

# Selective Pseudo-Labeling with Reinforcement Learning for Semi-Supervised Domain Adaptation

Bingyu Liu<sup>1,3</sup>, Yuhong Guo<sup>1,2</sup>, Jieping Ye<sup>1</sup>, Weihong Deng<sup>3</sup>

<sup>1</sup> Didi Chuxing <sup>2</sup> Carleton University <sup>3</sup> Beijing University of Posts and Telecommunications

## Abstract

Recent domain adaptation methods have demonstrated impressive improvement on unsupervised domain adaptation problems. However, in the semi-supervised domain adaptation (SSDA) setting where the target domain has a few labeled instances available, these methods can fail to improve performance. Inspired by the effectiveness of pseudo-labels in domain adaptation, we propose a reinforcement learning based selective pseudo-labeling method for semi-supervised domain adaptation. It is difficult for conventional pseudo-labeling methods to balance the correctness and representativeness of pseudo-labeled data. To address this limitation, we develop a deep Q-learning model to select both accurate and representative pseudo-labeled instances. Moreover, motivated by large margin loss's capacity on learning discriminative features with little data, we further propose a novel target margin loss for our base model training to improve its discriminability. Our proposed method is evaluated on several benchmark datasets for SSDA, and demonstrates superior performance to all the comparison methods.

## 1 Introduction

Deep convolutional neural networks (CNNs) (Simonyan and Zisserman 2014; Krizhevsky, Sutskever, and Hinton 2012; He et al. 2016; Szegedy et al. 2015; Xie et al. 2017; Wang and Deng 2018a; Li and Deng 2018) have achieved remarkable success in image classification tasks. When trained on large-scale labeled data, deep networks can learn discriminative representations and present great performance. However, it is difficult to collect and annotate datasets for many domains. A good option is to use the labeled data available in other domains for training models, which however often presents a domain shift challenge between the two domains and degrades the test performance. To address this problem, many unsupervised domain adaptation (UDA) methods (Wang and Deng 2018b; Ganin and Lempitsky 2015; Long et al. 2015, 2018; Saito et al. 2018a) have been proposed. UDA aims to improve the generalization performance on unlabeled target domains. However, in reality a few labeled target instances can be available in target domains, and this semi-supervised domain adaptation (SSDA) setting is more common. According to (Saito et al. 2019), UDA methods often fail to improve performance compared with just training on the unified data of labeled source and target samples in the semi-supervised setting.

For the task of domain adaptation, the purpose is to improve the generalization performance in the target domain. In the SSDA setting, we have a few labeled target samples, but the number of them is too small to represent the distribution of target unlabeled data. To increase the number of labeled instances in the target domain without incurring annotation cost, one intuitive strategy is to exploit pseudo-labels of the target domain samples produced by a current prediction model. However, the pseudo-labels can often be very noisy and contain many wrong labels, while training with the mislabeled samples can negatively impact the original model. This motivates the straightforward selective pseudo-labeling strategy which selects the most confident pseudo-labels to increase their probability of correctness (Chen et al. 2019). This simple strategy can select more accurate samples but non-necessarily the most useful samples for the prediction model. For example, the more confident and accurate samples may be closer to the labeled data and fail to represent the distribution of the target domain. It is more reasonable but difficult to perform selective pseudo-labeling by balancing the accuracy and the representativeness of the selected samples. To address this challenge, in this paper we propose a reinforcement learning based selective pseudo-labeling method. Our strategy is to use deep Q-learning to learn appropriate selection policies with reward functions that reflect both factors of label correctness and data representativeness.

In addition, due to the lack of labeled data in the target domain, the training methods typically have limited capacity in learning discriminative decision boundaries for the target domain. Inspired by the observation that large margin loss functions (Liu et al. 2017; Wang et al. 2018a; Deng et al. 2019) can help learn discriminative features, we propose a contrastive target margin loss function over the labeled data from the source and target domains. As illustrated in Fig. 1, the target margin loss make the labeled target samples play a bigger role in learning the decision boundaries.

Overall, the contribution of this paper can be summarized as follows: (1) We propose a novel reinforcement learning framework for selective pseudo-labeling for semi-supervised domain adaptation. (2) We propose a contrastive target margin loss for SSDA that takes both generalization and discrimination into consideration. (3) Extensive experiments are conducted on DomainNet (Peng

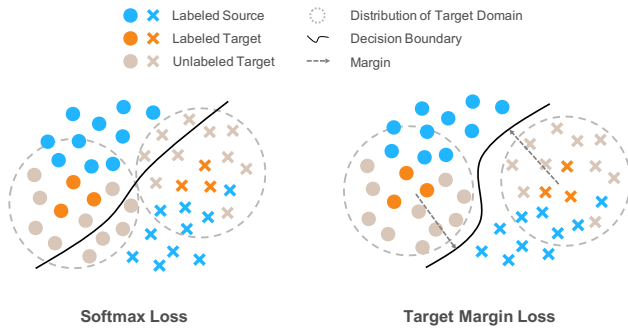


Figure 1: An illustration of the proposed target margin loss. In SSDA setting, the number of labeled target samples is much smaller than that of labeled source samples so that the decision boundary is mainly depending on the source domain. Our target margin loss can make the labeled target data more important for the decision boundary so that the decision regions in target domain will be more separated

et al. 2019), Office-31 (Saenko et al. 2010) and Office-Home (Venkateswara et al. 2017). The results show that our proposed method achieves the state-of-the-art semi-supervised domain adaptation performance.

