training.

V. CONCLUSION

In this article, we have proposed to leverage a reinforcement learning framework coupled with domain adversarial learning for partial domain adaptation, where the source domain and the target domain have different label spaces. Specifically, two components are developed in our framework. On the one hand, the deep Q-learning component selects source instances in the shared classes to avoid the negative transfer. On the other hand, the domain adversarial learning component reduces the domain shift and provides effective rewards to the agent to promote the positive transfer. Those two components are jointly learned with an iterative optimization procedure. Extensive experiments on various benchmarks have demonstrated the effectiveness of our proposed solution.

REFERENCES

[1] Y. B. Kim, K. Stratos, and R. Sarikaya, "Frustratingly easy neural domain adaptation," in *Proc. Int. Conf. Comput. Linguistics (COLING)*, 2016, pp. 387–396.
[2] H. Daumé, III, and J. Jagarlamudi, "Domain adaptation for machine translation by mining unseen words," in *Proc. Assoc. Comput. Linguistics (ACL)*, 2011, pp. 407–412.
[3] M. Freitag and Y. Al-Onaizan, "Fast domain adaptation for neural machine translation," 2016, *arXiv:1612.06897*. [Online]. Available: http://arxiv.org/abs/1612.06897

[4] M. A. Sultan, J. Boyd-Graber, and T. Sumner, "Bayesian supervised domain adaptation for short text similarity," in *Proc. Assoc. Comput. Linguistics (ACL)*, 2016, pp. 927–936.

[5] B. Chen, C. Cherry, G. Foster, and S. Larkin, "Cost weighting for neural machine translation domain adaptation," in *Proc. 1st Workshop Neural Mach. Transl.*, 2017, pp. 40–46.

[6] R. Gopalan, R. Li, and R. Chellappa, "Domain adaptation for object recognition: An unsupervised approach," in *Proc. Int. Conf. Comput. Vis. (ICCV)*, Nov. 2011, pp. 999–1006.

[7] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 97–105.

[8] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Unsupervised domain adaptation with residual transfer networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 136–144.

[9] W.-S. Tseng, L.-K. Hsu, L.-W. Kang, and Y.-C.-F. Wang, "Cross-view action recognition via low-rank based domain adaptation," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2013, pp. 3244–3248.

[10] L. Niu, W. Li, and D. Xu, "Exploiting privileged information from Web data for action and event recognition," *Int. J. Comput. Vis.*, vol. 118, no. 2, pp. 130–150, Jun. 2016.

[11] L. Wei, S. Zhang, W. Gao, and Q. Tian, "Person transfer GAN to bridge domain gap for person re-identification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 79–88.

[12] W. Deng, L. Zheng, Q. Ye, G. Kang, Y. Yang, and J. Jiao, "Image-image domain adaptation with preserved self-similarity and domain-dissimilarity for person re-identification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 994–1003.

[13] J. Liu, B. Ni, Y. Yan, P. Zhou, S. Cheng, and J. Hu, "Pose transferrable person re-identification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 4099–4108.

[14] Z. Zhong, L. Zheng, Z. Zheng, S. Li, and Y. Yang, "Camera style adaptation for person re-identification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 5157–5166.

[15] J. Lv, W. Chen, Q. Li, and C. Yang, "Unsupervised cross-dataset person re-identification by transfer learning of spatial-temporal patterns," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 7948–7956.

[16] K. Lai and D. Fox, "Object recognition in 3D point clouds using Web data and domain adaptation," *Int. J. Robot. Res.*, vol. 29, no. 8, pp. 1019–1037, Jul. 2010.

[17] K. Sohn, S. Liu, G. Zhong, X. Yu, M.-H. Yang, and M. Chandraker, "Unsupervised domain adaptation for face recognition in unlabeled videos," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5917–5925.

[18] F. Nourbakhsh, E. Granger, and G. Fumera, "An extended sparse classification framework for domain adaptation in video surveillance," in *Proc. Asian Conf. Comput. Vis. (ACCV)*, 2016, pp. 360–376.

[19] F. Mueller *et al.*, "GANerated hands for real-time 3D hand tracking from monocular RGB," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 49–59.

[20] Y. Mansour, M. Mohri, and A. Rostamizadeh, "Domain adaptation: Learning bounds and algorithms," in *Proc. 22nd Conf. Learn. Theory (COLT)*, 2009, pp. 1–16.

[21] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," 2014, *arXiv:1412.3474*. [Online]. Available: http://arxiv.org/abs/1412.3474

[22] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 443–450.

