Adversarial Reinforcement Learning for Unsupervised Domain Adaptation

Youshan Zhang[1], Hui Ye[2], and Brian D. Davison[1]

[1]Computer Science and Engineering, Lehigh University
[2]Computer Science, Georgia State University

{yoz217, bdd3}@lehigh.edu, hye2@student.gsu.edu

Abstract

Transferring knowledge from an existing labeled domain to a new domain often suffers from domain shift in which performance degrades because of differences between the domains. Domain adaptation has been a prominent method to mitigate such a problem. There have been many pre-trained neural networks for feature extraction. However, little work discusses how to select the best feature instances across different pre-trained models for both the source and target domain. We propose a novel approach to select features by employing reinforcement learning, which learns to select the most relevant features across two domains. Specifically, in this framework, we employ Q-learning to learn policies for an agent to make feature selection decisions by approximating the action-value function. After selecting the best features, we propose an adversarial distribution alignment learning to improve the prediction results. Extensive experiments demonstrate that the proposed method outperforms state-of-the-art methods.

1. Introduction

There is high demand for automatic classification of all kinds of data. However, large quantities of labeled training data is a prerequisite for high quality machine learning models. Unfortunately, manual annotation typically involves tremendous human costs. Therefore, it is often necessary to transfer knowledge from an existing labeled domain to an unlabeled new domain. However, due to the phenomenon of domain shift [26], machine learning models do not generalize well from an existing domain to a novel unlabeled domain. Domain adaptation (DA) has been an effective method to mitigate the domain shift problem. Both traditional and deep learning based models are explored.

Traditional methods rely on feature representation of instances (e.g., images) to perform DA tasks. With the emergence of different deep neural networks trained on large datasets, the features for traditional methods changed from low-level features to deep features (Alexnet [18],

ResNet50 [24], Xception [52], InceptionResNet [47] etc.). Distribution alignment includes the alignment of marginal distribution [25, 22, 16], conditional distribution [10, 40] and joint distributions [41]. Subspace learning includes methods in Euclidean space [25, 49] and Riemannian space [9, 52]. However, the performance of traditional methods is heavily affected by the extracted features; a better ImageNet model produces better features than a lower accuracy ImageNet model [50].

Recently, deep learning methods have shown great success for domain adaptation. Most deep domain adaptation networks either design novel distance metrics to measure the discrepancy between two domains or learn domain invariant features using adversarial learning. Distance based methods aim to minimize the discrepancy between source and target domain [38, 20, 7]. Inspired by generative adversarial network (GAN) [11], adversarial learning-based methods consist of a domain classifier and discriminator. The classifier aims to distinguish the source domain and target domain, while the discriminator aims to fool the classifier. By playing the minimax game, the distance between source and target domain can be minimized [6, 37, 48]. The Domain-Adversarial Neural Network (DANN) considers a minimax loss to integrate a gradient reversal layer to promote the discrimination of source and target domain [6]. The Adversarial Discriminative Domain Adaptation (ADDA) method uses an inverted label GAN loss to split the source and target domain, and features can be learned separately [37]. Zhang et al. reweighed the target samples using the degree of confusion between source and target domains. The target samples are assigned by higher weights, which can confuse the domain discriminator [44]. Domain Symmetric Network (SymNet) includes a symmetrically designed source and target classifier. The proposed category level loss can improve the domain level loss by learning the invariant features between two domains [51].

In this paper, we propose to select the best feature pairs across the source and target domains using reinforcement learning, and it interacts with the adversarial distribution alignment learning module, so that we can learn the domain

(a) Source t-SNE view (b) Combination (c) Target t-SNE view
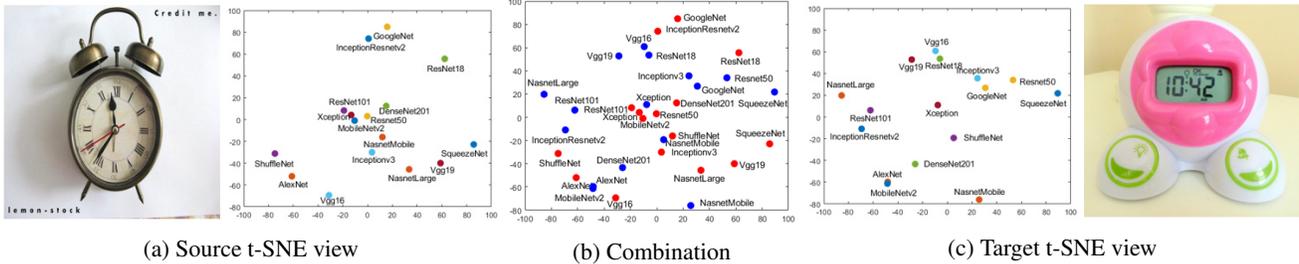
Figure 1: T-SNE view of two images from source domain: Art (a) and target domain: Real world (c) in Office-Home dataset. Different colors in (a) and (c) represent different feature extractors, while red colors in (b) are points from source domain, and blue colors denote points in target domain. The distance between source and target image can be minimized by selecting a proper feature extractor. Source features from ShuffleNet and target features from NasnetMobile have the shortest distance.

invariant features and further minimize the domain discrepancy. Our contributions are three-fold:

1. We propose a novel framework called adversarial reinforcement learning for unsupervised domain adaptation (ARL). Reinforcement learning is employed as a feature selector to identify the closest feature pair between source and target domains.

2. We also develop a new reward across both source and target domains. The proposed deep correlation reward on the target can guide the agent to learn the best policy and select the closest feature pair for both domains.

3. The proposed adversarial learning and domain distribution alignment together mitigate the discrepancy between source and target domains.

Extensive experiments on benchmark datasets demonstrate significant improvements in classification accuracy over the state of the art.

2. Related work

Existing domain adaptation methods including both traditional methods and deep learning based methods more or less rely on pre-trained models for feature extraction. In prior work [50], we explored the effect of model selection of sixteen deep pre-trained ImageNet models on twelve domain adaptation methods. We found that a higher accuracy ImageNet model will produce better features for unsupervised DA. However, that work did not explore whether features from the same pre-trained model are optimal.