## 2 Related Work

**Domain Adaptation.** Domain adaptation (Wang and Deng 2018b; Gong et al. 2012; Ganin and Lempitsky 2015; Sun, Feng, and Saenko 2016) aims to generalize models across different domains of different distributions. Most recent methods are focusing on unsupervised domain adaptation (UDA) which has a label-rich source domain and an unlabeled target domain. A widely used way in UDA methods is to add a domain discriminator which classifies whether a sample is drawn from the source or target domain. Then adversarial learning is applied to minimize the distance between the feature distributions in source and target domain. The domain-adversarial neural network (DANN) (Ganin and Lempitsky 2015) proposes the standard domain adversarial architecture and introduces a gradient reversal layer (GRL) for domain confusion loss produced by the discriminator. Inspired by Conditional Generative Adversarial Networks (CGANs) (Goodfellow et al. 2014), Long et al. (Long et al. 2018) proposes Conditional Domain Adversarial Network (CDAN) by combining the discriminative information conveyed in the classifier predictions into the adversarial adaptation. There are also other methods (Saito et al. 2018a,b) encouraging to generate more discriminative features for the target domain. In fact, semi-supervised domain adaptation (SSDA) is a more common setting in real-world datasets in which limited labeled data can be available in target domain. However, it has not been studied extensively, especially in the field of deep learning. A little conventional work (Donahue et al. 2013; Yao et al. 2015; Ao, Li, and Ling 2017) has concentrated on this important task. As for deep learning based methods, Saito et al. (Saito et al. 2019) propose a Min-

imax Entropy (MME) method by alternately maximizing the conditional entropy of unlabeled target data and minimizing it to optimize the classifier and the feature extractor respectively. Meanwhile, (Saito et al. 2019) shows that UDA methods can rarely improve accuracy in SSDA. Therefore, we propose a reinforcement learning based pseudo-labeling method which can effectively exploit the information of target labeled data to generate more discriminative features with distributions closer to the target domain.

**Pseudo-Labeling.** Pseudo-labeling is an effective way to extend label set when the number of labels is limited. As for SSDA, pseudo-labeling can be used for target domain which has little labeled data. There are two strategies for pseudo-labeling without selection, hard labeling (Long et al. 2013; Zhang, Li, and Ogunbona 2017; Wang et al. 2018b) and soft labeling (Pei et al. 2018). The hard labeling strategy assigns a pseudo-label with only one class predicted by the classifier to each target unlabeled instance. Then the pseudo-labeled target data will be combined with original labeled data to train an improved model. However, due to the weak classifier in the initial stage of training, many samples will be mis-labeled. Using these mis-labeled data for supervised training can cause serious harm to the model. Thus, soft labeling has been employed. Taking account of the confidence of prediction, the soft labeling strategy assigns the conditional probability of each class predicted by the classifier to the target unlabeled data. As for selective pseudo-labeling (Zhang et al. 2018; Wang, Bu, and Breckon 2019; Chen et al. 2019), a subset of target unlabeled samples which are most confident in the prediction are selected to be pseudo-labeled. This sample selection strategy can make the pseudo-labels for training more accurate while it also has a limitation that the selected samples are generally too “easy” to represent the distribution of target data. In this work we use reinforcement learning to learn appropriate policies for selecting more accurate and representative pseudo-labeled samples.

**Reinforcement Learning.** Reinforcement learning (RL) is a technique which trains an agent to learn policies based on trial and error in a dynamic environment. The training strategy is to maximize the accumulated reward from the environment. No longer limited to its traditional applications in robotics and control, RL has made great progress in many vision tasks, such as action recognition (Yeung et al. 2017), person re-identification (Zhang, Wang, and Zhang 2018) and face recognition (Liu et al. 2019). Yeung et al. (Yeung et al. 2017) design an agent for learning to label noisy web data so that they can select right examples for training a classifier. Zhang et al. (Zhang, Wang, and Zhang 2018) propose to choose sufficient data pairs for multi-shot pedestrian re-identification by training an agent. Liu et al. (Liu et al. 2019) introduce a policy network for adjusting a margin parameter in the loss function to learn more discriminative features from imbalanced face datasets. In this work, we train an agent with deep Q-learning to select more representative and accurate pseudo-labeled data.

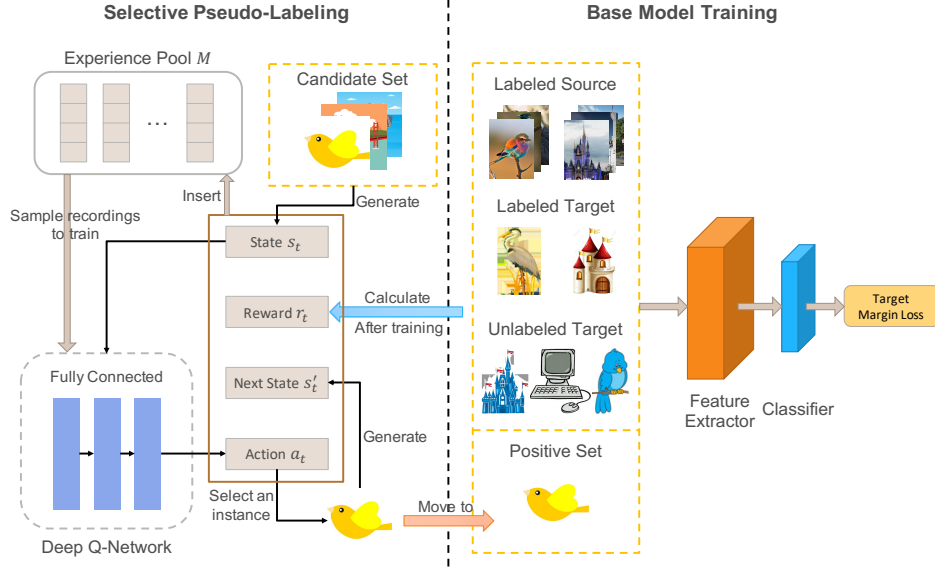


Figure 2: An overview of our method. We use the proposed target margin loss for the base model training. The candidate set consists of several samples pseudo-labeled by the pre-trained base model. Then we train an agent (deep Q-network) by deep Q-learning to select appropriate pseudo-labeled samples and move them to positive set for the improved model training. Specifically, we generate state  $s_t$  from the candidate set as the input of the agent and get an output action  $a_t$  to select an pseudo-labeled instance. Then the state transfers to the next state  $s'_t$  and the reward  $r_t$  will be calculated after an epoch of the improved model training. As for the training of the agent, we use an experience pool  $M$  to collect a series of recordings  $\{(s_i, a_i, r_i, s'_i)\}$  defined in section 3.2 and sample batches of recordings from  $M$  to train the deep Q-network