[23] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7167–7176.

[24] Y. Ganin and V. S. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 1180–1189.

[25] Z. Pei, Z. Cao, M. Long, and J. Wang, "Multi-adversarial domain adaptation," in *Proc. Assoc. Adv. Artif. Intell. (AAAI)*, 2018, pp. 1941–3934.

[26] S. Sankaranarayanan, Y. Balaji, C. D. Castillo, and R. Chellappa, "Generate to adapt: Aligning domains using generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 8503–8512.

[27] Y. Li, N. Wang, J. Shi, X. Hou, and J. Liu, "Adaptive batch normalization for practical domain adaptation," *Pattern Recognit.*, vol. 80, pp. 109–117, Aug. 2018.

[28] F. M. Cariucci, L. Porzi, B. Caputo, E. Ricci, and S. R. Bulo, "AutoDIAL: Automatic domain alignment layers," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5067–5075.

[29] F. M. Carlucci, L. Porzi, B. Caputo, E. Ricci, and S. R. Bulò, "Just dial: Domain alignment layers for unsupervised domain adaptation," in *Proc. Int. Conf. Image Anal. Process. (ICIAP)*, 2017, pp. 357–369.

[30] M. Mancini, L. Porzi, S. R. Bulo, B. Caputo, and E. Ricci, "Boosting domain adaptation by discovering latent domains," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 3771–3780.

[31] Y. Ganin *et al.*, "Domain-adversarial training of neural networks," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2030–2096, 2016.

[32] J. Shen, Y. Qu, W. Zhang, and Y. Yu, "Wasserstein distance guided representation learning for domain adaptation," in *Proc. Assoc. Adv. Artif. Intell. (AAAI)*, 2018, pp. 4058–4065.

[33] J. Zhang, Z. Ding, W. Li, and P. Ogunbona, "Importance weighted adversarial nets for partial domain adaptation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 8156–8164.

[34] Z. Cao, M. Long, J. Wang, and M. I. Jordan, "Partial transfer learning with selective adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 2724–2732.

[35] Z. Cao, L. Ma, M. Long, and J. Wang, "Partial adversarial domain adaptation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 139–155.

[36] Z. Cao, K. You, M. Long, J. Wang, and Q. Yang, "Learning to transfer examples for partial domain adaptation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2985–2994.

[37] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, p. 1054, Sep. 1998.

[38] L.-J. Lin, "Reinforcement learning for robots using neural networks," School Comput. Sci., Carnegie-Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-93-103, 1993.

[39] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[40] V. Mnih *et al.*, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*. [Online]. Available: http://arxiv.org/abs/1312.5602

[41] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical sequence training for image captioning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7008–7024.

[42] T.-H. Chen, Y.-H. Liao, C.-Y. Chuang, W.-T. Hsu, J. Fu, and M. Sun, "Show, adapt and tell: Adversarial training of cross-domain image captioner," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 521–530.

[43] X. Wang, W. Chen, J. Wu, Y.-F. Wang, and W. Y. Wang, "Video captioning via hierarchical reinforcement learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4213–4222.

[44] L. Li and B. Gong, "End-to-end video captioning with multitask reinforcement learning," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2019, pp. 339–348.

[45] Z. Ren, X. Wang, N. Zhang, X. Lv, and L.-J. Li, "Deep reinforcement learning-based image captioning with embedding reward," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 290–298.

[46] S. Yeung, V. Ramanathan, O. Russakovsky, L. Shen, G. Mori, and L. Fei-Fei, "Learning to learn from noisy Web videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5154–5162.

[47] Y. Tang, Y. Tian, J. Lu, P. Li, and J. Zhou, "Deep progressive reinforcement learning for skeleton-based action recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5323–5332.

[48] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Y. Choi, "Action-decision networks for visual tracking with deep reinforcement learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1349–1358.

[49] L. Ren, J. Lu, Z. Wang, Q. Tian, and J. Zhou, "Collaborative deep reinforcement learning for multi-object tracking," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 605–621.

[50] L. Ren, X. Yuan, J. Lu, M. Yang, and J. Zhou, "Deep reinforcement learning with iterative shift for visual tracking," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 684–700.

[51] X. Dong, J. Shen, W. Wang, Y. Liu, L. Shao, and F. Porikli, "Hyperparameter optimization for tracking with continuous deep Q-learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 518–527.