Fig. 1 is the t-SNE view of one source and one target image using different feature extractors. Previous work only evaluated their models on the same domain (e.g., train source data using ResNet50 features and test target data also using ResNet50 features). However, such a strategy might not be optimal, since the distance between the source and target from different feature extractors can be shortened. In

Fig, 1b), ShuffleNet and NasnetMobile are closer to each other than others in projected 2D space. Similarly, in the original space, these features can be closer to each other than others, which suggests these two feature sets have minimal distance. It is hence important to identify such close features between two domains.

Reinforcement learning (RL) shows its robustness in learning sophisticated policies by interacting with a complex environment and is widely used in many tasks such as text/image recognition and object tracking. Feng et al. used RL as an instance selector to filter out noisy data and select high quality sentences to improve relation classification accuracy [5]. Carr et al. presented an algorithm using RL to initialize the adversarial autoencoder network; they transferred source domain state representation space to the target domain using a random policy [1]. The DisentAngled Representation Learning Agent (DARLA) model employed denoising autoencoders to learn a visual system to encode the observations from the environment as a disentangled representation and learned a source policy for zero-shot DA [13].

The closest work is DARL (Domain Adversarial Reinforcement Learning) [3], focusing on selecting the data instance from a label-rich source domain to a label scarce target domain called partial domain adaptation, but it does not generalize to unsupervised domain adaptation if there are insufficient labels in the source domain. DARL only selected relevant source data using RL, and then applied adversarial learning to minimize the distance between the source and target domain. However, this RL paradigm relies on the rich labels in the source domain and will fail if the number of labels in the source domain is equal to that in the target domain. Therefore, the RL paradigm on unsupervised domain adaptation should be further explored.

3. Background

In this section, we briefly recap the concepts of domain adaptation, reinforcement learning and adversarial learning, which are the main components of our model.

3.1. Unsupervised Domain Adaptation (UDA)

For unsupervised domain adaptation, given a source domain $\mathcal{D}_S = \{\mathcal{X}_{Si}, \mathcal{Y}_{Si}\}_{i=1}^{n_s}$ of n_s labeled samples in C categories and a target domain $\mathcal{D}_T = \{\mathcal{X}_{Tj}\}_{j=1}^{n_t}$ without any labels ($\mathcal{Y}_T$ for evaluation only). Our ultimate goal is to learn a classifier f under a feature extractor $\mathcal{F}$, that ensures lower generalization error in the target domain.

We propose a new framework for unsupervised domain adaptation, which is able to select the best feature pair between two domains from different pre-trained neural networks using reinforcement learning.

3.2. Reinforcement learning (RL)

The RL paradigm aims to train an agent to interact with an unknown environment, and to maximize the cumulative reward for the task. The agent receives observations and a reward from the environment and performs actions to the environment. This problem can be modeled as a Markov Decision Process (MDP) [33], which is defined as a tuple (S, A, T, R, γ) where S is a set of states, A is a set of actions, T is the transition function $T(s, s', a) = P(s'|s, a)$, which models the possibility of next state s' given action a in state s. R is the reward function $R(s, s', a)$, which gets reward R from state s to s' with discount factor γ in which $0 \leq \gamma \leq 1$. The goal of RL is to learn a policy $\pi(a|s)$, that maximizes the discounted expected reward: $\mathbb{E}[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)]$. T is the timestep at which each episode ends.

In our work, we employ a Q-learning agent to maintain a critic $Q(S, A)$ to estimate the value function. It takes observation S and action A as inputs and outputs the corresponding expectation of reward.

3.3. Adversarial learning

Adversarial learning minimizes the domain discrepancy by a feature extractor and a domain discriminator. The domain discriminator aims to distinguish the source domain from the target domain, while the feature extractor aims to learn domain-invariant representations to fool the domain discriminator.

Given the feature representation of feature extractor $\mathcal{F}$, we can learn a discriminator D, which can distinguish the two domains using the following loss function:

$$\mathcal{L}_\mathcal{A}(\mathcal{X}_S, \mathcal{X}_T) = -\frac{1}{n_s} \sum_{i=1}^{n_s} \log(1 - D(\mathcal{F}(\mathcal{X}_{Si}))) \\ -\frac{1}{n_t} \sum_{j=1}^{n_t} \log(D(\mathcal{F}(\mathcal{X}_{Tj}))) \tag{1}$$

The discriminator D learns the domain distributions by maximizing $\mathcal{L}_\mathcal{A}$ with the fixed $\mathcal{F}$, and the feature extractor $\mathcal{F}$ aims to learn domain-invariant representations via minimizing $\mathcal{L}_\mathcal{A}$ with the optimal discriminator D. For the task in the labeled source domain, it minimizes the following cross-entropy loss.

$$\mathcal{L}_S(f(\mathcal{F}(\mathcal{X}_S)), \mathcal{Y}_S) = -\frac{1}{n_s} \sum_{i=1}^{n_s} \sum_{c=1}^{C} \mathcal{Y}_{Sic} \log(f_c(\mathcal{F}(\mathcal{X}_{Si}))), \tag{2}$$

where f is classifier of the source domain and $\mathcal{Y}_{Sic} \in [0, 1]^C$ is the probability of each class in true label, and $f_c(\mathcal{F}(\mathcal{X}_{Si}))$ is the predicted probability.

4. Methods

4.1. Motivation

Most previous work in UDA only considered a feature extractor $\mathcal{F}$ from a single pre-trained model without exploring other models [24, 2]. As aforementioned, previously [50] we explored how different ImageNet models affect the domain adaptation problem, and found that a higher-performing ImageNet model would produce better features for unsupervised domain adaptation. However, identifying the best set of features (from any source) for UDA has not yet been explored. Do features from the same deep network achieve the best performance? We explore this question using reinforcement learning by exploring all possible pairings of features from sixteen different pre-trained ImageNet models.

4.2. Adversarial reinforcement learning (ARL)

We now formalize unsupervised domain adaptation scenarios in a reinforcement learning setting by considering different pre-trained models.