### 3 Method

For semi-supervised domain adaptation, we have a sufficient labeled dataset from the source domain,  $\mathcal{D}_s = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{N_s}$ . In the target domain, we only have a limited number of labeled instances  $\mathcal{D}_t = \{(\mathbf{x}_i^t, y_i^t)\}_{i=1}^{N_t}$ , but a large set of unlabeled instances  $\mathcal{D}_u = \{(\mathbf{x}_i^u)\}_{i=1}^{N_u}$ . The goal is to train a good prediction model on all these available data  $\mathcal{D}_s$ ,  $\mathcal{D}_t$  and  $\mathcal{D}_u$  and evaluate it on  $\mathcal{D}_u$  with the hidden true labels, as described in (Saito et al. 2019). In this section, we present a novel selective pseudo-labeling method for semi-supervised domain adaptation. The method is based on a deep Q-learning framework and a target margin loss. The framework of the proposed method is depicted in Fig. 2. First, we use the proposed target margin loss to train a CNN consisting of a feature extractor  $F$  and a classifier  $C$  for a  $K$ -class classification problem. Then, we generate pseudo-labels for the unlabeled samples in the target domain based on the trained CNN classifier. Finally, we alternately train an agent with deep Q-learning and use the agent to select pseudo-labeled samples for the CNN training.

#### 3.1 Target Margin Loss

Large margin loss functions (Liu et al. 2017; Wang et al. 2018a; Deng et al. 2019) (based on traditional softmax loss function) effectively make CNN features more discriminative. For semi-supervised domain adaptation, there is a gap in feature distributions between domains and the number of labeled samples in the target domain is much smaller than

in the source domain. Therefore, we propose to add a relative angular margin to the loss on the target labeled data, by contrast to the loss on the source labeled data. This can be considered as making the decision region separations more aligned with the target domain feature distribution. Then the proposed target margin loss can be formulated as follows:

$$\mathcal{L}_{tml} = -\frac{1}{N_s} \sum_{i=1}^{N_s} \log \frac{e^{s \cos \theta_{y_i}^s}}{\sum_{j=1}^K e^{s \cos \theta_j^s}} - \frac{1}{N_t} \sum_{i=1}^{N_t} \log \frac{e^{s \cos(\theta_{y_i}^t + m)}}{e^{s \cos(\theta_{y_i}^t + m)} + \sum_{j=1, j \neq y_i}^K e^{s \cos \theta_j^t}} \quad (1)$$

where  $\cos \theta_j = \frac{\mathbf{w}_j^T}{\|\mathbf{w}_j\|} \cdot \frac{F(\mathbf{x}_i)}{\|F(\mathbf{x}_i)\|}$  is the cosine similarity between the  $i$ -th instance feature vector and the  $j$ -th class weight. The  $\cos(\theta_{y_i} + m)$  can be computed by the addition formula of the trigonometric functions with the values of  $\cos \theta_{y_i}$  and the margin  $m$ .  $s$  is a re-scaling constant.

In addition, the suitable class separation can benefit from taking the unlabeled target domain data into account. To this end, we adopt the entropy loss to cluster the target unlabeled features into the corresponding decision regions, which is written as follows:

$$\mathcal{L}_{ent} = -\frac{1}{N_u} \sum_{i=1}^{N_u} \sum_{j=1}^K p(y = j | \mathbf{x}_i^u) \log p(y = j | \mathbf{x}_i^u) \quad (2)$$

where  $p(y = j | \mathbf{x}_i^u)$  represents the probability of categorizing  $\mathbf{x}_i^u$  to class  $j$ . Note that the samples should be clus-

tered around the representative centers of the corresponding classes to decrease the entropy, resulting in the desired discriminative features.

Combining Eq. (1) and Eq. (2) provides the following formulation of our final semi-supervised loss function:

$$\mathcal{L} = \mathcal{L}_{tml} + \alpha \mathcal{L}_{ent} \quad (3)$$

where  $\alpha$  is a hyper-parameter that balances the target margin loss and the entropy loss.

### 3.2 Selective Pseudo-Labeling by Reinforcement Learning

Pseudo-labeling target unlabeled data by the CNN can generate many mis-labeled samples which can cause serious harm to the subsequent learning process. To address this issue, selective pseudo-labeling is used in many works, which selects the most confident unlabeled samples for pseudo-labeling. This sample selection strategy however is subject to the problem that the selected samples are generally “easy” ones or belong to “easy” classes. As a result, the selected samples could not represent the distribution of the target domain well. To address this drawback, we propose to use reinforcement learning to select more representative and accurate pseudo-labeled samples.

We formulate the problem of selecting pseudo-labeled samples as a Markov Decision Process (MDP), described by  $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$  as the states, actions, transitions and rewards respectively, and train an agent to select pseudo-labeled samples. We define a candidate set  $\mathcal{D}_c$  which consists of pseudo-labeled samples to be selected and is initially randomly sampled from  $\mathcal{D}_u$  with pseudo-labels, and a positive set  $\mathcal{D}_p$  which is composed of the selected pseudo-labeled samples and initialized to be empty. During our CNN training, a series of reinforcement learning samples will be generated, which can be represented as  $\{(s_i, a_i, r_i, s'_i)\}$ . Here,  $s_i \in \mathcal{S}$  is the state and  $r_i \in \mathcal{R}$  is the reward.  $a_i \in \mathcal{A}$  is the action taken by the agent at state  $s_i$ , which is equivalent to selecting a pseudo-labeled sample from the candidate set  $\mathcal{D}_c$  and moving it to the positive set  $\mathcal{D}_p$ .  $s'_i \in \mathcal{S}$  represents the next state which the agent turns to through the action  $a_i$ . Then, we alternately train the agent by using the reinforcement learning samples and use the agent to select pseudo-labeled samples for our CNN training.