[52] J. C. Caicedo and S. Lazebnik, "Active object localization with deep reinforcement learning," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2488–2496.

[53] Z. Jie, X. Liang, J. Feng, X. Jin, W. Lu, and S. Yan, "Tree-structured reinforcement learning for sequential object localization," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 127–135.

[54] S. Mathe, A. Pirinen, and C. Sminchisescu, "Reinforcement learning for visual object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2894–2902.

[55] J. Huang, N. Li, T. Zhang, and G. Li, "A self-adaptive proposal model for temporal action detection based on reinforcement learning," in *Proc. Assoc. Adv. Artif. Intell. (AAAI)*, 2018, pp. 6951–6958.

[56] A. Pirinen and C. Sminchisescu, "Deep reinforcement learning of region proposal networks for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 6945–6954.

[57] N. Dong and E. P. Xing, "Domain adaption in one-shot learning," in *Proc. Eur. Conf. Mach. Learn. Princ. Pract. Knowl. Discovery Databases (ECML-PKDD)*, 2018, pp. 573–588.

[58] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2014, pp. 2672–2680.

[59] X. Wang, L. Li, W. Ye, M. Long, and J. Wang, "Transferable attention for domain adaptation," in *Proc. Assoc. Adv. Artif. Intell. (AAAI)*, 2019, pp. 5345–5352.

[60] L. Ren, X. Yuan, J. Lu, M. Yang, and J. Zhou, "Deep reinforcement learning with iterative shift for visual tracking," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 697–713.

[61] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[62] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 293–321, May 1992.

[63] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 2066–2073.

[64] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2010, pp. 213–226.

[65] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Inst. Technol., Pasadena, CA, USA, 2007.

[66] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, "Deep hashing network for unsupervised domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5385–5394.

[67] X. Peng, B. Usman, N. Kaushik, J. Hoffman, D. Wang, and K. Saenko, "VisDA: The visual domain adaptation challenge," 2017, *arXiv:1710.06924*. [Online]. Available: http://arxiv.org/abs/1710.06924

[68] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.

[69] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

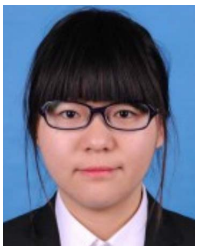
[70] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. Adv. Neural Inf. Process. Syst. Workshop Deep Learn. Unsupervised Feature Learn.*, 2011, p. 5.

[71] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 5, pp. 550–554, May 1994.

[72] Z. Chen, C. Chen, Z. Cheng, B. Jiang, K. Fang, and X. Jin, "Selective transfer with reinforced transfer network for partial domain adaptation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 12703–12711.

[73] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–15.

[74] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.



Jin Chen received the B.S. degree in computer science from the Beijing Institute of Technology, Beijing, China, in 2017, where she is currently pursuing the Ph.D. degree in computer science with the Beijing Laboratory of Intelligent Information Technology.

Her current research interests include domain adaptation, reinforcement learning, and machine learning.

Xinxiao Wu (Member, IEEE) received the B.S. degree in computer science from the Nanjing University of Information Science and Technology, Nanjing, China, in 2005, and the Ph.D. degree in computer science from the Beijing Institute of Technology (BIT), Beijing, China, in 2010.

From 2010 to 2011, she was a Post-Doctoral Research Fellow with Nanyang Technological University, Singapore. She is currently an Associate Professor with the School of Computer Science, BIT. Her research interests include machine learning, computer vision, and video analysis and understanding.

Lixin Duan received the B.Eng. degree from the University of Science and Technology of China, Hefei, China, in 2008, and the Ph.D. degree from Nanyang Technological University, Singapore, in 2012.

He is currently a Full Professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. He received the Microsoft Research Asia Fellowship in 2009. His research interests include machine learning algorithms, especially in transfer learning and domain adaptation, and their applications in computer vision.

Dr. Duan was a recipient of the Best Student Paper Award at the IEEE Conference on Computer Vision and Pattern Recognition 2010.

Shenghua Gao (Member, IEEE) received the B.E. degree (Hons.) from the University of Science and Technology of China, Hefei, China, in 2008, and the Ph.D. degree from Nanyang Technological University, Singapore, in 2012.

From 2012 to 2014, he was a Post-Doctoral Fellow with the Advanced Digital Sciences Center, Singapore. He is currently an Associate Professor and a Principal Investigator with ShanghaiTech University, Shanghai, China. His research interests include computer vision and machine learning.