Since we have several well-trained ImageNet models, we employ a second feature extractor G (which we name the pre-trained feature extractor) to represent the source and target images using pre-trained models[1]. For a given feature extractor, we consider the maximum mean discrepancy (MMD) as a non-parametric distance-measure to compare the distributions of source and target domain by mapping data into Reproducing Kernel Hilbert Space (RKHS). Let $\mathcal{P}_{\mathcal{X}_S}$ and $\mathcal{P}_{\mathcal{X}_T}$ be the distributions of the source and target domains. Then the distance between $\mathcal{P}_{\mathcal{X}_S}$ and $\mathcal{P}_{\mathcal{X}_T}$ can be denoted as:

$$Dist_k(\mathcal{X}_S, \mathcal{X}_T) = ||\frac{1}{n_s} \sum_{i=1}^{n_s} G_k(\mathcal{X}_{Si}) - \frac{1}{n_t} \sum_{j=1}^{n_t} G_k(\mathcal{X}_{Tj})||_\mathcal{H}$$

where G_k is the $k^{th} \in \{1, 2, \cdots K\}$ feature extractor from pre-trained models ($K = 16$), and $\mathcal{H}$ is the universal RKHS,

---

[1]G extracts features from the last fully connected layer of any pre-trained model, and extracted features have the same dimensionality regardless of network.
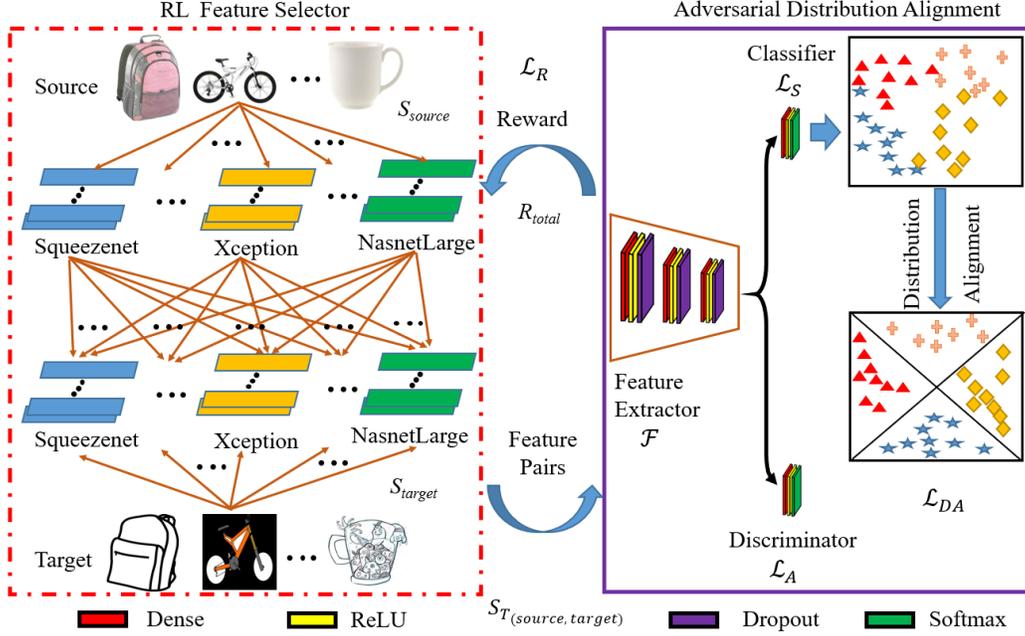
Figure 2: The overall progress of the proposed model. We first employ reinforcement learning as a feature selector, and the adversarial distribution alignment will provide the reward for the feature selector and then the features from the best pair of pre-trained models will be chosen as the input for the distribution alignment model (S_{source} and S_{target} are all possible states for the source and target domain; $S_{T(source,target)}$ is the terminal state with two selected sources of features. $\mathcal{L}_{\mathcal{R}}$ is the reinforcement learning loss; $\mathcal{L}_{\mathcal{S}}$ is the source classification loss; $\mathcal{L}_{\mathcal{A}}$ is the adversarial domain loss and $\mathcal{L}_{\mathcal{DA}}$ is the domain alignment loss).

and $G : \mathcal{X} \to \mathcal{H}$. However, with a different G_k, such a distance can be varied. As seen in Fig. 1, we use the t-SNE view to show one image from the source and target domain, which belong to the same class. We can find that all sixteen feature extractors represent the same image with different projection locations, which shows the differences of varied G. However, we notice that source and target features from the same extractor might not be the closest features between source and target. Therefore, we aim to minimize the distance across all pairs of pre-trained feature extractors in Eq. 3.

$$Dist(\mathcal{X}_{\mathcal{S}}, \mathcal{X}_{\mathcal{T}}) = \frac{1}{K^2} \sum_{k_s=1}^{K} \sum_{k_t=1}^{K} \|\frac{1}{n_s} \sum_{i=1}^{n_s} G_{k_s}(\mathcal{X}_{\mathcal{S}i}) \\ - \frac{1}{n_t} \sum_{j=1}^{n_t} G_{k_t}(\mathcal{X}_{\mathcal{T}j})\|_{\mathcal{H}}, \quad (3)$$

where G_{k_s} and G_{k_t} are pre-trained feature extractors ($k_s/k_t \in \{1, 2, \cdots, K\}$). Therefore, we have many possible feature instances for two domains, the complexity of the exhaustive search is $\mathcal{O}(n_s n_t k^2)$. We treat the RL paradigm as a feature selector, which can select the best feature for two domains. However, the challenge is that extracted features are inevitably noisy because they were extracted from different neural networks, and some lower performance pre-

trained models will cause negative transfer, which means that some features are "functional" but not "informational". If the less informational features are selected, it will hurt the performance on the target domain.

To address this issue, the ARL model learns to choose the most suitable source-target feature pair via their ability to optimize the model. As shown in Fig. 2, we first employ RL to select the best feature pair and then feed them into an adversarial domain alignment framework.

We apply a Q-learning algorithm to select the feature sets for the two domains. The Q-learning agent is a value-based reinforcement learning agent which trains a critic to estimate the return or future rewards. A reinforcement learning policy is a mapping that selects an action to take based on observations from the environment. During training, the agent tunes the parameters of its policy representation to maximize the long-term reward. The details of State, Action, Reward and policy gradient are described as follows.

State. The state S represents the current selected features. As shown in Fig. 2, for one image, we have K possible states for the feature selector. The state $(S_{k_s i}, S_{k_t j})$ is one selected source-target pair including two extracted feature vectors: source state $S_{k_s i}$ and target state $S_{k_t j}$.