**States.** We consider that the representative ability and accuracy of a pseudo-labeled target instance can be related to three parts, which are itself, the data with labels in  $\mathcal{D}_t$  and  $\mathcal{D}_p$ , and the unlabeled data in  $\mathcal{D}_u$ . Note that pseudo-labeled instances are selected to make the current distribution of labeled data similar to the distribution of unlabeled data. Therefore, we formulate the state as a concatenation of three vectors, dependent on  $\mathcal{D}_c$ ,  $\mathcal{D}_t \cup \mathcal{D}_p$  and  $\mathcal{D}_u$ , respectively. For the first part, given the candidate set  $\mathcal{D}_c = \{(\mathbf{x}_i^c, \hat{y}_i^c)\}_{i=1}^{N_c}$ , we use a vector  $[F(\mathbf{x}_i^c)^T, C(\mathbf{x}_i^c)^T] \in \mathbb{R}^{d+K}$  to represent each instance, where  $F(\mathbf{x}_i^c)$  denotes the  $d$ -dimensional feature vector of instance  $\mathbf{x}_i^c$  extracted by the feature extractor  $F$ ,  $C(\mathbf{x}_i^c)$  denotes the softmax output of the classifier  $C$ . The

entire vector of this part is a flattened concatenation of all the instances in  $\mathcal{D}_c$ . After taking an action to move an instance from  $\mathcal{D}_c$  to  $\mathcal{D}_p$ , we replace the selected instance with a zero-valued vector. For the second part, we also use a vector  $[F(\mathbf{x}_i^{tp})^T, C(\mathbf{x}_i^{tp})^T] \in \mathbb{R}^{d+K}$  to represent each instance in  $\mathcal{D}_t \cup \mathcal{D}_p$ . The difference is that we calculate the average vectors in each class and then concatenate them to a vector which has a dimension of  $K \times (d + K)$ . The last part is a vector represented by the instances in  $\mathcal{D}_u$  with the same operation as the second part. Thus, the state  $s_i$  is a flattened concatenation of these three parts.

**Actions.** We define the action as selecting one instance from the current candidate set  $\mathcal{D}_c$ . For each state  $s_i$ , the agent takes an action  $a_i$  to select the  $a_i$ -th instance in  $\mathcal{D}_c$  and move it to  $\mathcal{D}_p$ . The number of actions is equivalent to the number of instances in  $\mathcal{D}_c$ , i.e.,  $N_c$ .

**Rewards.** The rewards should reflect whether the actions taken by the agent are appropriate or not. In other words, selecting more representative and accurate pseudo-labeled instances should lead to positive rewards, and vice versa. We first define a metric function to measure the representative ability and accuracy, which can be formulated as follows:

$$\varphi(\mathbf{x}_i, \hat{y}_i) = \log p_c(y = \hat{y}_i | \mathbf{x}_i) + \beta \log p_f(y = \hat{y}_i | \mathbf{x}_i) + \lambda \Delta_e \quad (4)$$

where  $\beta$  and  $\lambda$  are hyper-parameters.  $p_c(y = \hat{y}_i | \mathbf{x}_i)$  represents the probability of the pseudo class  $\hat{y}_i$  predicted by the classifier. It indicates the confidence of the prediction. As for  $p_f(y = \hat{y}_i | \mathbf{x}_i)$ , we take advantage of the target labeled data and define it as follows:

$$p_f(y = \hat{y}_i | \mathbf{x}_i) = \frac{e^{s \cos \langle F(\mathbf{x}_i), \mathbf{z}_{\hat{y}_i}^{tp} \rangle}}{\sum_{j=1}^K e^{s \cos \langle F(\mathbf{x}_i), \mathbf{z}_j^{tp} \rangle}} \quad (5)$$

where  $\mathbf{z}_j^{tp}$  represents the feature center of the  $j$ -th class in target labeled and current positive set  $\mathcal{D}_t \cup \mathcal{D}_p$ , which can be formulated as follows:

$$\mathbf{z}_j^{tp} = \frac{\sum_{i=1}^{N_t} F(\mathbf{x}_i) \mathbb{I}(y_i = j) + \sum_{i=1}^{N_p} F(\mathbf{x}_i) \mathbb{I}(\hat{y}_i^p = j)}{\sum_{i=1}^{N_t} \mathbb{I}(y_i = j) + \sum_{i=1}^{N_p} \mathbb{I}(\hat{y}_i^p = j)} \quad (6)$$

$\mathbb{I}$  is the indicator function. Therefore,  $p_f(y = \hat{y}_i | \mathbf{x}_i)$  is the softmax output of the cosine distance between  $\mathbf{x}_i$  and the feature center of its pseudo class  $\hat{y}_i$  in  $\mathcal{D}_t \cup \mathcal{D}_p$ .

The first two terms reflect the confidence of the pseudo-label prediction through two aspects, i.e., the output of the classifier and the similarity with the feature center of the pseudo class in target domain. Since the classifier is more dependent on the source domain data, we add the second term to specifically take the distribution of the target domain into consideration so that the metric function can evaluate the accuracy of a target pseudo-labeled sample better. We also add a third term  $\Delta_e$ , which represents the decrease of the entropy of the target unlabeled data and can be formulated as follows:

$$\Delta_e = H - H' \quad (7)$$

where  $H$  and  $H'$  represent the entropy at state  $s_i$  and the next state  $s'_i$  respectively and can be calculated in the same way as the  $\mathcal{L}_{ent}$  in Eq. (2). In other words, we first calculate  $H$  at state  $s_i$  and then add a pseudo-labeled sample according to the action  $a_i$  for one-epoch training. After the training, we calculate  $H'$  and derive  $\Delta_e$ . In order to make the entropy not affected by  $\mathcal{L}_{ent}$ , we only use  $\mathcal{L}_{tml}$  in Eq. (1) to optimize the model during the one-epoch training. Note that the more representative the selected sample is, the more the entropy will decrease. Therefore, larger  $\Delta_e$  means stronger representative ability, and vice versa. In addition, we perform a log operation on the first two terms to keep these three terms at the same scale.

Directly using the metric function as reward can result in very small differences between the rewards of good and bad actions. Hence, we propose to use the final reward function defined as follows:

$$r_i = \begin{cases} +1, & \varphi(\mathbf{x}_i, \hat{y}_i) > \tau \\ -1, & \varphi(\mathbf{x}_i, \hat{y}_i) \leq \tau \end{cases} \quad (8)$$

where  $\tau$  is a threshold and we set it to  $(1 + \beta) \log(0.9)$  in our experiments. We use this binary reward instead of the metric function in order to provide the agent more explicit guidance.