The source state and target state representations are from

different feature extractors:

$$S_{source} = \{S_{k_s i}|_{k_s/i=1}^{K/n_s}\}, \quad \text{where} \quad S_{k_s i} = G_{k_s}(\mathcal{X}_{\mathcal{S}i})$$
$$S_{target} = \{S_{k_t j}|_{k_t/j=1}^{K/n_t}\}, \quad \text{where} \quad S_{k_t j} = G_{k_t}(\mathcal{X}_{\mathcal{T}j})$$
$$(4)$$

The initial state $S_0 = \emptyset$, and the terminal state for each feature pair is defined as:

$$S_{T(k_s i, \, k_t j)} = S_{k_s i} \circ S_{k_t j}. \tag{5}$$

Therefore, we can define all possible states as a four components set: $S = \{S_0, S_{k_s i}, S_{k_t j}, S_{T(k_s i, \, k_t j)}\}$, where $k_s/k_t \in \{1, 2, \cdots K\}$, $i \in \{1, 2, \cdots n_s\}$ and $j \in \{1, 2, \cdots n_t\}$. Without loss of generality, the inputs for the adversarial domain alignment are the terminal states of all source and target pairs ($S_{T(source, target)}$).

Action. We define an action $a \in \{0, 1\}$ to indicate whether the source feature or target feature is selected as the input for the adversarial domain alignment. The number of actions is the same as the number of states. The optimal action taken by the agent at timestep t is calculated as:

$$a_t = \max_a Q(S_t, a) \tag{6}$$

where S_t is the state in the timestep t, and the critic $Q(S_t, a)$ is the accumulated reward of taking action a.

Reward. The reward function is an indicator of the utility if features are selected. The difficulty of the current model is how to define a proper reward for both the source and target domain. For the source domain, the reward can be defined considering the training accuracy. For the target domain, we need to define an unsupervised reward; we then propose a deep correlation reward.

Deep correlation reward. Our features are extracted from a well-trained model; we assume that two highly correlated examples should belong to the same class as follows.

$$\mathcal{Y}_i = \mathcal{Y}_j \text{ if } Sim(G(\mathcal{X}_i), \, G(\mathcal{X}_j)) > Sim(G(\mathcal{X}_i), \, G(\mathcal{X}_{j \neq i})),$$

where Sim is the cosine similarity. We then rank all similarity scores and calculate the top-1 cosine similarity matrix for target domains. Hence, we can compare correlated labels with the predicted labels using the source classifier. The reward in the target domain can be defined as:

$$R(f(\mathcal{F}(G(\mathcal{X}_{\mathcal{T}}))), \mathcal{Y}_{\mathcal{T}corr}) = \frac{1}{n_t} \sum_{i=1}^{n_t} (\mathcal{Y}_{pred_i} == \mathcal{Y}_{pred}[\mathcal{Y}_{corr_i}]),$$

where $\mathcal{Y}_{pred}$ is the prediction from the source classifier f, and $\mathcal{Y}_{corr}$ is the correlation label with the size of $N_t \times 1$; it shows the top-1 index, which is highly related to the instance that should be in the same class. Therefore, the reward measures how different the predicted label is to its

nearest neighbors. We then define the reward of the each selected pair as:

$$R(f(\mathcal{F}(G(\mathcal{X}_{\mathcal{S}_i/\mathcal{T}_i}))), \mathcal{Y}_{\mathcal{S}_i/\mathcal{T}_{corr_i}}) =$$
$$\begin{cases} 0 \text{ if } f(\mathcal{F}(G(\mathcal{X}_{\mathcal{S}_i/\mathcal{T}_i}))) \neq \mathcal{Y}_{\mathcal{S}_i/\mathcal{T}_{corr_i}} \\ 1 \text{ if } f(\mathcal{F}(G(\mathcal{X}_{\mathcal{S}_i/\mathcal{T}_i}))) = \mathcal{Y}_{\mathcal{S}_i/\mathcal{T}_{corr_i}} \end{cases} \tag{7}$$

We then define the total reward of the source and target domain in Eq. 8.

$$R_{total} = \frac{1}{n_s} \sum_{i=1}^{n_s} R(f(\mathcal{F}(G(\mathcal{X}_{\mathcal{S}i}))), \mathcal{Y}_{\mathcal{S}i})$$
$$+ \frac{1}{n_t} \sum_{j=1}^{n_t} R(f(\mathcal{F}(G(\mathcal{X}_{\mathcal{T}j}))), \mathcal{Y}_{\mathcal{T}corr_j}), \tag{8}$$

where $R(f(\mathcal{F}(G(\mathcal{X}_{\mathcal{S}i}))), \mathcal{Y}_{\mathcal{S}i})$ is the reward for one source data, and the reward for one target data is denoted as $R(f(\mathcal{F}(G(\mathcal{X}_{\mathcal{T}j}))), \mathcal{Y}_{\mathcal{T}corr_j})$.

The loss function of the RL paradigm is denoted as:

$$\mathcal{L}_{\mathcal{R}}(\mathcal{R}_{\mathcal{S}}, \mathcal{R}_{\mathcal{T}}) = \mathcal{L}(f(\mathcal{F}(G(\mathcal{X}_{\mathcal{S}}))), \mathcal{Y}_{\mathcal{S}})$$
$$+ \mathcal{L}(f(\mathcal{F}(G(\mathcal{X}_{\mathcal{T}}))), \mathcal{Y}_{\mathcal{T}corr})$$

Policy gradient. We use the standard policy gradient to propagate the reward to the training of the state and action network. In all T step, it first calculates the accumulated reward R_T for each of the t steps: $R_T = \sum_{t=1}^{T} \gamma R_{total_t}$, where γ is discounted factor.

We optimize the parameters of the state and action using the following standard policy gradient.

$$Q(\text{S}, \text{a})^\star \longleftarrow Q(S, a) + \alpha[R_{total_t} + \gamma \max_a Q(S', a) - Q(S, a)], \tag{9}$$

where $Q(\text{S}, \text{a})^\star$ is the updated target value, and α is the learning rate.

4.3. Adversarial distribution alignment learning

After selecting each feature pair, we first apply adversarial learning to identify the domain distribution by minimizing the loss function in Eqs. 1-2. We then align marginal and conditional distributions of both domains as follows.

Manifold Embedded Distribution Alignment (MEDA), proposed by Wang et al. [41], aligns learned features from manifold learning. The original MEDA used the GFK model to learn manifold features, and the GFK model is based on the SGF model. However, Zhang et al. showed that there are defects in the SGF model, which cannot estimate the geodesic of sub-source and sub-target domains [52]. We modified the domain alignment loss as follows:

$$\mathcal{L}_{\mathcal{DA}}(\mathcal{D}_{\mathcal{S}}, \mathcal{D}_{\mathcal{T}}) = \arg\min(\mathcal{L}_{\mathcal{G}}(f(\mathcal{F}(G(\mathcal{X}_{\mathcal{S}}))), \mathcal{Y}_{\mathcal{S}}) + \eta||f||^2$$
$$+ \lambda \overline{D_f}(\mathcal{D}_{\mathcal{S}}, \mathcal{D}_{\mathcal{T}}) + \rho R_f(\mathcal{D}_{\mathcal{S}}, \mathcal{D}_{\mathcal{T}})) \tag{10}$$

where f is the source classier in Eq. 2, $\mathcal{L}_\mathcal{G}$ is the sum of squares loss; $||f||^2$ is the squared norm of f; and the first two terms minimize the structural risk of the source domain. $\overline{D_f}(\cdot, \cdot)$ represents the dynamic distribution alignment; $R_f(\cdot, \cdot)$ is a Laplacian regularization; η, λ, and ρ are regularization parameters. Specifically, $\overline{D_f}(\mathcal{D}_\mathcal{S}, \mathcal{D}_\mathcal{T}) = (1 - \mu)D_f(P_\mathcal{S}, P_\mathcal{T}) + \mu \sum_{c=1}^{C} D_f^c(Q_\mathcal{S}, Q_\mathcal{T})$, where μ is an adaptive factor to balance the marginal distribution $(P_\mathcal{S}, P_\mathcal{T})$, and conditional distribution $(Q_\mathcal{S}, Q_\mathcal{T})$, and $c \in \{1, \cdots, C\}$ is the class indicator [41].

4.4. Overall objective function

The overall optimization problem of the proposed ARL model is in Eq. 11.

$$
\begin{aligned}
\mathcal{L}(\mathcal{X}_\mathcal{S}, \mathcal{Y}_\mathcal{S}, \mathcal{X}_\mathcal{T}, \mathcal{Y}_{\mathcal{T}corr}) = \arg\min(&\mathcal{L}_\mathcal{R}(\mathcal{R}_\mathcal{S}, \mathcal{R}_\mathcal{T}) \\
+ &\mathcal{L}_\mathcal{S}(f(\mathcal{F}(G(\mathcal{X}_\mathcal{S}))), \mathcal{Y}_\mathcal{S}) + \mathcal{L}_\mathcal{A}(G(\mathcal{X}_\mathcal{S}), G(\mathcal{X}_\mathcal{T})) \\
+ &\mathcal{L}_{\mathcal{DA}}(\mathcal{D}_\mathcal{S}, \mathcal{D}_\mathcal{T})),
\end{aligned}
$$

(11)

where $\mathcal{L}_\mathcal{A}$ and $\mathcal{L}_\mathcal{S}$ are from Eq. 1 and Eq. 2 except that the inputs are the extracted features from pre-trained models ($G(\mathcal{X}_\mathcal{S})$ and $G(\mathcal{X}_\mathcal{T})$) not the raw images. The detailed procedures of our ARL model are shown in Algorithm 1.

Algorithm 1 Adversarial Reinforcement Learning UDA

Input: Source domain $\mathcal{X}_\mathcal{S}$ and $\mathcal{Y}_\mathcal{S}$, target images $\mathcal{X}_\mathcal{T}$, K, and $numEpochs$.
Output: Optimized features extractor $\mathcal{F}$ and source classifier f.
 1: Extract features from pre-trained models $\{G_k(\mathcal{X}_\mathcal{S}), G_k(\mathcal{X}_\mathcal{T})\}_{k=1}^{K}$;
 2: Initialize the critic $Q(S, a)$;
 3: **for** $i = 1$ to $numEpochs$
 4:    Initialize the state S;
 5:    **while** S is not a terminal state **do**
 6:       Take an action a_t using the policy in Eq. 6;
 7:       Execute action a_t, observe the reward R_{total_t} by Eq. 8 and next state S';
 8:       Update the critic $Q(\text{S, a})^\star$ by Eq. 9;
 9:       Set the state S to S';
10:    **end while**
11:    Update $\mathcal{F}$ and f by Eq. 11.
12: **end for**

---

5. Experiments

5.1. Datasets

We evaluate our ARL model using two benchmark datasets, which are widely used in UDA. We follow the protocol of prior work [50] which extracted features from sixteen pre-trained neural networks (Squeezenet [15], Alexnet [18], Googlenet [35], Shufflenet [45], Resnet18 [12], Vgg16 [30], Vgg19 [30], Mobilenetv2 [29], Nasnetmobile [53], Resnet50 [12],

Resnet101 [12], Densenet201 [14], Inceptionv3 [36], Xception [4], Inceptionresnetv2 [34], Nasnetlarge [53]). All extracted features are from the last fully connected layer [46], and each image has feature size of 1,000.

Office + Caltech-10 [9] is a standard benchmark for domain adaptation, which contains Office 10 and Caltech 10 datasets. It consists of 2,533 images in four domains: Amazon (A), Webcam (W), DSLR (D) and Caltech (C). In the experiments, C → A means learning knowledge from domain C and which is applied to domain A. There are twelve transfer tasks in Office + Caltech-10 dataset.

Office-31 [28] consists of 4,110 images in 31 classes from three domains: Amazon (A), which contains images downloaded from amazon.com, Webcam (W), and DSLR (D), both containing images that are taken by a web camera or a digital SLR camera with different settings, respectively. We evaluate methods across all six transfer tasks.

Office-Home [39] contains 15,588 images from four domains, and it has 65 categories. Specifically, Art (Ar) denotes artistic depictions for object images, Clipart (Cl) describes picture collection of clipart, Product (Pr) shows object images with a clear background and is similar to Amazon category in Office-31, and Real-World (Rw) represents object images collected with a regular camera. There are also twelve tasks in this dataset. Therefore, we have a total of 30 tasks in our experiments.