**Deep Q-learning.** We apply deep Q-learning (Mnih et al. 2015) to learn policies for selecting pseudo-labeled instances. For each state and action  $(s_i, a_i)$ , the Q function  $Q(s_i, a_i)$  can represent the discounted accumulated rewards for the state and action. Given a reinforcement learning training sample  $(s_i, a_i, r_i, s'_i)$ , the target value of  $Q(s_i, a_i)$  can be calculated as follows:

$$V_i = r_i + \gamma \max_{a'_i} Q(s'_i, a'_i) \quad (9)$$

where  $\gamma$  is a discount factor to decide the importance of future accumulated reward compared with the current reward. During the training, we iteratively update the deep Q-network by:

$$\Omega \leftarrow \Omega - \varepsilon \sum_i \frac{dQ(s_i, a_i)}{d\Omega} (Q(s_i, a_i) - V_i) \quad (10)$$

where  $\Omega$  represents the parameters of the Q-network. As for the entire model training, we alternately train the classification network and the Q-network. The details of our training strategy are summarized in Algorithm 1.

We simply use a three-layer fully connected network as the deep Q-network. Each fully connected layer is followed by a ReLU activation function. When the agent (i.e., the deep Q-network) is called to select a pseudo-labeled instance, it will output an action  $a_t$  using a policy as follows:

$$a_t = \arg \max_a Q(s_t, a) \quad (11)$$

where  $s_t$  is the current state representation.

## 4 Experiments

### 4.1 Datasets and Baselines

**Datasets.** We perform our experiments on three datasets, DomainNet (Peng et al. 2019), Office-31 (Saenko et al.

---

### Algorithm 1 Selective pseudo-labeling by deep Q-learning

---

**Input:** Source labeled set  $\mathcal{D}_s$ , target labeled set  $\mathcal{D}_t$  and target unlabeled set  $\mathcal{D}_u$

**Output:** Feature extractor  $F$ , classifier  $C$  and deep Q-network  $Q$

- 1: Pre-train  $F$  and  $C$  with  $\mathcal{D}_s$ ,  $\mathcal{D}_t$  and  $\mathcal{D}_u$  by optimizing the loss in Eq. (3);
  - 2: Initialize the positive set  $\mathcal{D}_p = \emptyset$ ;
  - 3: Initialize the experience pool  $M = \emptyset$ ;
  - 4: **while** not converge **do**
  - 5:   Assign pseudo-labels to the unlabeled data in  $\mathcal{D}_u$  by current  $F$  and  $C$ ;
  - 6:   Copy the parameters of  $F$  and  $C$  to  $F'$  and  $C'$ ;
  - 7:   Initialize  $\mathcal{D}_c$  with random pseudo-labeled samples from  $\mathcal{D}_u$  and generate the state  $s_0$ ;
  - 8:   **while**  $\mathcal{D}_c \neq \emptyset$  **do**
  - 9:     Get an output action  $a_t$  using Eq. (11) with the current state  $s_t$  as input;
  - 10:     Update  $\mathcal{D}_c$  and  $\mathcal{D}_p$  by taking the action  $a_t$ ;
  - 11:     Update  $F'$  and  $C'$  with  $\mathcal{D}_s$ ,  $\mathcal{D}_t$  and  $\mathcal{D}_p$  by optimizing the loss in Eq. (1);
  - 12:     Generate the next state  $s'_t$ ;
  - 13:     Calculate the reward  $r_t$  by Eq. (8) with  $F'$  and  $C'$ ;
  - 14:     Insert the recording  $(s_t, a_t, r_t, s'_t)$  into  $M$ ;
  - 15:     Sample a batch of recordings  $\{(s_i, a_i, r_i, s'_i)\}$  from  $M$  to update the deep Q-network  $Q$  by Eq. (10);
  - 16:     **if**  $r_t < 0$  **then**
  - 17:       Break;
  - 18:     **end if**
  - 19:   **end while**
  - 20:   Update  $F$  and  $C$  with  $\mathcal{D}_s$ ,  $\mathcal{D}_t$ ,  $\mathcal{D}_u$  and  $\mathcal{D}_p$  by optimizing the loss in Eq. (3).
  - 21: **end while**
- 

2010) and Office-Home (Venkateswara et al. 2017). DomainNet is a large-scale domain adaptation benchmark dataset, which contains six domains. Compared with the other two datasets, it has much more data with 345 classes. Office-31 is a widely used dataset for visual domain adaptation, with 31 categories collected from three different domains: Amazon (A), Webcam (W) and DSLR (D). Office-Home is another domain adaptation dataset in office and home settings, which is more difficult than Office-31. It consists of four distinct domains Artistic images (A), Clipart (C), Product images (P) and Real-World images (R). For the SSDA setting and domain adaptation scenarios, we follow the strategies in (Saito et al. 2019). We randomly select three labeled samples per class as the labeled training target samples to form a three-shot SSDA setting. Due to some noisy domains and categories in DomainNet, we pick 4 domains and 126 categories for DomainNet experiments. Following (Saito et al. 2019), we form 7 adaptation scenarios for testing with the 4 domains, Real (R), Clipart (C), Painting (P) and Sketch (S). As for Office-31, we construct 2 scenarios with Amazon as the target domain since Webcam and DSLR do not have enough samples for effective evaluation.



**Baselines.** S+T baseline directly trains a model using the labeled source data and labeled target data without unlabeled target data. For the UDA methods (DANN (Ganin and Lempitsky 2015), ADR (Saito et al. 2018a), CDAN (Long et al. 2018)) as baselines, we modify their training strategies following (Saito et al. 2019) so that the models can be trained with all the labeled source set, labeled target set and unlabeled target set. ENT (Grandvalet and Bengio 2005) is a baseline method using standard entropy minimization which trains a model to minimize the entropy of unlabeled target data. MME (Saito et al. 2019) is a model with the classifier and the feature extractor optimized by maximizing the conditional entropy of unlabeled target data and minimizing it respectively. We also design a baseline TML\_SPL with a selective pseudo-labeling strategy to compare with our reinforcement learning based selective pseudo-labeling method. TML\_SPL uses our target margin loss for the entire training and selects the most confident pseudo-labeled samples with 0.9 as the threshold to assist to train an improved model.

## 4.2 Implementation Details

As for the architecture of the networks, we adopt ResNet-34 (He et al. 2016) for experiments on DomainNet and VGG-16 (Simonyan and Zisserman 2014) for experiments on Office-31 and Office-Home, finetuned from ImageNet pre-trained models (Russakovsky et al. 2015). All our experiments are implemented in PyTorch (Paszke et al. 2017). We adopt mini-batch SGD with momentum of 0.9 and the learning rate adjusting schedule as (Ganin et al. 2016). The weight decay is set at 0.0005. As for the  $s$  and margin  $m$  in Eq. (1), a very small  $s$  or a very large  $m$  can make the model difficult to converge while a very large  $s$  or a very small  $m$  can make the margin ignored during training. Therefore, we choose an appropriate pair of values 30 and 0.5 for  $s$  and  $m$  respectively following previous large margin works. As for the trade-off parameters, we set  $\alpha$  in Eq. (3) at 0.1 to keep the two losses at a similar scale so that we can prevent any loss from being ignored during the training.  $\beta$  and  $\lambda$  balance the second and the third terms in Eq. (4). The first two terms in Eq. (4) are both logarithmic forms and the third term is the decrease of the entropy. Thus, we set  $\beta$  and  $\lambda$  at 1 and 0.1 to keep the three terms at a similar scale so that all the terms can make sense.

For the deep Q-learning, we apply the  $\varepsilon$ -greedy strategy (Mnih et al. 2015) and the experience replay strategy (Lin 1992). The  $\varepsilon$ -greedy strategy allows the deep Q-network to randomly select an action with  $\varepsilon$  as the probability and  $\varepsilon$  changes from 1 to 0 during the training. We use this strategy for the reason that the output by the deep Q-network at early stage does not reflect the reward and the deep Q-network needs more diverse training samples. The experience replay strategy means that we use an experience pool  $M$  to collect a series of recordings  $\{(s_i, a_i, r_i, s'_i)\}$  and sample batches of the recordings from  $M$  to train the deep Q-network. It can make the deep Q-network learn from both current and past information. Our deep Q-network is composed of three fully connected layers, with two hidden layers of 1024 and 512 units respectively. The discount factor  $\gamma$  in Eq. (9) is set to 0.9.

Table 1: Comparisons for CML and TML using DomainNet based on ResNet-34

| Method     | R to C      | R to P      | P to C      | C to S      | S to P      | R to S      | P to R      | Mean        |
|------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| CML        | 66.3        | 63.8        | 67.2        | 58.8        | 61.0        | 57.2        | 71.7        | 63.7        |
| TML (Ours) | <b>72.5</b> | <b>71.6</b> | <b>72.9</b> | <b>61.0</b> | <b>67.7</b> | <b>62.8</b> | <b>79.2</b> | <b>69.8</b> |

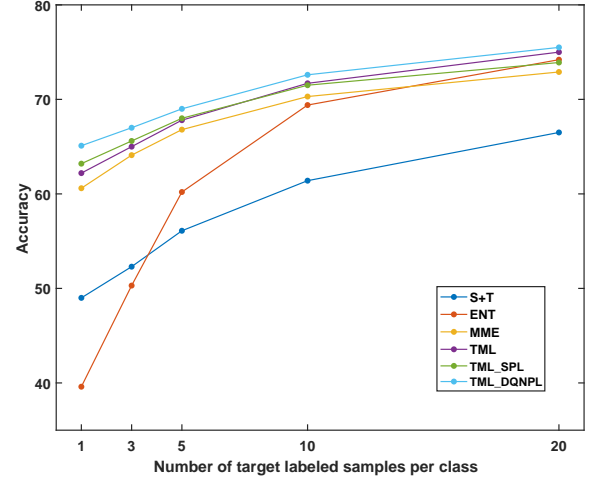


Figure 3: Accuracy vs the number of labeled samples per class in target domain

## 4.3 Validation Experiments

In order to show the effectiveness of our method, we design several validation experiments. We pick a scenario in DomainNet as an example scenario with Real as the source domain and Clipart as the target domain.

**Extending Margin to Source Domain.** Margin loss can make features more discriminative while our target margin loss only adds a margin on target domain. We extend the margin parameter to source domain to form a complete margin loss (CML) for comparison, which has a similar formulation to the original margin loss and can be written as:

$$\mathcal{L}_{cml} = \frac{1}{N_{s+t}} \sum_{i=1}^{N_{s+t}} \log \frac{e^{s \cos(\theta_{y_i}^t + m)}}{e^{s \cos(\theta_{y_i}^t + m)} + \sum_{j=1, j \neq y_i}^K e^{s \cos \theta_j^t}} \quad (12)$$

where the  $s + t$  means all the data in labeled source set and labeled target set. The CML method replaces the  $\mathcal{L}_{tml}$  in Eq. (3) with  $\mathcal{L}_{cml}$ . The results are shown in Table 1. Our TML method performs much better than CML though the difference between the two methods is quite small. The reason can be that if both the labeled source data and the labeled target data is constrained by the margin then the decision boundary will still be more dependent on the source domain with much more data.

**Varying Number of Target Labeled Samples.** We verify the number of labeled samples in the target domain from 1 to 20 per class to explore the performance of our method in

Table 2: Results on the 7 scenarios in DomainNet based on ResNet-34. The top-performing method in each scenario is bold and the best method without pseudo-labeling is underlined.