5.2. Implementation details

The number of pre-trained backbone networks is fixed (i.e., $K = 16$.) We apply the ϵ-greedy strategy for the Q-learning with $\epsilon = 0.9$, the discount factor is $\gamma = 0.99$, $numEpochs = 1000$ and the learning rate α is $1e - 3$. The numbers of units of the dense layer are 512, 256, and 128, respectively. The rate of the Dropout layer is 0.5. It ends with a Dense layer (the number of units is the number of classes in each dataset (10 and 65 in our experiments). Parameters in domain distribution alignment are $\eta = 0.1$, $\lambda = 10$, and $\rho = 10$, which are fixed based on previous research [41]. ϵ, γ, α and $numEpochs$ are determined by the performance of source domain. After selecting features using ARL model, adversarial domain alignment learning is used to generate a final class prediction on the target domain, whose accuracy is reported.

5.3. Evaluation results

We also compare our results with state-of-the-art methods (including both traditional methods and deep neural networks). The performance on Office + Caltech-10, Office-31 and Office-Home are shown in Tables 1-3. For a fair comparison, we highlight in bold those methods that are re-implemented using our extracted features (we concatenate all sixteen deep neural network features), and other methods are directly reported from their original papers. We note

Table 1: Accuracy (%) on Office + Caltech-10 dataset

Task	C→A	C→W	C→D	A→C	A→W	A→D	W→C	W→A	W→D	D→C	D→A	D→W	Ave.
GFK [9]	94.6	94.9	95.5	92.6	90.5	94.3	93.5	95.7	**100**	93.6	95.9	98.6	95.0
CORAL [31]	96.3	97.3	98.1	94.3	96.3	98.7	94.4	96.3	**100**	93.3	95.6	98.6	96.6
JGSA [43]	95.1	97.6	96.8	93.9	94.2	96.2	95.5	95.9	**100**	94.0	**96.3**	99.3	96.2
ARTL [19]	96.3	94.9	96.2	93.9	98.3	97.5	94.7	96.7	**100**	94.4	96.2	99.7	96.6
MEDA [41]	96.3	98.3	96.2	94.6	99.0	**100**	94.8	96.6	**100**	93.6	96.0	99.3	97.0
MDA [47]	**96.5**	97.6	99.4	93.5	99.0	**100**	94.3	**96.8**	**100**	91.3	96.1	**100**	97.1
DDC [38]	91.9	85.4	88.8	85.0	86.1	89.0	78.0	83.8	**100**	79.0	87.1	97.7	86.1
DCORAL [32]	89.8	97.3	91.0	91.9	**100**	90.5	83.7	81.5	90.1	88.6	80.1	92.3	89.7
DAN [20]	92.0	90.6	89.3	84.1	91.8	91.7	81.2	92.1	**100**	80.3	90.0	98.5	90.1
RTN [23]	93.7	96.9	94.2	88.1	95.2	95.5	86.6	92.5	**100**	84.6	93.8	99.2	93.4
MDDA [27]	93.6	95.2	93.4	89.1	95.7	96.6	86.5	94.8	**100**	84.7	94.7	99.4	93.6
ARL	96.1	**99.3**	**100**	**95.4**	98.6	99.4	**95.7**	96.3	**100**	**95.3**	**96.3**	99.7	**97.7**

Table 2: Accuracy (%) on Office-Home dataset

Task	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	Ave.
GFK [9]	40.0	66.5	71.8	56.8	66.4	65.1	58.1	43.0	74.1	65.3	44.9	76.3	60.7
CORAL [31]	50.5	77.5	81.6	66.3	77.0	77.3	69.1	51.0	81.7	73.7	52.0	83.8	70.1
JGSA [43]	45.8	73.7	74.5	52.3	70.2	71.4	58.8	47.3	74.2	60.4	48.4	76.8	62.8
ARTL [19]	56.5	80.4	81.4	68.7	81.6	81.7	70.1	56.2	83.3	72.8	58.2	85.6	73.0
MEDA [41]	49.1	75.6	79.1	66.7	77.2	75.8	68.2	50.4	79.9	71.9	53.2	82.0	69.1
MDA [47]	57.5	80.4	82.1	72.4	82.5	82.2	69.7	55.7	82.8	74.6	58.1	84.9	73.5
DANN [8]	45.6	59.3	70.1	47.0	58.5	60.9	46.1	43.7	68.5	63.2	51.8	76.8	57.6
JAN [24]	45.9	61.2	68.9	50.4	59.7	61.0	45.8	43.4	70.3	63.9	52.4	76.8	58.3
CDAN-RM [21]	49.2	64.8	72.9	53.8	62.4	62.9	49.8	48.8	71.5	65.8	56.4	79.2	61.5
CDAN-M [21]	50.6	65.9	73.4	55.7	62.7	64.2	51.8	49.1	74.5	68.2	56.9	80.7	62.8
TADA [42]	53.1	72.3	77.2	59.1	71.2	72.1	59.7	53.1	78.4	72.4	60.0	82.9	67.6
SymNets [51]	47.7	72.9	78.5	64.2	71.3	74.2	64.2	48.8	79.5	74.5	52.6	82.7	67.6
ARL	**58.6**	**83.3**	**84.1**	**74.4**	**84.2**	**83.4**	**73.4**	**58.4**	**84.9**	**76.8**	**75.6**	**87.1**	**75.6**

Table 3: Accuracy (%) on Office-31 dataset

Task	A→W	A→D	W→A	W→D	D→A	D→W	Ave.
GFK [9]	78.3	77.9	72.1	98.0	68.9	95.8	81.8
CORAL [31]	87.9	86.6	75.7	98.2	74.0	96.7	86.5
JGSA [43]	87.2	85.1	76.1	99.0	74.9	97.2	86.6
ARTL [19]	89.1	91.0	77.9	**100**	77.6	98.2	89.0
MEDA [41]	91.7	89.2	77.1	97.4	76.5	96.2	88.0
MDA [47]	90.9	93.0	78.1	99.0	78.2	98.1	89.6
RTN [23]	84.5	77.5	64.8	99.4	66.2	96.8	81.6
DANN [8]	82.0	79.7	67.4	99.1	68.2	96.8	81.6
ADDA [37]	86.2	77.8	68.9	98.4	69.5	96.2	82.9
JDDA [2]	82.6	79.8	66.7	99.7	57.4	95.2	80.2
JAN [24]	85.4	84.7	70.0	99.8	68.6	97.4	84.3
SymNets [51]	90.8	93.9	72.5	**100**	74.6	98.8	88.4
CAN [17]	94.5	95.0	77.0	99.8	78.0	**99.1**	90.6
ARL	**96.3**	**95.9**	**79.9**	99.3	**80.9**	98.3	**91.8**

that classification performance can be low in some tasks (e.g., W→D in Office-31 dataset). One underlying reason is that $SymNets$ and CAN are more likely to reduce the discrepancy between domain W and domain D than other methods. However, the ARL model wins more tasks than any other, and outperforms all state-of-the-art methods in terms of average accuracy (especially in the Office-Home

dataset). This demonstrates the benefits of RL in selecting similar features between the source and target domain for UDA. It is compelling that our ARL model substantially enhances the classification accuracy across different datasets.