| Method                           | R to C      | R to P      | P to C      | C to S      | S to P      | R to S      | P to R      | Mean        |
|----------------------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| S+T                              | 60.0        | 62.2        | 59.4        | 55.0        | 59.5        | 50.1        | 73.9        | 60.0        |
| DANN (Ganin and Lempitsky 2015)  | 59.8        | 62.8        | 59.6        | 55.4        | 59.9        | 54.9        | 72.2        | 60.7        |
| ADR (Saito et al. 2018a)         | 60.7        | 61.9        | 60.7        | 54.4        | 59.9        | 51.1        | 74.2        | 60.4        |
| CDAN (Long et al. 2018)          | 69.0        | 67.3        | 68.4        | 57.8        | 65.3        | 59.0        | 78.5        | 66.5        |
| ENT (Grandvalet and Bengio 2005) | 71.0        | 69.2        | 71.1        | 60.0        | 62.1        | 61.1        | 78.6        | 67.6        |
| MME (Saito et al. 2019)          | 72.2        | 69.7        | 71.7        | <u>61.8</u> | 66.8        | 61.9        | 78.5        | 68.9        |
| TML (Ours)                       | <u>72.5</u> | <u>71.6</u> | <u>72.9</u> | 61.7        | <u>67.7</u> | <u>62.8</u> | <u>79.2</u> | <u>69.8</u> |
| TML_SPL                          | 73.2        | 72.2        | 73.3        | 62.1        | 68.5        | 63.4        | 80.0        | 70.4        |
| TML_DQNPL (Ours)                 | <b>75.8</b> | <b>74.5</b> | <b>75.1</b> | <b>64.3</b> | <b>69.7</b> | <b>64.4</b> | <b>82.6</b> | <b>72.3</b> |

Table 3: Results on the 12 scenarios in Office-Home based on VGG-16. The top-performing method in each scenario is bold and the best methods without pseudo-labeling are underlined.

| Method           | R to C      | R to P      | R to A      | P to R      | P to C      | P to A      | A to P      | A to C      | A to R      | C to R      | C to A      | C to P      | Mean        |
|------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| S+T              | 49.6        | 78.6        | 63.6        | 72.7        | 47.2        | 55.9        | 69.4        | 47.5        | 73.4        | 69.7        | 56.2        | 70.4        | 62.9        |
| DANN             | 56.1        | 77.9        | 63.7        | 73.6        | 52.4        | 56.3        | 69.5        | 50.0        | 72.3        | 68.7        | 56.4        | 69.8        | 63.9        |
| ADR              | 49.0        | 78.1        | 62.8        | 73.6        | 47.8        | 55.8        | 69.9        | 49.3        | 73.3        | 69.3        | 56.3        | 71.4        | 63.0        |
| CDAN             | 50.2        | 80.9        | 62.1        | 70.8        | 45.1        | 50.3        | 74.7        | 46.0        | 71.4        | 65.9        | 52.9        | 71.2        | 61.8        |
| ENT              | 48.3        | 81.6        | 65.5        | 76.6        | 46.8        | 56.9        | 73.0        | 44.8        | 75.3        | 72.9        | 59.1        | 77.0        | 64.8        |
| MME              | <u>56.9</u> | 82.9        | 65.7        | 76.7        | 53.6        | 59.2        | <u>75.7</u> | <u>54.9</u> | 75.3        | 72.9        | <u>61.1</u> | 76.3        | 67.6        |
| TML (Ours)       | <u>56.9</u> | <u>83.2</u> | <u>67.0</u> | 76.8        | <u>54.5</u> | <u>59.9</u> | <u>75.7</u> | <u>54.9</u> | <u>75.9</u> | <u>73.2</u> | <u>61.1</u> | <u>77.5</u> | <u>68.1</u> |
| TML_SPL          | 55.4        | 82.1        | 67.1        | 76.5        | 55.3        | 60.7        | 75.5        | 53.0        | 75.9        | 73.4        | 60.4        | 77.6        | 67.7        |
| TML_DQNPL (Ours) | <b>58.4</b> | <b>84.0</b> | <b>69.1</b> | <b>78.5</b> | <b>56.8</b> | <b>61.7</b> | <b>77.0</b> | <b>55.9</b> | <b>77.1</b> | <b>74.5</b> | <b>61.9</b> | <b>78.8</b> | <b>69.5</b> |

Table 4: Results on the 2 scenarios in Office-31 based on VGG-16. The top-performing method in each scenario is bold and the best methods without pseudo-labeling are underlined.

| Method           | W to A      | D to A      | Mean        |
|------------------|-------------|-------------|-------------|
| S+T              | 73.2        | 73.3        | 73.3        |
| DANN             | 75.4        | 74.6        | 75.0        |
| ADR              | 73.3        | 74.1        | 73.7        |
| CDAN             | 74.4        | 71.4        | 72.9        |
| ENT              | 75.4        | 75.1        | 75.3        |
| MME              | 76.3        | <u>77.6</u> | 77.0        |
| TML (Ours)       | <u>76.6</u> | <u>77.6</u> | <u>77.1</u> |
| TML_SPL          | 75.7        | 77.2        | 76.5        |
| TML_DQNPL (Ours) | <b>77.5</b> | <b>78.8</b> | <b>78.2</b> |

different settings. As illustrated in Fig. 3, our TML method can outperform MME and ENT in all the settings with varying number of target labeled samples while MME gradually performs worse than the simple ENT baseline as the number increasing. Furthermore, when the target labeled samples are much enough, the confidence based selective pseudo-labeling method TML\_SPL doesn't work well and can even hurt the original model. Our reinforcement learning based selective pseudo-labeling method TML\_DQNPL can always make progress to the base model due to the representative and accurate selected pseudo-labels.

#### 4.4 Results

The results of our main experiments on the DomainNet dataset are shown in Table 2. As mentioned in (Saito et al.

2019), the UDA methods cannot improve the performance in some cases. Compared with the present state-of-the-art SSDA method MME (Saito et al. 2019), our method with only target margin loss (TML) can perform better except for only one case where it performs similarly to MME. On the basis of TML, our final method TML\_DQNPL with deep Q-network for selective pseudo-labeling can outperform the baseline TML\_SPL with selective pseudo-labeling by confidence, which demonstrates that our deep Q-network can help select more representative and accurate pseudo-labels.

The results on Office-Home and Office-31 are shown in Table 3 and Table 4 respectively. With these small-scale datasets, our TML method also has better performance than MME in most cases and can perform the same as MME in other cases. In addition, we observe that TML\_SPL can hurt the performance in some cases while our TML\_DQNPL also makes a progress. The potential reason can be that adding mis-labeled target samples for training causes more damage to the model when the number of the original training samples is small in small-scale datasets. Therefore, these comparisons can further confirm the effectiveness of our method.