5.4. Ablation study

To better demonstrate the performance of our model, we report the effects of different loss functions on classification accuracy in Tab. 4. Notice that we should at least maintain the source classifier (i.e., $\mathcal{L}_S$) to train the source domain and test on the target domain. "ARL$-\mathcal{R}/\mathcal{A}/\mathcal{DA}$" is implemented without reinforcement learning, adversarial domain loss, and distribution alignment loss. It is a simple model, which only trains the source domain without minimizing the domain discrepancy. It is also the worst baseline since it did not consider other loss functions. "ARL$-\mathcal{A}/\mathcal{DA}$" reports results without performing the additional adversarial domain loss and domain distribution alignment. "ARL$-\mathcal{R}$" ignores the feature selector using reinforcement learning and reduces the domain discrepancy with adversarial learning and the distribution alignment. In Tab. 4, all variations with "$-\mathcal{R}$" omit the reinforcement learning for fea-

ture selection, and the model is trained with the best features, which are extracted from the pre-trained Nasnetlarge model. We observe that as additional loss functions are included, the robustness of our model keeps improving. In addition, the performance of models that include reinforcement learning is better than those who exclude reinforcement learning. For instance, "$ARL - \mathcal{R}$" is worse than "$ARL - \mathcal{A}$" and "$ARL - \mathcal{AD}$". This implies that the feature selector using reinforcement learning is powerful and important to identify the most similar paired features for the source and target domain. Adversarial learning and domain alignment also further improve the classification accuracy. Therefore, we can conclude that each of these loss functions are important in minimizing the target domain risk.

6. Discussion

In these extensive experiments, our method achieves the highest average accuracy. Therefore, the quality of our model exceeds that of the state-of-the-art methods. There are two prominent reasons for this success. First of all, RL discovers the closest features between source and target, which produces the minimum discrepancy between two domains from feature space. Secondly, the proposed adversarial learning further reduces the domain discrepancy, and domain distribution alignment jointly aligns both the conditional and marginal distributions of two domains, which guarantees the agreement in label space. Although the absolute improvement in the Office+caltech-10 dataset is not large, it reduces the average error by more than 20% over the best baseline method.

One challenge of our model is that the reward of the target domain is not directly calculated based on the true labels since we do not have ground truth for the target domain. A better unsupervised reward function for the target domain may improve performance. Although we extract features from sixteen different pre-trained ImageNet models, these features are fixed once they are extracted and do not need to be re-extracted during model training, meaning that feature extraction has a fixed cost. The RL paradigm

Table 4: Ablation experiments on Office-31. ($\mathcal{R}$: reinforcement learning, $\mathcal{A}$: adversarial domain loss, and $\mathcal{DA}$: distribution alignment loss.)

Task	A→W	A→D	W→A	W→D	D→A	D→W	**Ave.**
ARL$-\mathcal{R}/\mathcal{A}/\mathcal{DA}$	91.9	88.5	78.3	98.1	77.8	97.3	88.7
ARL$-\mathcal{R}/\mathcal{DA}$	92.6	89.9	78.5	98.2	78.9	97.6	89.3
ARL$-\mathcal{R}/\mathcal{A}$	92.8	92.9	78.9	98.6	79.1	97.6	90.0
ARL$-\mathcal{R}$	92.9	93.0	79.4	98.6	79.2	97.7	90.1
ARL$-\mathcal{A}/\mathcal{DA}$	94.6	94.3	78.9	99.1	78.4	97.4	90.5
ARL$-\mathcal{DA}$	95.6	95.1	79.1	99.0	79.8	97.9	91.1
ARL$-\mathcal{A}$	96.1	95.3	79.3	99.2	79.8	98.0	91.3
ARL	**96.3**	**95.9**	**79.9**	**99.3**	**80.9**	**98.3**	**91.8**
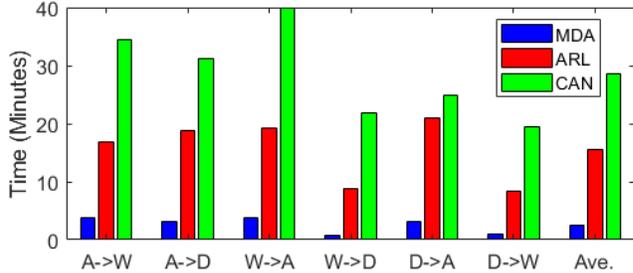
Figure 3: The computation time of different Office-31 transfer tasks for ARL, MDA, and CAN models.

consumes the major computation time in our model. As shown in Fig. 3, we compare the computation time of six transfer tasks in Office-31 dataset with the best traditional method: MDA [52], and the best deep learning based method: CAN [17]. Our model ARL uses more computation time than the MDA model since MDA does not need to find the best feature pair. Meanwhile, our model requires less time than the CAN model and achieves the highest accuracy since the CAN model takes more time to re-train all raw images. Therefore, our model can achieve higher performance with reasonable computation time.

What can we learn from the ARL model? The ARL model employs a reinforcement learning paradigm to explore all possible pairings of features from sixteen pre-trained ImageNet models. Extensive experiments reveal that RL is effective in finding the best feature pair between the source and target domain and outperforms all other baseline methods. As mentioned in Sec. 4.1, do features from the same deep network achieve the best performance? The answer is "No". We find that features from the same deep network surprisingly do not generate the best performance in the UDA. In most tasks, the best pair is "Xception-Nasnetlarge", which means training the model with Xception features and applying the model to Nasnetlarge features achieves the best results. One possible reason is that the discrepancy between Xception and Nasnetlarge features are smaller than other combinations. Therefore, we recommend choosing the Xception model as a feature extractor for the source domain and using the Nasnetlarge model for the target domain to promote classification accuracy if the complex RL framework is excluded.

7. Conclusion

This paper selects features via reinforcement learning. After determining the most similar sets of features for the source and target domains, we utilize adversarial learning to reduce further the source-target discrepancy. In addition, we align the joint distribution between the two domains to achieve the highest classification accuracy. Extensive experiments show that the proposed ARL model is better than state-of-the-art domain adaptation methods.

References

[1] T. Carr, M. Chli, and G. Vogiatzis. Domain adaptation for reinforcement learning on the atari. In *Proceedings of the 18th International Conference on Autonomous Agents and Multi-Agent Systems*, pages 1859–1861. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[2] C. Chen, Z. Chen, B. Jiang, and X. Jin. Joint domain alignment and discriminative feature learning for unsupervised deep domain adaptation. In *Proceedings of 33rd AAAI Conference on Artificial Intelligence*, 2019.

[3] J. Chen, X. Wu, L. Duan, and S. Gao. Domain adversarial reinforcement learning for partial domain adaptation. *IEEE Transactions on Neural Networks and Learning Systems*, 2020. In press.

[4] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1251–1258, 2017.

[5] J. Feng, M. Huang, L. Zhao, Y. Yang, and X. Zhu. Reinforcement learning for relation classification from noisy data. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[6] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[7] M. Ghifary, W Bastiaan Kleijn, M. Zhang, and D. Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2551–2559, 2015.

[8] M. Ghifary, W. B. Kleijn, and M. Zhang. Domain adaptive neural networks for object recognition. In *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*, pages 898–904. Springer, 2014.

[9] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2066–2073. IEEE, 2012.

[10] M. Gong, K. Zhang, T. Liu, D. Tao, C. Glymour, and B. Schölkopf. Domain adaptation with conditional transferable components. In *Proceedings of the International Conference on Machine Learning*, pages 2839–2848, 2016.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[13] I. Higgins, A. Pal, A. Rusu, L. Matthey, C. Burgess, A. Pritzel, M. Botvinick, C. Blundell, and A. Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1480–1490. JMLR.org, 2017.

[14] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[15] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[16] M. Jiang, W. Huang, Z. Huang, and G. G. Yen. Integration of global and local metrics for domain adaptation learning via dimensionality reduction. *IEEE Transactions on Cybernetics*, 47(1):38–51, 2017.

[17] G. Kang, L. Jiang, Y. Yang, and A. G. Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4893–4902, 2019.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[19] J. Long, M.and Wang, G. Ding, S. J. Pan, and S. Y. Philip. Adaptation regularization: A general framework for transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 26(5):1076–1089, 2013.

[20] M. Long, Y. Cao, J. Wang, and M. I. Jordan. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015.

[21] M. Long, Z. Cao, J. Wang, and M. I. Jordan. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pages 1647–1657, 2018.

[22] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu. Transfer joint matching for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1410–1417, 2014.

[23] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Unsupervised domain adaptation with residual transfer networks. In *Advances in Neural Information Processing Systems*, pages 136–144, 2016.

[24] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Deep transfer learning with joint adaptation networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 2208–2217. JMLR.org, 2017.

[25] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *IEEE Trans. on Neural Networks*, 22(2):199–210, 2011.

[26] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[27] M. M. Rahman, C. Fookes, M. Baktashmotlagh, and S. Sridharan. On minimum discrepancy estimation for deep domain adaptation. In *Domain Adaptation for Visual Understanding*, pages 81–94. Springer, 2020.

[28] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *Proceedings of the European Conference on Computer Vision*, pages 213–226. Springer, 2010.

[29] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[31] B. Sun, J. Feng, and K. Saenko. Correlation alignment for unsupervised domain adaptation. In *Domain Adaptation in Computer Vision Applications*, pages 153–171. Springer, 2017.

[32] B. Sun and K. Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *Proc. of European Conference on Computer Vision*, pages 443–450. Springer, 2016.

[33] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[34] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017.

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[36] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[37] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

[38] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[39] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5018–5027, 2017.

[40] J. Wang, Y. Chen, L. Hu, X. Peng, and P. S. Yu. Stratified transfer learning for cross-domain activity recognition. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10. IEEE, 2018.

[41] J. Wang, W. Feng, Y. Chen, H. Yu, M. Huang, and P. S. Yu. Visual domain adaptation with manifold embedded distribution alignment. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, pages 402–410, 2018.

[42] X. Wang, L. Li, W. Ye, M. Long, and J. Wang. Transferable attention for domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5345–5352, 2019.

[43] J. Zhang, W. Li, and P. Ogunbona. Joint geometrical and statistical alignment for visual domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1859–1867, 2017.

[44] W. Zhang, W. Ouyang, W. Li, and D. Xu. Collaborative and adversarial network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3801–3809, 2018.

[45] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018.

[46] Y. Zhang, J. P. Allem, J. B. Unger, and T. B. Cruz. Automated identification of hookahs (waterpipes) on instagram: an application in feature extraction using convolutional neural network and support vector machine classification. *Journal of Medical Internet Research*, 20(11):e10513, 2018.

[47] Y. Zhang and B. D. Davison. Modified distribution alignment for domain adaptation with pre-trained Inception ResNet. *arXiv preprint arXiv:1904.02322*, 2019.

[48] Y. Zhang and B. D. Davison. Adversarial consistent learning on partial domain adaptation of PlantCLEF 2020 challenge. In *CLEF working notes 2020, CLEF: Conference and Labs of the Evaluation Forum*, 2020.

[49] Y. Zhang and B. D. Davison. Domain adaptation for object recognition using subspace sampling demons. *Multimedia Tools and Applications*, pages 1–20, 2020.

[50] Y. Zhang and B. D. Davison. Impact of ImageNet model selection on domain adaptation. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision Workshops*, pages 173–182, 2020.

[51] Y. Zhang, H. Tang, K. Jia, and M. Tan. Domain-symmetric networks for adversarial domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5031–5040, 2019.

[52] Y. Zhang, S. Xie, and B. D. Davison. Transductive learning via improved geodesic sampling. In *Proceedings of the 30th British Machine Vision Conference*, 2019.

[53] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2018.