## 5 Conclusions

We propose a novel reinforcement learning based selective pseudo-labeling method for semi-supervised domain adaptation (SSDA). We first design a target margin loss for our base model training, which can make the feature distribution closer to the target domain and improve the discriminative ability. Then we apply deep Q-learning to train an agent to select more representative and accurate pseudo-labeled samples for our improved model training. Our method obtains

competitive results on several domain adaptation benchmarks and outperforms the present state-of-the-art methods. In addition, the training of the deep Q-network is unrelated to the architecture of base model and will not change the training strategy of base model so that the proposed pseudo-labeling agent can be combined with other advanced methods to help train improved models in the SSDA setting.

## References

- Ao, S.; Li, X.; and Ling, C. X. 2017. Fast generalized distillation for semi-supervised domain adaptation. In *AAAI*.
- Chen, C.; Xie, W.; Huang, W.; Rong, Y.; Ding, X.; Huang, Y.; Xu, T.; and Huang, J. 2019. Progressive feature alignment for unsupervised domain adaptation. In *CVPR*.
- Deng, J.; Guo, J.; Xue, N.; and Zafeiriou, S. 2019. Arcface: Additive angular margin loss for deep face recognition. In *CVPR*.
- Donahue, J.; Hoffman, J.; Rodner, E.; Saenko, K.; and Darrell, T. 2013. Semi-supervised domain adaptation with instance constraints. In *CVPR*.
- Ganin, Y.; and Lempitsky, V. 2015. Unsupervised domain adaptation by backpropagation. In *ICML*.
- Ganin, Y.; Ustinova, E.; Ajakan, H.; Germain, P.; Larochelle, H.; Laviolette, F.; Marchand, M.; and Lempitsky, V. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research (JMLR)* 17(1): 2096–2030.
- Gong, B.; Shi, Y.; Sha, F.; and Grauman, K. 2012. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*.
- Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NeurIPS*.
- Grandvalet, Y.; and Bengio, Y. 2005. Semi-supervised learning by entropy minimization. In *NeurIPS*.
- He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.
- Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NeurIPS*.
- Li, S.; and Deng, W. 2018. Deep facial expression recognition: A survey. *arXiv preprint arXiv:1804.08348*.
- Lin, L.-J. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning* 8(3-4): 293–321.
- Liu, B.; Deng, W.; Zhong, Y.; Wang, M.; Hu, J.; Tao, X.; and Huang, Y. 2019. Fair loss: Margin-aware reinforcement learning for deep face recognition. In *ICCV*.
- Liu, W.; Wen, Y.; Yu, Z.; Li, M.; Raj, B.; and Song, L. 2017. Sphereface: Deep hypersphere embedding for face recognition. In *CVPR*.
- Long, M.; Cao, Y.; Wang, J.; and Jordan, M. I. 2015. Learning transferable features with deep adaptation networks. In *ICML*.
- Long, M.; Cao, Z.; Wang, J.; and Jordan, M. I. 2018. Conditional adversarial domain adaptation. In *NeurIPS*.
- Long, M.; Wang, J.; Ding, G.; Sun, J.; and Yu, P. S. 2013. Transfer feature learning with joint distribution adaptation. In *ICCV*.
- Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540): 529.
- Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch. In *NeurIPS-W*.
- Pei, Z.; Cao, Z.; Long, M.; and Wang, J. 2018. Multi-adversarial domain adaptation. In *AAAI*.
- Peng, X.; Bai, Q.; Xia, X.; Huang, Z.; Saenko, K.; and Wang, B. 2019. Moment matching for multi-source domain adaptation. In *ICCV*.
- Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)* 115(3): 211–252.
- Saenko, K.; Kulis, B.; Fritz, M.; and Darrell, T. 2010. Adapting visual category models to new domains. In *ECCV*.
- Saito, K.; Kim, D.; Sclaroff, S.; Darrell, T.; and Saenko, K. 2019. Semi-supervised domain adaptation via minimax entropy. In *ICCV*.
- Saito, K.; Ushiku, Y.; Harada, T.; and Saenko, K. 2018a. Adversarial dropout regularization. In *ICLR*.
- Saito, K.; Watanabe, K.; Ushiku, Y.; and Harada, T. 2018b. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*.
- Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- Sun, B.; Feng, J.; and Saenko, K. 2016. Return of frustratingly easy domain adaptation. In *AAAI*.
- Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *CVPR*.
- Venkateswara, H.; Eusebio, J.; Chakraborty, S.; and Panchanathan, S. 2017. Deep hashing network for unsupervised domain adaptation. In *CVPR*.
- Wang, H.; Wang, Y.; Zhou, Z.; Ji, X.; Gong, D.; Zhou, J.; Li, Z.; and Liu, W. 2018a. Cosface: Large margin cosine loss for deep face recognition. In *CVPR*.
- Wang, J.; Feng, W.; Chen, Y.; Yu, H.; Huang, M.; and Yu, P. S. 2018b. Visual domain adaptation with manifold embedded distribution alignment. In *ACM MM*.
- Wang, M.; and Deng, W. 2018a. Deep face recognition: A survey. *arXiv preprint arXiv:1804.06655*.



- Wang, M.; and Deng, W. 2018b. Deep visual domain adaptation: A survey. *Neurocomputing* 312: 135–153.
- Wang, Q.; Bu, P.; and Breckon, T. P. 2019. Unifying unsupervised domain adaptation and zero-shot visual recognition. In *IJCNN*.
- Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; and He, K. 2017. Aggregated residual transformations for deep neural networks. In *CVPR*.
- Yao, T.; Pan, Y.; Ngo, C.-W.; Li, H.; and Mei, T. 2015. Semi-supervised domain adaptation with subspace learning for visual recognition. In *CVPR*.
- Yeung, S.; Ramanathan, V.; Russakovsky, O.; Shen, L.; Mori, G.; and Fei-Fei, L. 2017. Learning to learn from noisy web videos. In *CVPR*.
- Zhang, J.; Li, W.; and Ogunbona, P. 2017. Joint geometrical and statistical alignment for visual domain adaptation. In *CVPR*.
- Zhang, J.; Wang, N.; and Zhang, L. 2018. Multi-shot pedestrian re-identification via sequential decision making. In *CVPR*.
- Zhang, W.; Ouyang, W.; Li, W.; and Xu, D. 2018. Collaborative and adversarial network for unsupervised domain adaptation. In *CVPR*.